

# Het nauwkeurig bemeten van de werkelijke snelheid van speed-pedelecgebruikers met smartphones

**Olivier BUREZ**

Promotor(en): Emilia Motoasca

Co-promotor(en): (Ing. B. Rotthier)  
(Jeroen De Maeyer)

Masterproef ingediend tot het behalen van  
de graad van master of Science in de  
industriële wetenschappen: Energie  
Elektrotechniek

©Copyright KU Leuven

Zonder voorafgaande schriftelijke toestemming van zowel de promotor(en) als de auteur(s) is overnemen, kopiëren, gebruiken of realiseren van deze uitgave of gedeelten ervan verboden. Voor aanvragen i.v.m. het overnemen en/of gebruik en/of realisatie van gedeelten uit deze publicatie, kan u zich richten tot KU Leuven Technologiecampus Gent, Gebroeders De Smetstraat 1, B-9000 Gent, +32 92 65 86 10 of via e-mail [iiw.gent@kuleuven.be](mailto:iiw.gent@kuleuven.be).

Voorafgaande schriftelijke toestemming van de promotor(en) is eveneens vereist voor het aanwenden van de in deze masterproef beschreven (originele) methoden, producten, schakelingen en programma's voor industrieel of commercieel nut en voor de inzending van deze publicatie ter deelname aan wetenschappelijke prijzen of wedstrijden.

# **Dankwoord**

Dank aan mezelf

Dank aan een ander

# **Abstract**

Speed pedelecs kunnen fietsers assisteren tot 45km/u. Maar rijden fietsers wel degelijk zo snel? Welk verschil zit er tussen de werkelijke , gemiddelde en mogelijke snelheid? Om veel data te kunnen verzamelen is gekozen de GPS op gebruikers hun smartphone te gebruiken. De ruwe data van apps zijn onderling vergeleken alsook deze van verschillende gsms met eenzelfde app. De metingen zijn samen met een meetfiets uitgevoerd waarmee de ruwe data vergeleken werd. Uit de ruwe data vallen de coördinaten met tijdstippen te halen. Er is gekeken naar methodes om coordinaten in afstand om te zetten. Om fouten uit de ruwe data te filteren enonzekerheden te verkleinen zijn verscheidene filters gestest.. Er is een script in Scilab geschreven, met Vincenty-formule en Kalman-filter, waarmee de data kan verwerkt worden zodat positie, snelheid, versnelling en hun gemiddeldes kunnen bepaald worden. Met deze informatie kan via een ray casting algoritme gekeken worden met welke snelheid in bepaalde (snelheids)zones gereden wordt.

# Inhoudsopgave

<b>1 Inleiding</b>	<b>1</b>
<b>2 Hoe werkt een GPS?</b>	<b>2</b>
2.1 Basisprincipe van de werking . . . . .	2
2.1.1 De satelliet . . . . .	3
2.1.2 GPS-Onvanger . . . . .	3
2.1.3 Tussenstations . . . . .	4
2.1.4 Nauwkeurigheid van de GPS . . . . .	5
<b>3 Kiezen van de app</b>	<b>7</b>
3.1 Criteria . . . . .	7
<b>4 Proeven met apps op smartphones</b>	<b>9</b>
4.1 Gebruikte apparatuur . . . . .	9
4.2 Uitvoering van de meting . . . . .	10
<b>5 Verwerkingswijze van de ruwe data</b>	<b>11</b>
5.1 Data halen uit de apps en meetfiets . . . . .	11
5.2 Verkregen data . . . . .	12
5.2.1 Ruwe data van Runkeeper . . . . .	12
5.2.2 Ruwe data van Map My Ride . . . . .	12
5.2.3 Ruwe data van Move! Bike Computer . . . . .	13
5.2.4 Ruwe data van Meetfiets . . . . .	14
5.3 Rekenvoorbeeld met coördinaten . . . . .	15
5.3.1 Kaartprojecties . . . . .	15
5.3.2 Afstandbepaling via bolafstanden . . . . .	18
5.3.3 Vergelijking berekeningswijzen . . . . .	20
<b>6 Testen met apps en meetfiets</b>	<b>24</b>
6.1 Resolutie en precisie van de afstand bij de meetfiets en de apps . . . . .	24
6.1.1 Resolutie van de meetfiets . . . . .	25
6.1.2 Precisie van apps in literatuur . . . . .	26

6.1.3	Resolutie van apps uit ruwe data . . . . .	28
6.2	Verschil in beginpositie . . . . .	28
6.3	Aantal datapunten . . . . .	30
6.3.1	Effect gsm op de sample tijd . . . . .	30
6.3.2	Effect app op de sample tijd . . . . .	31
6.4	Hoogteverschil bij apps . . . . .	31
6.5	Verschill in gemiddelde en maximale snelheid . . . . .	33
6.6	Stops . . . . .	33
6.7	Schommelingen snelheid . . . . .	35
6.8	Gelijk leggen curves . . . . .	36
6.8.1	Het gelijk leggen van het beginpunt van een rit . . . . .	36
6.8.2	Schalingsfactor in de tijd . . . . .	37
6.9	Resultaten apps . . . . .	38
6.10	Metingen met 1 app en meerdere gsms . . . . .	39
6.11	Besluiten apps . . . . .	41
6.11.1	Keuze apps . . . . .	41
6.11.2	Mogelijke verbeteringen . . . . .	41
<b>7</b>	<b>Smoothers</b>	<b>43</b>
7.1	Lokaal gemiddelde smoother . . . . .	43
7.2	Gauss kernel smoother . . . . .	44
7.2.1	Bepalen van de spreiding $\sigma_{optimaal}$ . . . . .	49
7.3	De Kalman-filter . . . . .	49
7.3.1	Algemeen overzicht . . . . .	49
7.3.2	De initiële/vorige staat . . . . .	51
7.3.3	Nieuwe staat / voorspelde waarde . . . . .	52
7.3.4	Nieuwe meting . . . . .	54
7.3.5	Update met nieuwe meting . . . . .	54
7.3.6	Output van de geüpdate staat . . . . .	55
7.4	Resultaten met de Kalman-filter . . . . .	55
7.4.1	RMSE op Kalmanresultaat . . . . .	55
7.4.2	$\chi^2$ op Kalmanresultaat . . . . .	58
7.5	Besluit smoothers . . . . .	63
<b>8</b>	<b>Snelheidszones</b>	<b>64</b>
8.1	Zones herkennen aan de hand van coördinaten . . . . .	64
8.2	Ingeven van grenzen en gebruiken script . . . . .	66
8.2.1	Plotten route gebruiker app . . . . .	66
8.2.2	Gemeente/stads mobiliteitsdienst bereiken . . . . .	67
8.2.3	afbakenen snelheidszones . . . . .	67

8.2.4 Invoegen in Excel-bestand . . . . .	69
<b>9 Verwerken van ritten van speed pedelec gebruikers</b>	<b>70</b>
9.1 Algemeen overzicht van te zetten stappen en gebruiken programma's . . . . .	70
9.2 Resultaten van speed-pedelecgebruikers . . . . .	72
9.3 Resultaten bij verschillende voertuigen . . . . .	73
<b>10 Algemeen besluit</b>	<b>75</b>
<b>A Keuze apps</b>	<b>78</b>
<b>B Bronnen apps</b>	<b>80</b>
<b>C Testparcour</b>	<b>82</b>
<b>D Stappenplan voor het gebruiken van de apps</b>	<b>84</b>
<b>E Van app naar Scilab</b>	<b>86</b>
E.0.1 Omzetten van .gpx naar .xls . . . . .	86
E.0.2 Openen en bewerken .xls bestand in Scilab . . . . .	87
<b>F De cosinus-regel</b>	<b>88</b>
<b>G De Haversine formule</b>	<b>89</b>
<b>H Formule van Vincenty</b>	<b>90</b>
<b>I Standaardafwijking</b>	<b>92</b>
<b>J Afwijking van afstand door bochten bij apps</b>	<b>93</b>
<b>K Voorbeelden ritten apps</b>	<b>94</b>
<b>L Gauss Curve</b>	<b>96</b>
<b>M Stelling van Jordan</b>	<b>97</b>
<b>N Script voor het plotten van de polygonen</b>	<b>98</b>
<b>O Scilab script 'Algemene filter' in detail</b>	<b>99</b>
<b>P Scilab script 'Herkennen snelheidszones' in detail</b>	<b>106</b>
<b>Q Poster</b>	<b>107</b>

# Lijst van figuren

2.1 DOP . . . . .	4
2.2 Verbetering nauwkeurigheid . . . . .	6
2.3 Schatting snelheid met mogelijke afwijking van 3m . . . . .	6
5.1 Type projecties . . . . .	16
5.2 Lambertprojectie voor europa . . . . .	17
5.3 Verschuiving van de assen voor Belgische Lambertprojectie . . . . .	17
5.4 De vergelijking tussen berekeningswijzen . . . . .	20
5.5 4 Berekeningswijzen afstand . . . . .	21
5.6 Sequentiële afstandsberekening tussen 2 punten . . . . .	22
6.1 accuraatheid vs precisie . . . . .	25
6.2 accuraatheid vs precisie . . . . .	25
6.3 Histogram met de sample tijd van meetfiets bij 9400 datapunten . . . . .	26
6.4 Piek in snelheid bij de meetfiets tijdens stilstand . . . . .	27
6.5 Beginpunten . . . . .	29
6.6 hoogte van alle apps . . . . .	32
6.7 hoogtes apps afzonderlijk resp. ct, mmr, rk . . . . .	32
6.8 Start stop rit met Runkeeper . . . . .	34
6.9 Experimenten met stoptijden . . . . .	35
6.10 Rit zonder bijsnijden aanvang en einde . . . . .	36
6.12 Rit zonder schalingsfactor . . . . .	37
6.11 Rit met bijsnijden aanvang en einde . . . . .	37
6.13 Rit met schalingsfactor . . . . .	38
6.14 Meerdere gsms die gebruik maken van Runkeeper . . . . .	40
7.1 Smoother op basis van lokaal gemiddelde . . . . .	44
7.2 Epanechnikov, Gaussische & Tricube kernel . . . . .	45
7.3 visualisatie smoothing met Gaussische kernel . . . . .	46
7.4 Gauss-kernel smoother op coördinaten en snelheid . . . . .	47
7.5 wijziging van $\sigma$ bij de Gauss-kernel smoother . . . . .	48

7.6 Schema Kalman-filter . . . . .	50
7.7 Onzekerheden bij de Kalmanfilter . . . . .	51
7.8 Verduidelijking $w_k$ . . . . .	53
7.9 RMSE ifv versnelling in kalmanfilter . . . . .	56
7.10 Extreme versnellingen in Kalmanfilter . . . . .	57
7.11 kritieke grootheid . . . . .	59
7.12 Kritieke grootheid op $\chi^2$ -curve . . . . .	59
7.13 Het effect van df op de $\chi^2$ -curve . . . . .	60
7.14 Histogram van de snelheden in bereiken van 1 km/u . . . . .	61
7.15 Histogram df=6 . . . . .	62
7.16 Histogram van de snelheden met bereiken $\pm$ RMSE 3,4 km/u . . . . .	62
 8.1 Zone afbakening op basis van 4 coordinaten . . . . .	64
8.2 Zone 50 Gent . . . . .	65
8.3 Voorbeeld code raytracing mbv stelling Jordan . . . . .	65
8.4 Stelling van Jordan . . . . .	66
8.5 Weergave traject gpx-bestand . . . . .	67
8.6 Snelheidszones Gent . . . . .	68
8.7 Aanduiden zone 30 polygonen . . . . .	69
8.8 Excel-bestand met snelheidszones . . . . .	69
 9.1 Algemeen overzicht te doorlopen stappen voor verwerking data . . . . .	71
9.2 snelheden bij verschillende voertuigen . . . . .	72
9.3 Snelheid speed pedelec in zone 30 . . . . .	73
9.4 snelheden bij verschillende voertuigen . . . . .	74
 A.1 Lijst van 20 apps . . . . .	79
B.1 Bronnen van de 20 pregeselecteerde apps . . . . .	81
C.1 Parcour afgefiest voor de testen . . . . .	83
L.1 Standaard Gauss curves . . . . .	96

# Lijst van tabellen

4.1 Vergelijking gsms . . . . .	9
5.1 Ruwe data van Runkeeper . . . . .	12
5.2 Ruwe data van Map My Ride . . . . .	13
5.3 Ruwe data van Move! Bike Computer . . . . .	13
5.4 meetfietsdata 1e toestel . . . . .	14
5.5 meetfietsdata 2e toestel . . . . .	14
5.6 Standaardafwijking berekningswijzen . . . . .	21
6.1 Gemiddelde afwijking van beginpunten van apps . . . . .	29
6.2 verschil tussen gsms . . . . .	29
6.3 Aantal samples voor 3 apps en 2 gsms . . . . .	30
6.4 resultaten apps . . . . .	31
6.5 resultaten apps . . . . .	38
6.6 resultaten Runkeeper op 2 gsms . . . . .	40
7.1 resultaten smoother . . . . .	48
7.2 Optimale versnelling volgens RMSE bij rk . . . . .	57
7.3 Optimale versnelling volgens RMSE bij ctg . . . . .	58
7.4 Optimale versnelling volgens RMSE bij speed pedelec . . . . .	58
7.5 Vergelijking histogrammen na kalmanfilter met $\chi^2$ en df=18 . . . . .	60
7.6 Vergelijking histogrammen na kalmanfilter met $\chi^2$ en df=6 . . . . .	62

# **Hoofdstuk 1**

## **Inleiding**

De voorbije jaren is de elektrische fiets erg populair geworden. De pedelec die de fietser assisteert tot 25 km/u leek volgens de wetgeving eenvoudig te classeren. De snelheden lagen niet hoger dan bij de gewone fietser. Bij de intro van de speed pedelec lag dit anders. Deze fietsen kunnen fietsers assisteren tot 45 km/u. Als er bij brommers een rijbewijs verwacht wordt , moet dit dan bij deze fiets ook? Welke typegoedkeurig moeten deze fietsen krijgen? Waar mag deze fietser rijden op de openbare weg? Deze thesis maakt deel uit van de doctoraatsstudie van Ing. B. Rotthier die deze zaken op dit onverkende terrein zal kaderen.

Dit werk heeft tot doel het de snelheden waarmee de speed pedelec-gebruikers in kaart te brengen. De aanpak poogde zoveel mogelijk nationalistische data van de gebruikers te verzamelen op een eenvoudige manier. Via de smartphone die ingeburgerd is werd via app's data gelogd. Maar hoe juist is deze data en moet deze nog verwerkt worden vooraleer deze gebruiksklaar is?

Er wordt in dit onderzoek eerst gekeken naar het werkingsprincipe van de GPS en de randfactoren. Vervolgens worden onderling vergeleken op verschillende smartphones met als referentie een meetfiets. Eens de keuze voor de apps gevallen was werd er gezocht naar een manier om de data te smoothen. Aan de hand van coördinaten werd gekeken of herkend kon worden wanneer fietsers zich in zone 30 bevonden. Met het Scilab-script om deze stappen te doorgaan werd tot slot gekeken naar enkele nationalistische ritten met speed pedelecs om een eerste blik te werpen op een aantal resultaten.

## **Hoofdstuk 2**

# **Hoe werkt een GPS?**

Het doel van dit hoofdstuk is om data via GPS verkregen te verwerken. Om waarde aan data verkregen door een GPS-toestel te kunnen hechten moeten we een basiskennis hebben over de GPS. Hieronder wordt besproken wat de werking van de GPS is, de types communicatie die er met GPS te stellen zijn, de invloedsfactoren, wat de nauwkeurigheid is en welke instellingen belangrijk zijn voor een optimale werking.

### **2.1 Basisprincipe van de werking**

Wanneer er opzoekwerk over de GPS gedaan wordt, kan men al snel overspoeld worden door technische informatie zonder die te kunnen plaatsen. Hieronder wordt een kort overzicht gegeven over de benamingen, werking en fysische beperking van de GPS.

GPS staat voor global positioning system. Omdat GPS op het moment van dit werk gebruikt wordt en dit een ingeburgerde term is zal in de loop van dit onderzoek het woord GPS gebruikt worden en niet GNSS<sup>1</sup>. Een minimum van 24 satellieten op 20,2km hoogte <sup>2</sup>, die met 95% zekerheid werken(?), hebben tot doel de positie van een gebruiker te bepalen op de aarde. Dit aantal zorgt voor een constante dekking van elk gebied op aarde door minimaal 4 satellieten. Elke satelliet zendt radiogolven van laag vermogen (20-50 Watt)<sup>3</sup> op verschillende frequenties uit (vooraf bepaalde L1, L2, etc.). Elk heeft een eigen unieke code (Y-code). De signalen bevatten informatie over de locatie van de satelliet en het tijdstip waarop het signaal verstuurd is. Het gebruik van de gewone burger is afgesteld op L1. Andere lijnen zijn voorzien van ‘anti-spoofing’ die ongeautoriseerde toegang verhindert. Wanneer een gebruiker het signaal ontvangt op zijn toestel vergelijkt dit toestel de tijd tussen het versturen en ontvangen van het signaal en kan op deze wijze zijn afstand tot de satelliet bepalen. Met deze informatie kan er een straal rond de satelliet getrokken worden met de afstand waarop de gebruiker zich ergens bevindt. Wanneer er 3 satellieten zijn is het mogelijk 1 gezamelijk snijpunt te hebben tussen deze bollen. De golven gaan door wolken, glas en plastic, maar niet door solide objecten. De satelliet stuurt zijn data rechtstreeks of via een tussenstation naar de ontvanger/gebruiker. We kunnen het geheel dus in 3 stukken opsplitsen om hun invloed op de

---

<sup>1</sup>eigenlijk is het correcter van GNSS(Global Navigation Satellite System) te spreken gezien GPS Amerikaans is. Zo is er ook GLASNOSS, de Russische tegenhanger, beschikbaar. Rond 2019 zullen GALILEO(Europees) en Beidou(Chinees) ook volledig operationeel zijn.

<sup>2</sup>Tegenwoordig zetten ze 31 satellieten in om aan hun gestelde eisen te voldoen

<sup>3</sup>Voor radio is dit 100.000 Watt.

nauwkeurigheid te schetsen.

### 2.1.1 De satelliet

De satellieten en hun posities zijn het onderdeel dat de berekening van een gebruiker's positie mogelijk maakt. Er wordt kort besproken hoe deze plaatsbepaling in zijn werk gaat. Hiermee zal een deel van de fouten in de meetresultaten verklaard kunnen worden.

De satelliet verstuur zijn informatie via radiogolven. Hieronder worden de mogelijke oorzaken genoemd die effect hebben op de nauwkeurigheid(?).

- Klok : Ondanks het feit dat elke satelliet uitgerust is met een atoomklok zijn er verschillen tussen elke satelliet.
- Atmosferisch : De ionosfeer zit tussen de gebruiker en de satelliet. Deze heeft elektrisch geladen deeltjes die de snelheid van het signaal kunnen vertragen/buigen. In de troposfeer zit water die hetzelfde effect heeft. Je kan dit voorstellen als bij de breking van licht in water.
- Multipath : Signalen komen niet altijd rechtstreeks tot de gebruiker, maar kunnen bvb eerst op gebouwen kaatsen en dan naar de gebruiker. Dit vertraagt wederom het signaal waardoor het moeilijk is de tijd/korste afstand tot de satelliet te bepalen.
- Ephemeris : Ephemeris gaat in deze context over de aantrekking van de zon en maan die de satelliet uit zijn baan kunnen laten wijken.
- Het aantal beschikbare satellieten : hoe meer satellieten beschikbaar zijn, hoe kleiner de foutmarge zal zijn. Bij een zekere hoeveelheid die volstaat zullen extra satellieten dan weer geen noemenswaardige bijdrage leveren (?).
- DOP : Dilution of Precision. De onderlinge positie van de satellieten heeft invloed op de nauwkeurigheid. Wanneer ze verder uit elkaar staan kunnen ze een beter geometrisch resultaat geven. Dit wordt duidelijk voorgesteld in Figuur 2.1.
- SA : dit staat voor Selective Availability(?). Het Amerikaans leger had ervoor gekozen om het signaal voor de gewone gebruiker minder accuraat te maken voor strategische redenen. Dit is na 2000 wel afgeschaft. Het leger gebruikt nu net als de wetenschap DGPS<sup>4</sup> om de posities nauwkeurig te bepalen.

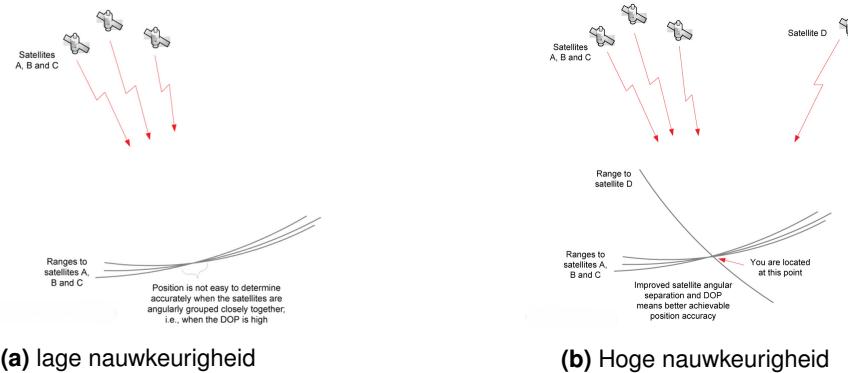
Geen van deze invloedsfactoren kunnen gemanipuleerd worden door de standaard gebruiker. Wat de gebruiker wel in de hand heeft wordt in de volgende sectie besproken.

### 2.1.2 GPS-Ontvanger

De ontvanger is het toestel dat de gebruiker in handen heeft en kan gebruikt worden voor auto's of toegespitst is op fietsers of wandelaars. Deze paragraaf gaat over de afwijkingen die veroorzaakt kunnen worden door de ontvanger.

---

<sup>4</sup>deze term wordt uitgelegd bij 2.1.3Tussenstations



Figuur 2.1: DOP

### Oorzaken afwijking door ontvanger

We kunnen eigenlijk stellen dat voor ons de fout tussen de atoomklokken van de satellieten te verwaarlozen is in vergelijking met de klok van de ontvanger. Dit valt op te lossen door ofwel een atoomklok te gebruiken, wat veel te duur is, of gebruik maken van enkele technieken die de fout deels wiskundig wegwerken. Er wordt gebruik gemaakt van een timer die start van het ontvangst van het eerste signaal en berekent hoe lang het duurt voor het volgende toekomt. De GPS-ontvanger is vaak maar accuraat tot 1% van een bit-tijd(?). Een bit-tijd is de tijd die nodig is om een bit te versturen van een Network Interface Card(NIC). Wanneer het verschil tussen de tijd dat het commando gegeven wordt en de bit effectief de NIC verlaat. Wanneer de bit-tijd op 10 nanoseconden gesteld wordt en verondersteld wordt dat de signalen aan de snelheid van het licht gaan kan de fout 3 meter bedragen. Wanneer er meer satellieten zijn zal de foutmarge verkleinen met trilateratie(dit is een ander woord voor het proces van positiebepaling dat zojuist is uitgelegd). Bij de ontvanger zijn er dus 3 parameters die de nauwkeurigheid beïnvloeden. De bit-tijd, de interne klok en het wiskundig model om de fouten weg te werken (oa. ook de atmosferische en multipath fouten). Wanneer er apps onderling vergeleken worden zal dit laatste hen wellicht onderscheiden. Men kan veronderstellen dat bij een groot verschil in prijsklasse van gsm de kans groter is dat de inwendige klok beter zal zijn en bijgevolg de resultaten nauwkeuriger.

### Instellingen op gsm

Op een gsm kan men vaak kiezen tussen 2 opties van GPS. Zo heb je A-GPS en GPS A-GPS is een ondersteunende dienst die ervoor zorgt dat de gsm weet waar de posities van de satellieten te vinden zijn. Zo wordt tijd gewonnen omdat de gsm deze zelf niet hoeft te localiseren. Om deze op te halen moet er wel dataverbinding zijn. Je kan ook voor gewone GPS kiezen, maar deze wijze kan langer duren om de minimaal <sup>5</sup> satellieten te localiseren.

#### 2.1.3 Tussenstations

Tussenstations zijn ingevoerd om de nauwkeurigheid te verhogen bij positiebepaling. De gsm zal niet gebruik maken van al deze methodes, maar om de precisie van de GPS te kunnen duiden in literatuur moet we weten vanuit welke basis vertrokken wordt. Er wordt gepoogd met dit deel de

<sup>5</sup>1 voor de tijd, 3 voor de positie

een beeld te vormen van de grootste onnauwkeurigheid die in rekening zal gebracht worden in de berekeningen.

*DGPS* : Differential GPS maakt gebruik van 2 ontvangers. Een ‘base station’ waarvan de positie precies gekend is en een bewegende ontvanger. De data wordt tussen beide vergeleken en zorgt zo voor een verhoogde nauwkeurigheid. Dit wordt vaak voor wetenschappelijke geografische doeleinden gebruikt en heeft een veel hogere precisie dan de ‘handheld-GPS’.

*SBAS* : Staat voor Satellite-Based Augmentation Systems. Het bestaat uit stations op de aarde waarvan de locatie precies gekend is. Het resultaat wordt verbeterd door de signalen onderling te vergelijken. WAAS was het eerste dergelijke systeem en is gevestigd in de VS ter ondersteuning van hun GPS. Gezien Europa tot nog toe ook gebruik maakt van GPS hebben ze ook besloten stations te plaatsen onder de naam EGNOS.

*PPK-RTK* : Post Processed Kinematic of Real Time Kinematic. Dit is een speciale versie van DGPS. Ze maakt gebruik van de fase-verschuiving bij gecodeerde signalen.

Er zijn nog meer types, maar deze zijn ofwel te duur, ofwel niet praktisch toepasbaar voor het bedoelde experiment. Er zijn dus veel factoren om in rekening te brengen.

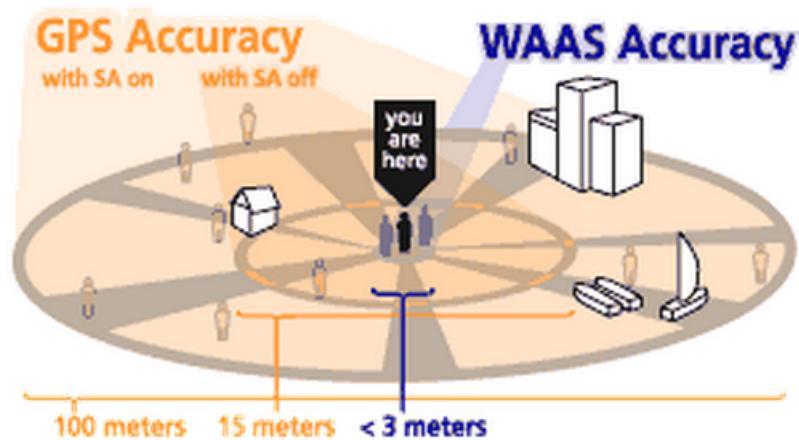
#### 2.1.4 Nauwkeurigheid van de GPS

Zoals eerder vermeld moet opgelet worden om geen data over de nauwkeurigheid voor het jaar 2000 te raadplegen. Niet enkel omdat AS is afgeschaft(?), maar ook omdat de technologie steeds verbeterd. EGNOS verbetert de nauwkeurigheid van 17 tot 3 m(?). RTK verhoogt deze nog eens met een factor 10 tot 100. gsm gebruikt RTK helaas niet. Kort samengevat geeft dit de accuraatheid weer(?):

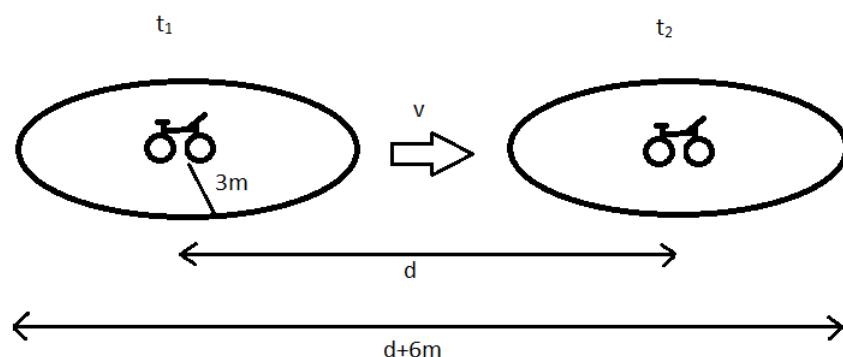
- Basis bestaat uit de basis GPS met SA,  $\pm 100\text{m}$
- Zonder SA  $\pm 15\text{ m}$
- Met SBAS (WAAS)  $\pm 1\text{-}3\text{ m } (?)$

Uit een GPS performantie analyse in 2014 (?) werd gesteld dat de verticale foutmarge bij 95% van de gevallen binnen de 4.684 m ligt en de horizontale 3.351 m. De verschillen tussen de horizontale en verticale hoogte zijn te verklaren door de eerder vermelde DOP.

Als we veronderstellen dat de GPS een sample tijd (dit is de tijd tussen 2 sequentiële metingen) heeft van 1s en er een afwijking van  $\pm 3\text{ m}$  kan zijn op de meting, dan kan de resulterende snelheid een afwijking van 6 m/s hebben (zie Figuur 2.3), ofwel 21.6 km/u. Dit is een probleem voor snelheden met de fiets. Bij een grotere afgelegde afstand zal de procentuele fout kleiner zijn. Dit kan kort verklaard worden. Met  $v_0 = d/t$  als ideale/theoretische snelheid en  $v=(d + 2 * fout)/t = v_0 + 2 * fout/t$  als reële snelheid waarbij de fout 3m is, kan gesteld worden dat bij een verhoging van d en het aandeel van de fout in v gelijk blijft. De grootte van de fout tov  $v_0$  verkleint dus bij een grotere d en dus zo ook het percentage.



Figuur 2.2: Verbetering nauwkeurigheid



Figuur 2.3: Schatting snelheid met mogelijke afwijking van 3m

## **Hoofdstuk 3**

# **Kiezen van de app**

Omdat het doel zoveel mogelijk nationalistische data verzamelen is, is gegaan naar de apps op smartphones. Nu volgt de volgende stap. De vraag stellen: " wat willen we van de app? ". In de volgende paragraaf worden verschillende criteria aangehaald voor de keuze, waarna er met een select groepje testen worden zal worden uitgevoerd. Deze apps zullen onderling vergeleken worden bij latere metingen. Ook de gsms waarmee ze gebruikt worden, hun invloed op de werking zal onder de loep genomen worden. Omdat er nog geen betrouwbare bronnen zijn die waardeoordelen over de apps geven, wordt er deels in het duister getast bij de keuze. Toch worden er enkele eisen gesteld aan de app die het mogelijk moet maken om bewerkingen met de data uit te voeren.

### **3.1 Criteria**

De criteria voor het kiezen van de app zijn gericht op zo laag mogelijke kosten, en zo'n correct mogelijke data over de snelheid. Hieronder wordt in volgorde van belang een lijst met criteria gegeven :

- Is de gemeten data downloadbaar?**

Indien de data niet ter beschikking gesteld wordt, zal er navenant niet met de metingen gewerkt kunnen worden.

- Wat is de kostprijs?**

Om de app zo toegankelijk mogelijk te maken zal de voorkeur gegeven worden aan apps die gratis verkrijgbaar zijn.

- Is de app zowel op Andriod als iOS af te spelen?**

Vele apps kunnen enkel op Android of enkel op iOS(gebruikt bij iPhones) afgespeeld worden gezien ze verschillende OS gebruiken. Er wordt gekozen voor apps die op beide kunnen functioneren.

- Meet de app de helling/hoogte?**

Indien de app een verschil in hoogte meet kan er een betere inschatting van het geleverde vermogen door fietser en motor gedaan worden en de afstand zelf.

- Is er een mogelijkheid tot verbinding met bluetooth?**

Wanneer bluetooth beschikbaar is, bestaat de mogelijkheid om extra meettoestellen toe te

voegen. Zo zijn er meettoestellen voor hartslag, cadance en omwentelingssnelheid van het wiel.

- **Zijn er extra betalende features?**

Wanneer de extra features betalend zijn kan dit gaan over de extra meettoestellen die met bluetooth aangesloten kunnen worden, maar gaat ook vaak over het gemak van de app voor de gebruiken. Zo kan je je prestatie met vrienden vergelijken, of jezelf een trainingschema opstellen. Deze laatsten zijn van minder belang voor het onderzoek.

- **Wat is de gebruiksvriendelijkheid van de app?**

De gebruiksvriendelijkheid van de app zal een proefpersoon sneller aanzetten tot het gebruik van de app. Elke app probeert dit natuurlijk wel te verzorgen voor zijn gebruikers. Dus staat dit laag op het lijstje van criteria. Deze kan bekijken worden a.d.h.v. de score die een app krijgt in de appstore.

- **Wat is de grootte van het bestand?**

Sommige gebruikers hebben onvoldoende ruimte op hun gsm om de app te installeren. De groottes van de apps gaan niet boven de 30MB, dus zal dit voor slechts weinigen een probleem zijn.

In Bijlage A is een tabel gegeven waarin de apps met hun kwalificaties vermeld staan. Er is besloten om met de volgende apps verder te gaan<sup>1</sup> :

- Map My Ride (mmr)
- Move! Bike Computer (mbc)
- Cyclingtracker (ct)
- Runkeeper (rk)
- Cycling Tracks GPS (ctg)
- Sports Tracker (st)

Deze apps stelden allemaal hun ruwe data ter beschikking, waren gratis en meetten hoogtes. Deze criteria bleken genoeg om het aanbod uit te dunnen tot de hierboven vermelde apps.

---

<sup>1</sup>Runtastic, Ride with GPS & Strava bleken hun meetdata op dat moment niet prijs te geven.

## Hoofdstuk 4

# Proeven met apps op smartphones

Nadat de meetapparatuur nader is bepaald zal worden besproken hoe de metingen in hun werk gegaan zijn. Hoe de ruwe data er uit ziet en hoe ze verwerkt wordt wordt in het volgende hoofdstuk besproken. Pas daarna zal gekeken worden hoe de apps onderling vergeleken konden worden. Dit heeft als doel een app te selecteren en niet van meerdere de accuraatheid te moeten bepalen.

### 4.1 Gebruikte apparatuur

Er wordt kort geschatst welke gsms gebruikt werden en de werking van het toestel op de meetfiets.

Er worden meerdere gsms gebruikt bij de testen. De reden hiervoor zal later vermeld worden. In Tabel 4.1 worden de verschillen tussen de gsms gegeven. De bluetooth optie staat hierbij om te weten of ze uit te breiden zijn met extra hardware<sup>1</sup>

Eigenschappen van de gsms			
Merken	<b>Wiko</b>	<b>iPhone</b>	<b>Samsung</b>
Type	Goa	3	GT-S7390
Processor	Dual Core 1 GHz, Cortex-A7	1 GHz Cortex-A8	Dual Core 1 GHz
Mobiel internet	Android Navigatie	Android Navigatie	3G
Bluetooth	Ja	Ja	Ja
Ontvangmodule <sup>2</sup>	A-GPS	A-GPS	A-GPS

Tabel 4.1: Vergelijking gsms

De gsm die bij de eerste proef gebruik wordt is van het merk Wiko. Een van de goedkoopste Android modellen beschikbaar op dit moment.

Het meettoestel op de meetfiets is na enkele testen met meerdere gsms kapot gegaan. Het initiële toestel meette ook de stroom en spanning die door de batterij geleverd werd en ook de cadance<sup>3</sup>

<sup>1</sup>de optie bluetooth kan de optie verschaffen om een relatief goedkoop alternatief bieden aan een extra sensor wanneer de apps op zichzelf ontoereikend zijn.

<sup>2</sup>dit is het aantal omwentelingen van de pedalen.

Het meettoestel bevatte geen verbinding met een inclino-meter<sup>4</sup>. Dit is vervangen geweest door een meettoestel met eenzelfde meetprincipe. Deze had wederom geen inclino-meter. Een magneet werd op een wiel gezet ter hoogte van een magneetsensor die op het frame van de fiets gemonteerd werd. De magneetsensor met een constante sample tijd  $t_s$  van 0.1 s registreerde of de magneet langst de magneetsensor kwam.

Er kon dus geen vergelijking gedaan worden tussen de apps en meetfiets om het juistheid van de hoogte te bekijken. Voor deze en andere redenen, die later worden besproken, werd telkens hetzelfde traject gefietst. Het afgelegde traject is zichtbaar in Bijlage C.

De gemeten data door het meettoestel op de meetfiets werd gelogd en rechtstreeks op een micro sd-kaart geschreven naar een .txt bestand. Dit kon handmatig op pc worden overgezet met behulp van een SD-kaart.

## 4.2 Uitvoering van de meting

Kort worden de stappen besproken om de metingen te starten en te beëindigen. Wanneer het nodig zou zijn om stapsgewijs het proces te volgen is er Bijlage D voorzien.

Zoals reeds vermeld is het noodzakelijk om steeds hetzelfde parcours af te leggen om de hoogtes te kunnen vergelijken. Het gekozen traject gegeven in Bijlage C ligt in Gent. De locatie is dicht bij de campus gekozen. Daarnaast is naar een fietsroute met relatief weinig verkeer gekeken. Langs het water is een groot fietspad voorzien. Het parcour bevat ook een noemenswaardige helling zodat deze herkend kan onderscheiden worden van een slechte meting. Om de apps te gebruiken moeten deze via wifi of mobiel internet gedownload worden en indien nodig een profiel aangemaakt worden.

De metingen gebeurden bij een lichtbewolkte of klare hemel en windsnelheden tussen 2-5 Beaufort. De meting werd gestart door alle apps aan te zetten bij stilstand op het begin/eindpunt<sup>5</sup> die op de Figuur is gegeven. Door enkele seconden te wachten voor vertrek kon een verschil van stilstand naar beweging gezien worden in de metingen is het eenvoudiger een gelijke start van de test te vinden in de data. Zo kan men een ‘cold start/stop’ vermijden<sup>6</sup>. De gsms werden aangezet en aan een open zak aan de buitenkant van een rugzak gestoken om geen verlies van data te hebben door een extra hindernis voor het signaal. Bij terugkomst werd steeds enkele tellen stilgestaan voor de apps uit te zetten. Hierna kon de data van de app verkregen worden op verschillende manieren. Hier zal op teruggekomen worden bij het volgende deel 5.1 *Data halen uit de apps en meetfiets*. Verscheidene metingen zijn mislukt door fout gebruik van de apps of foutmeldingen in de gsm of hardwarematige fouten bij de meetfiets.

<sup>4</sup> dit is een meettoestel dat de hoek meet waaronder het gehouden wordt.

<sup>5</sup> Het begin en eindpunt lagen op dezelfde plaats.

<sup>6</sup> Deze term wordt gebruikt wanneer apparatuur niet de tijd heeft gekregen de positie van de ontvanger nauwkeurig te bepalen bij starten en stoppen.

## **Hoofdstuk 5**

# **Verwerkingswijze van de ruwe data**

In het volgende stuk wordt respectievelijk besproken hoe we de data van de app kunnen halen, hoe ze verschijnt, wat ze betekend en hoe ze verwerkt kan worden. Er worden enkele apps apart besproken die onderling voldoende verschillen.

### **5.1 Data halen uit de apps en meetfiets**

De mogelijke wijzen om data van de app tot op de pc te krijgen worden kort besproken. Bij de app die uiteindelijk gekozen werd zullen de stappen meer in detail besproken worden.

Om de data te verkrijgen worden 4 methodes gebruikt.

- Na het aanmaken van een account wordt de data naar een online profiel gestuurd die via eenvoudige stappen uitlegt hoe je data kan downloaden.
- De app slaat de data op op de gsm waar deze via usb-aansluiting op de pc rechtstreeks kan overgezet worden.
- De app slaat de data wederom op op de gsm, maar biedt de mogelijkheid aan om het via mail of andere media te versturen.
- De app geeft de mogelijkheid om de data rechtstreeks naar *One Drive*<sup>1</sup> te sturen.

Vaak kan gekozen worden tussen verschillende type's data om te downloaden. Er werd indien mogelijk gekozen voor .gpx bestanden omdat deze overzichtelijk gelezen kunnen worden door Excel en daarna eenvoudiger verwerkbaar zijn in Scilab. Het is ook de meest voorkomende optie bij alle apps om ruwe data te krijgen over de rit. De meetfiets geeft zijn data na een hardwarematige verbinding met de pc in een kladblokbestand, die handmatig in een Excel-bestand gezet wordt om eenzelfde type bestand te krijgen als deze verkregen via de apps. Een handleiding over de stappen worden in detail besproken in de handleiding in Bijlage E

---

<sup>1</sup>dit is een online opslagruimte die bij het maken van een Google account beschikbaar wordt.

## 5.2 Verkregen data

Er zal kort een beeld gegeven worden van de data die verschijnt bij elk van deze applicaties. Dan worden de voor en nadelen besproken zodat we een gepaste verwerkingsmethode kunnen kiezen. Elke app heeft op zijn minst een kleine wijziging ten opzicht van anderen in het *.gpx* bestand. Hiervoor dient bij elk Scilab-script rekening mee gehouden te worden. Er is geen verschil in het *.gpx* bestand bij eenzelfde app en een verschillende gsm. Bij een verschillend OS<sup>2</sup> heeft dit ook geen invloed wanneer dezelfde app kan gebruikt worden.

### 5.2.1 Ruwe data van Runkeeper

De *.gpx* bestanden gehaald van het online profiel zijn omgezet in Excel-bestanden van het type *Excel 97-2003 Worksheet*<sup>3</sup>. Ze zijn makkelijk leesbaar en verwerkbaar in Scilab. In Tabel 5.1. worden de nuttige kolommen afgebeeld zoals ze te vinden zijn in het Excel-bestand, met enkele waarden erbij als voorbeeld. De getallen achter de titel zoals bij lat52 slaat op het nummer van de kolom en niet op de breedtegraad. Respectievelijk staan in de kolommen de digitale waarden van de latitude, deze van de longitude, de hoogte en de tijd.

Ruwe data van Runkeeper				
Lat52	Lon 53	ns1:ele54	ns1: time55	... <sup>4</sup>
51,050245	3,710118	9	2015-04-03T09:18:45Z	...
51,050315	3,710276	9	2015-04-03T09:19:17Z	...
51,050211	3,710328	9,1	2015-04-03T09:19:21Z	...

**Tabel 5.1:** Ruwe data van Runkeeper

De grootste uitdaging hier is dus het gebruiken van coördinaten om afstanden te bepalen. Daarna is het inlezen van de data en kunnen verweken relatief voor de hand liggend. De enige bijkomende moeilijkheid is dat de logging van verschillende apps en soms zelfs gsms niet overeen komen. Waardoor er niet een instant vergelijking kan gemaakt worden van snelheden gemeten tussen de gsms. In de loop van dit onderzoek heeft Runkeeper een update uitgevoerd waarbij de kolommen van plaats zijn veranderd. Er werd in de oudere versie veel ruimte voorzien voor mogelijke uitbreidingen via bluetooth. De laatste versie knipte deze lege kolommen weg.

### 5.2.2 Ruwe data van Map My Ride

In Tabel 5.2. zien we dat deze data gelijkaardig is aan deze van Runkeeper met de uitzondering dat hier de afgelegde afstand al berekend is en het tijdstip op een iets andere wijze genoteerd is. Deze data is ook van een online profiel gehaald.

<sup>2</sup>OS= Operating System. In deze context gaat over de software die de algemene werking bepaald op de gsm.

<sup>3</sup>deze versie is bij de gebruikte versie van Scilab leesbaar onder het commando `readxl`

Rupe data van Runkeeper				
ns1: Time	ns1: LatitudeDegrees	ns1:LongitudeDegrees	ns1:AltitudeMeters	ns1 : DistanceMeters2
2015-03-19T13:17:06.023000+00:00	51,04892885	3,71197466	10,89	208,2497559
2015-03-19T13:17:08.023000+00:00	51,04880004	3,712138601	10,75	226,6207123
2015-03-19T13:17:10.023000+00:00	51,04867371	3,712299511	10,55	244,6437378

**Tabel 5.2:** Ruwe data van Map My Ride

Data uit de apps Cyclingtracker , Strava, Cycling Tracks GPS en Sports tracker zijn gelijkaardig maar in andere volgorde of andere kolommen in het Excel-bestand. Dit zorgt voor kleine wijzigingen bij het inlezen van de ruwe data. ctg is de enige app die gebruik maakt van de data door te sturen naar One Drive. De andere apps stellen dit beschikbaar via hun site.

### 5.2.3 Ruwe data van Move! Bike Computer

De data die gegeven wordt in Tabel 5.3. is gehaald via usbverbinding van gsm naar pc. De kolommen bevatten geen titel en valt er dus maar uit te zoeken waar de getallen voor staan. De tabel zal soms met sprongen werken in rijen, aangeduid met ... om zaken waar schijnbaar patroon in zit of bij constant te ontdekken voor de lezer. Er werden geen beschrijvingen gegeven in de eerste rij van elke kolom. Er moet dus zelf uitgezocht worden waarvoor elke kolom staat.

Rupe data van Move! Bike Computer				
510.504	371.049	36,9	57.654	1,42677E+12
510.503	371.053	36,7	653.852	1,42677E+12
510.502	371.058	36,6	671.198	1,42677E+12
...	...	...	...	...
51.046	372.589	6	692.353	1,42677E+12
...	...	...	...	...
510.482	371.294	17,8	5.961	1,42677E+12

**Tabel 5.3:** Ruwe data van Move! Bike Computer

De 1e kolom geeft de latitude. Hier staan gehele getallen. Zoals men kan zien zijn sommige waarden in de eerste kolom plots kleiner dan de rest. Dit komt doordat app het kleinste digit niet weergeeft wanneer dit nul is. De reden hierachter zal een fout zijn in de code of foute afronding. Om dit tot zijn recht te laten komen zal eerst een vergelijking gemaakt moeten worden met de waarden en vermenigvuldigd met 10 of 100 indien nodig. En gelijkaardig probleem stelt zich in de 2e kolom, maar dit is niet in de tabel weergegeven.

De 2e kolom geeft de longitude. Desondanks dat deze een grootorde 10 kleiner is dat de latitude ( $51^{\circ}$  latitude en  $3^{\circ}$  longitude) is dit ook een getal van 6 cijfers.

De 3e kolom lijkt een snelheid te zijn, gezien de variërende curve en de waarde 0 wordt wanneer er een verschil van 0 is bij de opeenvolgende coördinaten. De eenheid voor de snelheid is duidelijk niet in km/u. De omzetting is op 2 manieren mogelijk. Zo kan er van m/s naar km/u gegaan worden door te delen met 36.000. Er kan een vergissing geweest zijn en een foutieve conversie van feet/u naar km/u  $ft * 10^4 / sec$ . Om deze om te zetten naar m/s moeten we dit vermenigvuldigen met 0,3048 en delen door 10.000.

Beiden geven geen bevredigend resultaat. Zonder duidelijkheid lijkt deze app ten dode opgeschreven nog voor we aan de testen beginnen.

De *4e kolom* gaat van 57.6 gestaag naar beneden tot -7,5 en klimt dan naar 21. De enige verklaring die voor deze waarden kan gegeven worden is de hoogte in meter.

De *laatste kolom* bevat getallen die als ze in Excel aangeduid worden telkens met 1000 verhogen (vb. 1426775026000). Het gaat hier om de tijd uitgedrukt in milliseconden. Dit kon worden achterhaald door naar de tijd van de hele rit te kijken. Het volledige getal kan op een datum wijzen of de start van de lancering van de app of een andere starttijd. Omdat de app zelf niet gebruikt zal worden is hier niet verder naar gezocht.

#### 5.2.4 Ruwe data van Meetfiets

De data zoals ze verkregen wordt door het meettoestel krijgt in Excelvorm een vorm zoals afgebeeld in Tabel ???. Hier stond ook geen titel bij de kolommen, maar de betekenissen hiervoor werden gegeven door de ontwerper. Er waren opties om stroom e.d. naar de motor van de fiets te meten in kolom 2 tem 4, maar deze waren nog niet afgesteld. Omdat dit niet nodig was bij de metingen was het geen probleem dat deze kolommen met 0 aangeduid werden. In de 1e kolom staat de tijd waarop gelogd wordt in ms. De 5e kolom gaat over de stroom in mA naar de fietsmotor komende van de batterij. De 6e kolom wordt de omwentelingssnelheid van het wiel in omw/min omgezet in km/u a.d.h.v. de straal van het wiel.

Ruwe data van de meetfiets					
581.440	0.00	0.0	0.00	54.35	19.14
591.450	0.00	0.0	0.00	76.92	19.29
601.450	0.00	0.0	0.00	56.82	19.24

Tabel 5.4: meetfietsdata 1e toestel

Zoals reeds vermeld is in de loop van het onderzoek het eerste meettoestel kapot gegaan. Het tweede toestel logde elke  $70 \pm 1$  ms. Tussen 2 detecties werd de snelheid berekend en constant weergegeven over de tijd tussen de 2 detecties. Een voorbeeld van deze data is gegeven in Tabel ???. De straal van het wiel werd bepaald als 0,35 m.

Ruwe data van de meetfiets	
12.844	5.74
12.913	5.74
12.984	5.74
13.054	5.21
13.125	5.21

Tabel 5.5: meetfietsdata 2e toestel

## 5.3 Rekenvoorbeeld met coördinaten

Er werd reeds een blik geworpen op de data die gegeven werden door de app. Slechts enkelen geven een afgelegde afstand. Waarop moeten we afgaan om de afstand te weten te komen indien dit niet gegeven is, en hoe nauwkeurig is deze? Om dit te weten te komen worden hier, via enkele voorbeelden, methodes besproken worden hoe de coördinaten omgezet kunnen worden in afstanden.

De eerste methode werd via de omzetting naar Lambert coördinaten op de kaart bepaald. De tweede methode maakte gebruik van bolcoördinaten en de 3e maakte gebruik van de afstand op een ellipsoïde. Op het einde van deze paragraaf werden de resultaten van elke methode vergeleken en werd een optimale omzettingswijze gekozen.

### 5.3.1 Kaartprojecties

In deze paragraaf wordt gekeken of het zinvol is om de afstand te bepalen op basis van kaarten. De kaarten van België worden het vaakst gegeven in Lambert-vorm.

Als we ons coördinaten voorstellen zijn deze meestal gegeven in graden-minuten-seonden (DMS ofwel degrees-minutes-seconds). Maar er zijn vele manieren om coördinaten voor te stellen. Ze worden gebruikt als globaal referentiesysteem, voor militair gebruik of lokale plaatsen op de aardbol<sup>5</sup>. Bij de gebruikte apps werd steeds gebruik gemaakt van decimale graden (DD ofwel decimal degrees). Gezien de aardbol geen bol maar een sferoïde is, is de afstand tussen 2 coördinaten geen lineair gegeven.

Aan de hand van een voorbeeld gaan we bekijken wat het verschil tussen begin en eindpositie is bij een test met de app Runkeeper. Het doel is bepalen hoe ver van elkaar deze punten liggen en de nauwkeurigheidsmarge daarop te bepalen.

- Beginpositie (DD) : Latitude = 51.020245 ; Longitude = 3.710118
- Eindpositie (DD) : Latitude = 51.050591 ; Longitude = 3.710231

Eerst en vooral zetten we ze om in DMS met behulp van de formule gegeven in vlg. (5.1) (Degrees-minutes-seconds)

In het Nederlands zijn geen termen voor de getallen voor de komma. In het Engels gebruikt men trunc(truncation). Mod staat voor modulus ofwel restwaarde. Eenvoudig uitgelegd, bestaat de formule uit 3 opeenvolgende stappen. De restwaarde van de vorige uitkomst —DD— wordt gedeeld door 60.

$$DMS = \begin{cases} D = \text{trunc}(DD) \\ M = \text{trunc}(\|DD\| * 60) \bmod 60 \\ S = (\|DD\| * 3600) \bmod 60 \end{cases} \quad (5.1)$$

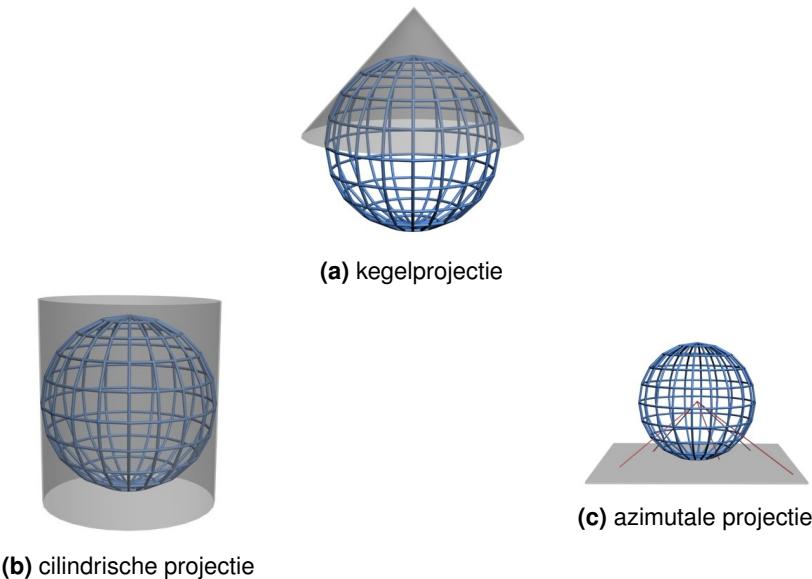
De voorbeeld waarden worden dan :

- Beginpositie (DMS) : Latitude = 51° 1' 12.882'' ; Longitude = 3° 42' 36.4248''
- Eindpositie (DMS) : Latitude = 51° 3' 2.1276'' ; Longitude = 3° 42' 36.8316''

<sup>5</sup>Andere voorstellingen van coördinaten zijn afgekort DDM,DMS,GARS,GEOREF,UTM,USNG,MGRS?.

Om de afstand te kunnen bepalen moeten we deze 2 posities op een kaart kunnen plaatsen. Elke kaart heeft zijn eigen voor-en nadelen qua nauwkeurigheid. Het is dus zinvol hier even op in te gaan.

De wereldbol voor te stellen op een plat vlak worden kan op veel manieren<sup>6</sup>, maar 3 springen naar voor. Ze worden op een kegelmantel, cilindermantel of een plat vlak geprojecteerd, en worden resp. kegelprojectie, cilinderprojectie en azimutale projectie genoemd.



**Figuur 5.1:** Type projecties

Daarnaast bestaan er nog conforme of hoekgetrouwe, equivalente en equidistante projecties.

- Conforme projectie : geeft een hoekgetrouw beeld
- Equivalente projectie : de verhouding van de oppervlaktes op kaart zijn gelijk aan deze in werkelijkheid ongeacht de schaal.
- Equidistante projectie : hierbij wordt geprojecteerd in de strikte zin van het woord, maar afstanden gemeten en overgebracht op de kaart. Door meerdere meridianen<sup>7</sup> op parallelcirkels<sup>8</sup> te projecteren of vice versa.

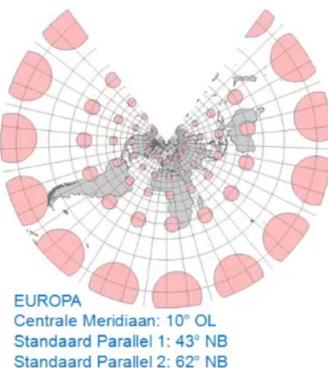
De Lambertprojectie, ofwel de ‘conforme conische projectie’, wordt vaak gebruikt om werelddelen af te beelden (zie Figuur 5.2). Ze is de basis voor de projecties die standaard zijn in België. Elke kaart heeft een kaartdatum. Een wijziging in datum geeft een nieuw cartografisch systeem aan. De aanpassingen worden in de loop van de tijd gedaan doordat men betere technieken heeft. Er zijn nog enkele redenen, maar deze zullen niet besproken worden gezien naast de kwestie ligt. Wat

<sup>6</sup>de meer dan 100 soorten projecties zijn te vinden op <http://www.progonos.com/furuti/MapProj/Normal/ProjTbl/projTbl.html>

<sup>7</sup>ook wel lengtecirkels genoemd

<sup>8</sup>ook wel breedtecirkel genoemd

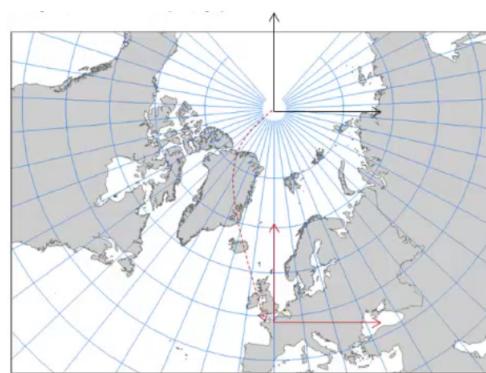
wel belangrijk is, is dat België 4 projectiesystemen gekend heeft.



**Figuur 5.2:** Lambertprojectie voor europa

- Lambert 50
- Lambert 72
- Lambert 2005
- Lambert 2008

Het Belgisch coördinatensysteem is cartesisch. Ze verlegt de oorsprong van de as, die normaal gezien op de top van de projectie ligt, onder België zodat alle coördinaten positief zijn. De verschuiving van de assen 150.000 m naar het westen en 5.400.000 m naar het zuiden worden false easting en false northing genoemd. In 1950 werd de centrale meridiaan op de locatie van de Koninklijke Sterrewacht(Ukkel) gezet. De kaart werd dan wat nauwkeuriger en de centrale meridiaan werd verplaatst naar de nieuwe locatie van de Koninklijke Sterrewacht(Kester). Er ontstond dus een coördinatenverschil met de vorige versie. Om fouten weg te werken werd terug gerekend naar de kaart van '50. Ondanks de nieuwere systemen is dit hetgene dat heden ten dage het meeste gebruikt wordt. In dit onderzoek is gekozen om met Lambert 2008 te gebruiken. Lambertcoördinaten worden uitgedrukt in meter. Lambert 2008 is aangeraden aan België om de datauitwisseling te vergemakkelijken en heeft een verhoogde nauwkeurigheid.



**Figuur 5.3:** Verschuiving van de assen voor Belgische Lambertprojectie

WGS 84 (World Geodetic System 1984) is het coördinatenreferentiesysteem dat gebruikt wordt door de meeste GPS'en. Er is ook het Internationaal Terrestrial Reference System (ITRS). Deze is enkele centimeters correcter, maar minder toegankelijk. De hoogte is bepaald ten opzichte van het wiskundig model van de onderliggende ellipsoïde. Dus dit zegt niet veel over het zeeniveau. Bij Vlaanderen ligt het zeeniveau op 43 m hoogte. Maar omdat er met hoogteverschil gewerkt wordt in de berekeningen zijn deze van de app voldoende. De omzetting gebeurt via een programma dat gemaakt is aan de Universiteit in Mons en Hainaut door Yvan Barbier ?.

Als we de DMS coördinaten omzetten naar Lambert 2008 krijgen we:

- Beginpositie (Lambert) : x= 603784 en y=690205
- Eindpositie (Lambert) : x=603822 en y=693580

De coördinaten werden uit het Excel-bestand gehaald. Ze werden in de online applicatie gestoken zodat de coördinaten konden geconverteerd worden. Door deze te uit de algemene data te halen als csv-bestand waarna deze de online applicatie zal verwerken. Hierna zal deze terug gezet moeten worden tussen de andere data van het bestand waarna de berekeningen met de andere data gebeurt. De fouten van de Lambert-kaart zijn slechts 1cm/km en worden dus buiten beschouwing gelaten bij de rest van de berekeningen.

### 5.3.2 Afstandbepaling via bolafstanden

Een afstand op een cirkel bepalen is eenvoudig gegeven door de formule(5.2), waarbij  $d$  de afstand op de cirkel is,  $r$  de straal van de cirkel en  $\Delta\sigma$  de radialen.

$$d = r * \Delta\sigma \quad (5.2)$$

Elke rechte afstand op de aardbol kan dus afgebeeld worden als een afstand op de omtrek van een cirkel. Deze cirkel wordt verkregen door de rechte door te trekken op de bol zodat het verlengde van beide einden elkaar raken. 2 parameters moeten dus bepaald worden. Er wordt begonnen met de straal van de cirkel.

#### Aardstraal

Op het eerste zicht lijkt dit een eenvoudige parameter, ware het niet dat de aarde geen perfecte bol zou zijn. Ze is een ellipsoïde waarvan de parameters gegeven zijn in vergelijking(5.3).  $a$  staat hierin voor de straal bij de evenaar en  $b$  voor de straal door de polen.

$$\begin{cases} a = 6.378,1370 \text{ km} \\ b = 6.356,7523 \text{ km} \end{cases} \quad (5.3)$$

De afstand van het middelpunt van een ellips naar een punt op de omtrek kan gegeven worden via de formule(5.4). Hierbij is  $R$  (de latitude) de afstand tot het centrum van de ellips en  $\varphi$  de hoek vertrekkende van de evenaar naar de noordpool.

$$R(\varphi) = \sqrt{\frac{(a^2 * \cos\varphi)^2 + (b^2 * \sin\varphi)^2}{(a * \cos\varphi)^2 + (b * \sin\varphi)^2}} \quad (5.4)$$

De volgende stap is de afstand in radialen bepalen van het doorlopen stuk op de ellips.

### *Radialen*

Er zijn 3 verschillende methodes om dit te doen. Dit gebeurt door enkel uit te gaan van  $\phi_1, \lambda_1$  en  $\phi_2, \lambda_2$  wat respectievelijk de 2 coördinaten zijn, gegeven in latitude en longitude.  $f$  is de primaire afplatting van een ellips en kan bekomen worden met formule(5.5).

$$f = \frac{a - b}{a} \quad (5.5)$$

De 3 methodes om de afstand in radialen, of afgelegde stuk op de mantel van de ellips, te bepalen worden in detail in de Bijlage beschreven. In dit stuk tekst wordt er kort overgegaan sinds de principes hier belangrijker zijn, de input eenduidig is en er geen paramters zijn die varieren. Dit zijn de 3 methodes :

- de cosinusregel op boloppervlaktes (5.6)
- de Haversine-formule (5.7)
- de Vincenty-formule (5.8)

### **De cosinusregel**

$$\Delta\sigma = \arccos(\sin\phi_1\sin\phi_2 + \cos\phi_1\cos\phi_2\cos\Delta\lambda) \quad (5.6)$$

Het nadeel van de cosinusregel op boloppervlaktes, is dat ze niet precies genoeg is voor kleine afstanden. Bij het berekenen van korte afstanden in Scilab is de hoek van de cosinus zo klein dat ze steeds afgerond wordt naar 0.

### **De Haversine-formule**

$$\Delta\sigma = 2\arcsin\sqrt{\sin^2\frac{\Delta\phi}{2} + \cos\phi_1\cos\phi_2\sin^2\frac{\Delta\lambda}{2}} \quad (5.7)$$

Het gebruiken van de Haversine-formule is goed voor kleine afstanden, maar maakt nog steeds geen gebruik van een sferoïde. Deze formule gebruikt nog steeds een ideale bol. Ze is wel een stuk eenvoudiger dan de Vincenty formule.

### **De Vincenty-formule**

$$s = \sigma(bA + \Delta\sigma) \quad (5.8)$$

De Vincenty-formule gaat af op afstanden berekenen op een ellipsoïde. In deze formule (5.8) staat  $\Delta\sigma$  niet voor de afstand maar wel  $s$ . De vergelijking die hier getoond wordt is slecht een klein deel in een stelsel vergelijkingen. De keuze van symbolen zijn ongewijzigd gebleven. Zo zal via een reeks vergelijkingen de ellipsvorm van de aarde in rekening gebracht worden. Er zijn 2 wijzen om de Vincenty-formule te gebruiken. De directe en indirecte wijze. De directe neemt een beginplaats en een hoek waaronder bewogen wordt en berekent het eindpunt. Wat voor de berekeningen hier nuttig is, is de inverse formule. Ze neemt begin en eindpunt en zal het afgelegde traject berkenen. Eventueel ook de hoek waaronder dit is gedaan, maar dit heeft hier geen belang. Er wordt meer in detail gegaan in Bijlage H.

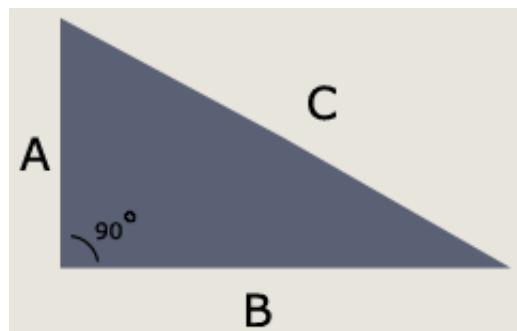
### 5.3.3 Vergelijking berekeningswijzen

Er werd in dit deel op zoek gegaan naar het antwoord op de volgende 2 vragen:

1. Welke methodes gebruiken de apps zelf om de afgelegde afstand te bepalen?
2. Hoe nauwkeurig is dit wanneer een gekende afstand wordt afgelegd?

#### Berekeningswijze van de apps

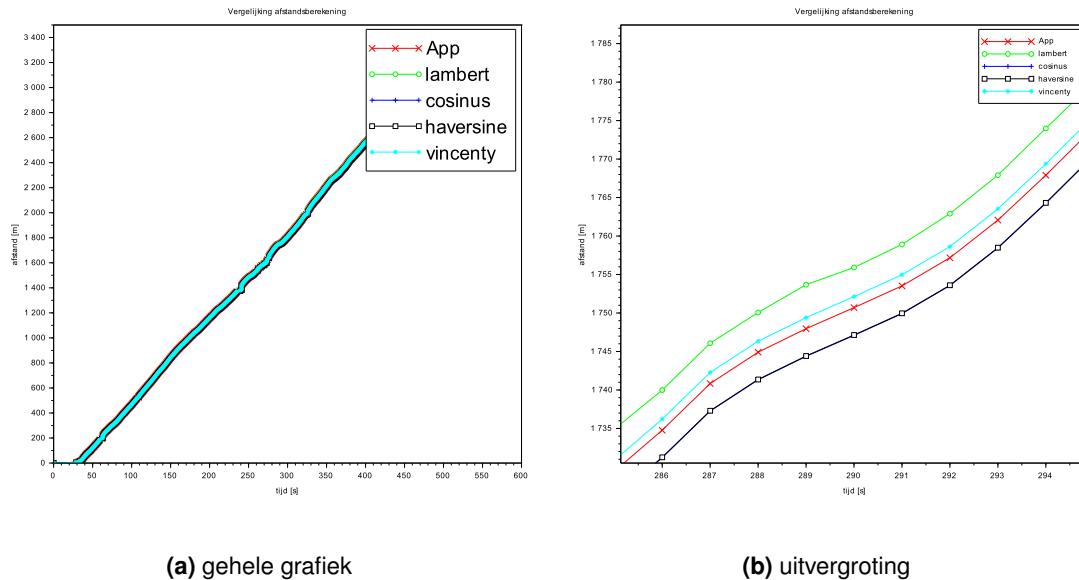
Om te weten te komen op welke manier de afstand berekend wordt in de apps worden de verschillende berekeningswijzen bekeken. In de app *Map My Ride* worden in de ruwe data zowel de coördinaten als de berekende afstand gegeven. De afstand die in de app berekend wordt, kan worden vergeleken met de verschillende berekeningswijzen die werden besproken hierboven om te weten te komen welke methode sommige apps gebruiken. Bij deze berekening werd de hoogte op basis van Pythagoras Figuur 5.9. verrekend in de afstand. Hierbij is  $B$  de afstand tussen 2 coördinaten en  $A$  het hoogteverschil daartussen. De schuine afgelegde weg  $C$  wordt dan berekend via de stelling van Pythagoras (5.9)



Figuur 5.4: De vergelijking tussen berekeningswijzen

$$C^2 = A^2 + B^2 \implies C = \sqrt{A^2 + B^2} \quad (5.9)$$

In Figuur ???. zien we een grafiek waarop eenzelfde afstand op 5 verschillende manieren berekend is. Het gaat over de waarden die door de app zijn berekend, via Lambert, cosinusregel, Haversine-formule en Vincenty-formule. Lambert werd aan dit lijstje toegevoegd omdat de eenvoud van het  $(x,y)$ -systeem uitgedrukt in meter een gemak zou kunnen betekenen. Figuur 5.5b geeft een vergroot beeld van de grafiek voorgesteld in Figuur 5.5a. Het is duidelijk dat alle berekeningswijzen dicht bij elkaar liggen.



**Figuur 5.5:** 4 Berekeningswijzen afstand

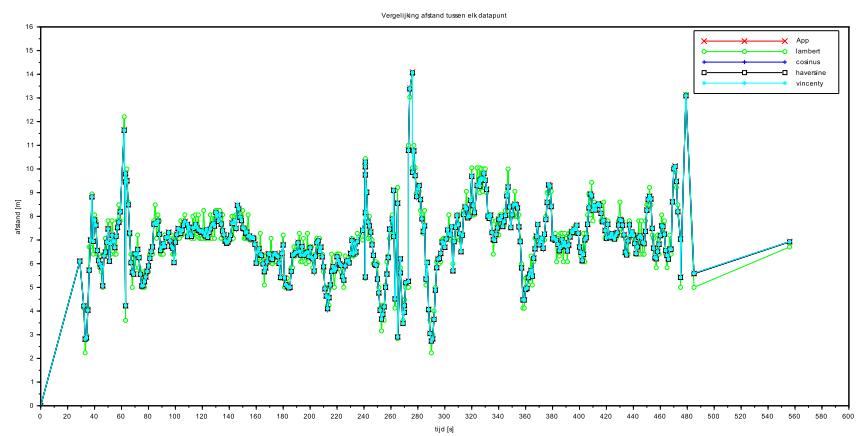
Bij de onderste curve liggen de berekeningswijzen via de cosinusregel op een bol en via de Haversine-formule zo dicht bij elkaar dat ze niet te onderscheiden zijn. Bij deze weergave is enkel de cumulatieve fout in verloop van de tijd zichtbaar. Dit maakt dat het verschil tussen 2 punten klein lijkt in deze grafiek.

In Figuur 5.6 worden de afstanden tussen 2 datapunten weergegeven in functie van de tijd. Hier valt te zien dat niet alle waarden een constant parallel verloop hebben met de waarden van de app. De Lambert-curve uitschieten die hier voor minder opvieren. De uitschieters zijn zowel positief als negatief. Bij verdere inspectie kan men ook zien dat de resolutie van Lambert lager ligt dan de andere waarden. De waarden werden afgerond in het programma die de conversie uitvoerde. In Figuur 5.5 kan bij nauwkeurige inspectie zien dat de waarden op de y-as (afstand tussen 2 datapunten) een veelvoud is van de minimale nauwkeurigheid.

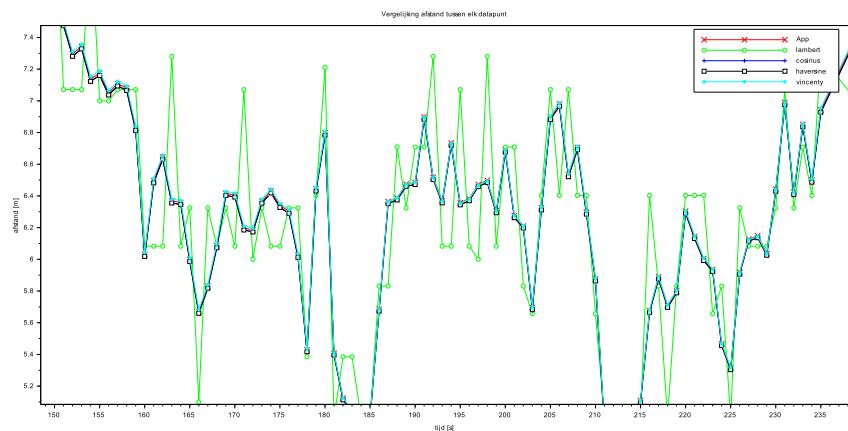
Rit nr.	$d_{gemiddeld}$	$\sigma_{Lambert}$	$\sigma_{cos}$	$\sigma_{haversine}$	$\sigma_{Vincenty}$	datapunten
1	16.094662	0.0273657	0.0032783	0.0032739	0.0010858	190
2	16.442411	0.0256709	0.0032725	0.0032689	0.0010192	190
3	6.9711284	0.0591181	0.0024288	0.0023625	0.0023215	451
4	7.0192527	0.0540068	0.0021016	0.0020742	0.0022324	447

**Tabel 5.6:** Standaardafwijking berekningswijzen

De waarden bij elke afstand zijn onderling vergeleken en daar is de standaardafwijking (Bijlage I)  $\sigma$  van berekend in Tabel 5.6. Dit is telkens gedaan tov de waarde die door de app is berekend. In Tabel 5.6 zijn de waarden van 4 gelijkeritten qua parcours & afstand gemaakt met de *Map My Ride* app om de eventuele verschillen te kunnen waarnemen. In de tweede kolom staat de gemiddelde afstand tussen 2 datapunten volgens de app. Het valt direct op dat deze bij rit 3 en 4 kleiner is dan bij de eerste tweeritten. Dit is niet gecorreleerd met het aantal datapunten die genomen



(a) gehele grafiek



(b) uitvergrooting grafiek

**Figuur 5.6:** Sequentiële afstands berekening tussen 2 punten

zijn. Deze zijn zichtbaar in de laatste kolom van Tabel 5.6. Deze metingen waren gedaan over eenzelfde afstand met 2 verschillende gsms en dezelfde app *Map my Ride*. Het aantal meetpunten ligt aanzienlijk kleiner bij 2 van de 4 metingen. Hier wordt bij de paragraaf 6.3 *Aantal datapunten* verder op in gegaan.

Er kon besloten worden dat :

- Als de data op de Lambert kaart geprojecteerd wordt, 2% afwijking mogelijk is.
- Er is geen noemenswaardig verschil bij de berekening via de cosinus-regel op een sfeer en de Haversine-formule.
- De formule van Vincenty het beste resultaat geeft om berekeningen uit te voeren over afstand en zal voor de rest van de testen zo gebruikt worden.

Hier werd met de hoogte gewerkt. Op het al dan niet toevoegen van hoogte in de berekeningen zal verder besproken worden in paragraaf 6.4 *Hoogteverschil bij apps*.

### Gekende afstand afgaan met apps

Er is voor dit experiment gegaan naar een openlucht atletiekpiste. 100 m in een rechte lijn zijn afgewandeld. Het bleek zo dat de waarden van de apps te ver zaten van de 100 m om iets zinnig te kunnen zeggen over de beste berekeningswijze. De conclusie blijft dezelfde als in de vorige paragraaf.

## **Hoofdstuk 6**

# **Testen met apps en meetfiets**

Hoe betrouwbaar is de snelheid die gemeten word door apps en kunnen we deze verbeteren? In dit hoofdstuk werden testen gedaan met de apps en een meetfiets tegelijk om een referentiewaarde te hebben. Op verschillende wijzen werd bekeken hoe de nauwkeurigheid van dit 'meettoestel' in kaart kon worden gebracht. Er werd naar de precisie van de meetfiets en apps gekeken alsook het verschil in positie tussen apps bij aanvang van eenzelfde rit. De reden achter het variërende aantal metingen per seconde werd besproken en de vreemde resultaten van de hoogtemetingen werden verklaard. Het herkennen van het stoppen van de bestuurder werd besproken en de gemiddelde en maximale snelheid. Met al deze parameters werden op verschillende apps en verschillende gsms testen uitgevoerd en in een tabell samengevat. Op basis van deze informatie werden 2 apps gekozen. De reden waarom er 2 werden gekozen ipv 1 wordt in de paragraaf hieronder kort verklaard.

In het begin van dit onderzoek werden enkele apps uitgekozen om deze testen mee te doen. Naar het einde toe van het onderzoek bleek dat er geen rekening gehouden was met de Windows Phone. De apps die zowel Android als iOS werkten bleken niet te werken voor Windows Phone.

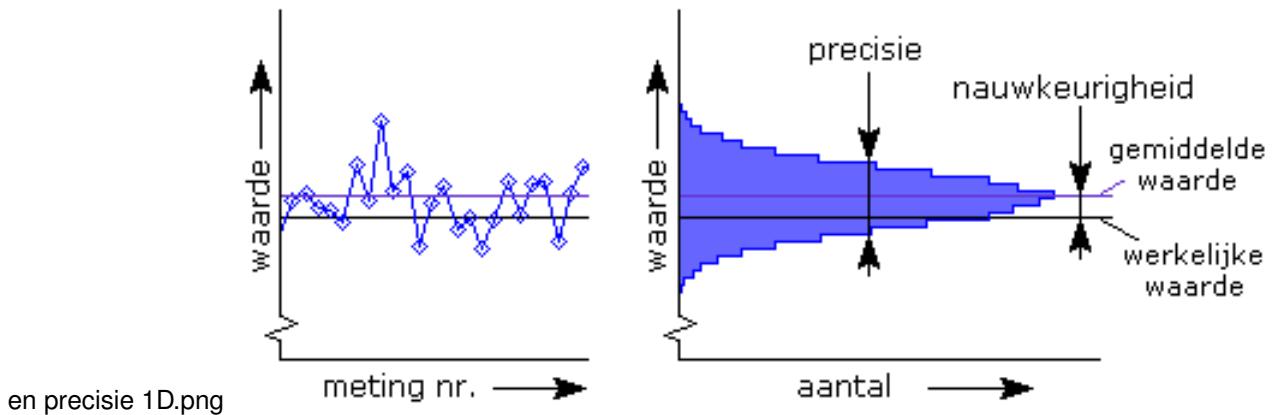
Er is een schaars aanbod van GPS apps die op Windows Phones werken en die aan de eisen voldoen die bij aanvang besproken werden. Daardoor werd er maar met 2 apps getest. Deze werden opgenomen bij het algemene resultaat van dit hoofdstuk maar komen niet voor bij de onderdelen waarbij de testen besproken werden.

### **6.1 Resolutie en precisie van de afstand bij de meetfiets en de apps**

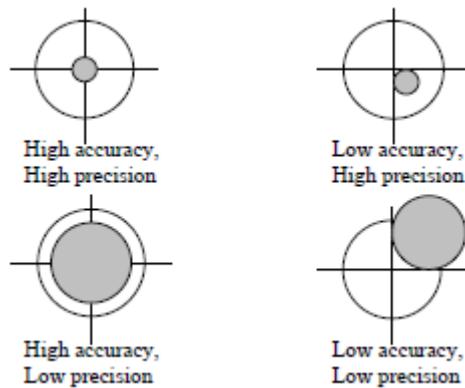
Precisie, juistheid, nauwkeurigheid, accuraatheid, resolutie. Dit zijn termen waarvan de betekenis dicht bij elkaar ligt. Maar het is belangrijk om een onderscheid te maken. In dit stuk werd een onderscheid gemaakt tussen de termen en werd gekeken naar de resolutie van de meetfiets, de accuraatheid van de GPS op basis van literatuur en de resolutie van de ruwe data.

De termen worden kort geduidt om verdere verwarring te voorkomen.

- *Accuraatheid* : de combinatie van precisie en juistheid
- *Juistheid of Nauwkeurigheid* : de afwijking tussen het gemiddelde van de metingen en de werkelijke waarde



Figuur 6.1: accuraatheid vs precisie



Figuur 6.2: accuraatheid vs precisie

- *Precisie* : de spreiding van de meetwaarden rond het gemeten gemiddelde
- *Resolutie* : de kleinste mogelijk nog te onderscheiden meetwaarde

Voor een meting met 1 variabele is dit duidelijk weergegeven in Figuur 6.1.

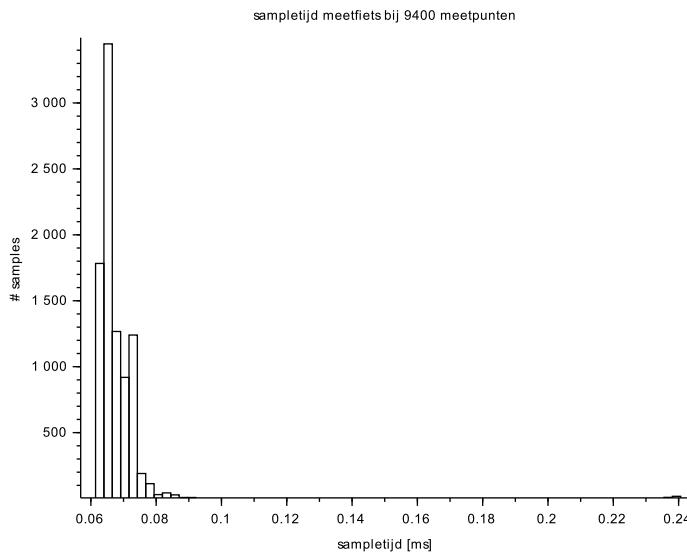
Om een beeld te hebben hoe dit er voor de positiebepaling uitziet is dit goed weergegeven in Figuur 6.2.

### 6.1.1 Resolutie van de meetfiets

We zullen eerst naar de nauwkeurigheid van de meetfiets kijken. De elektronica had een sample tijd van 0,01 s bij het eerste meettoestel. De diameter van het wiel is 26 inch of 660,4 mm. Via de formule voor de omtrek van de cirkel (zie vgl. 6.1), met  $r$  de straal van de cirkel, krijgen we uit de berekening vgl 6.2 een omtrek voor het fietswiel van ongeveer 2m. Dit gaf dus een fout in de afstand die afhankelijk was van de snelheid van de fiets. Precisie =  $0,01s \times m/s$ . Bij een snelheid van bvb 60 km/u<sup>1</sup> geeft dit dus een foutmarge van 0,16 m.

$$\text{omtrek} = 2 * \pi * r \quad (6.1)$$

<sup>1</sup>dit wordt later als maximale snelheid voor de fietser bestempeld



**Figuur 6.3:** Histogram met de sample tijd van meetfiets bij 9400 datapunten

$$2 * 3,14 * \frac{0,660,4 \text{ mm}}{2} = 2,075; \text{m} \quad (6.2)$$

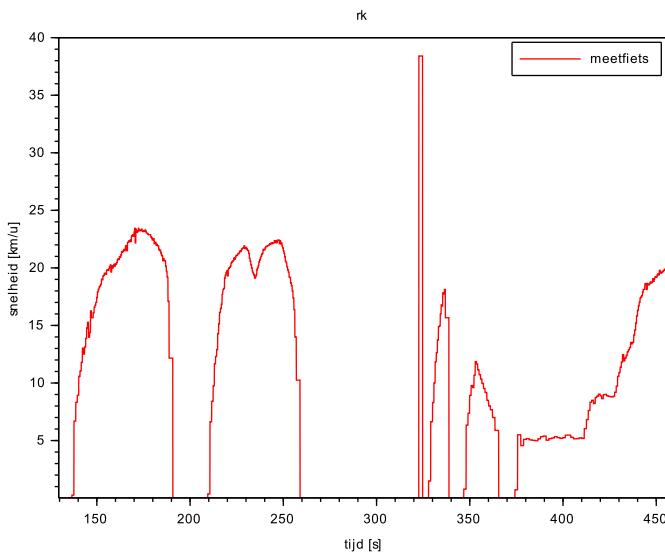
Figuur 6.3 geeft voor een rit met ongeveer 9400 metingen de sample tijd weer bij het 2e meettoestel. Er zit een spreiding op die niet normaal verdeeld is. Het meettoestel heeft een sample tijd van  $68 \pm 8$  ms.

Dit zou volgens dezelfde redenering als hierboven is de mogelijke fout afhankelijk van de snelheid van de fietser. Als er vb wederom 60 km/u genomen wordt  $11 \pm 1,3$  m. Dit is gelukkig niet het geval. De sample tijd geeft weer wanneer de magneet langst de sensor kwam. Daardoor lijkt verkeerdelyk dat de sample tijd lager ligt dan bij het eerste meettoestel.

Het gevaar met de magneetsensorzit bij de fiets in stilstand. Wanneer stilgestaan wordt aan bijvoorbeeld een kruispunt kan het zijn dat de magneet voor de sensor zweeft. Bij kleine bewegingen tijdens stilstand kan het lijken dat er grote versnellingen aan te pas komen. Of dat de fiets niet volledig tot stilstand komt. Dit is zichtbaar in Figuur 6.4 Hier valt helaas weinig aan te doen. De metingen waar dit gebeurde waren beperkt en zijn niet opgenomen in de berekeningen.

### 6.1.2 Precisie van apps in literatuur

Om data te vergelijken bij GPS data moet gebruik gemaakt worden van *one way ANOVA of one way analysis of variance*. Deze methode in statistische analyse waar slechts 1 input factor gewijzigd wordt. ? Hier is dit helaas niet mogelijk gezien er geen kennis is over de stand van de satellieten en signaalsterkte ten alle tijden.



**Figuur 6.4:** Piek in snelheid bij de meetfiets tijdens stilstand

De DOP waarden van  $\geq 3$  worden gezien als goede waarden. Waarden tussen 3 en 6 worden beschouwd als aanvaardbaar en waarden  $\geq 6$  als slechte waarden. Deze waarden zijn standaarddeviaties uitgedrukt in meter. ? Een lage DOP-waarde impliceert niet automatisch een lage fout in positie. Deze hangt af van DOP, signaalsterktes, multipath etc.

Bij testen aan verschillende snelheden? met een Garmin eTrex Vernture GPS bleek de GPS slechts een gemiddelde fout te hebben van minder dan 7% op de snelheid. Wanneer er scherpe bochten gemaakt werden schoot de fout wel de lucht in tot 80,16%. Testen onder constante snelheden hadden een lage fout gemiddeld 1km/u. Bij lage snelheden  $\leq 15$  km/u werd de fout groter en ging naar 1,72km/u.

Een ander onderzoek ? in 2007 ging dan weer in op verschillende types GPS'en en vergeleek ze met open lucht en een pad in het bos. Met 2 verschillende GPS merken kwam hij aan een beschikbaarheid van satellieten van 99,1-99,9% voor DGPS bij open hemel en 96,6-99,2% onder/tussen het bladerdek van bomen. De HDOP(= Horizontal DOP) ging van 0,9-1,5 bij open hemel naar 0,9-2,31 onder de dekking van bomen.

Bij dan weer een ander onderzoek werd de accuraatheid van de GPS data van gsms onderzocht ?, en gevonden dat de data een horizontale error tussen 5 en 8m had bij stilstand. Een factor 2 tot 3 maal slechter dan GPS toestellen. Buiten ging de fout nooit over 30m en indoor nooit over 100m.

Deze paper? stelt dat met een gsm de gemiddelde snelheden aardig geschat kunnen worden, maar de ogenblikkelijke snelheid op gsm een foutmarge kent van  $\pm 5 \text{ mph}$  (of 8 km/u) met een precisie van 84 %.

Over het testen van nauwkeurigheid van snelheden bij de GPS'en op gsms in voorlopig nog niet geschreven of gevonden.

### 6.1.3 Resolutie van apps uit ruwe data

Nu zal even kort besproken worden tot welke grootorde de positie wordt weergegeven door de apps.

Alle apps geven hun positie weer op basis van latitude en longitude in 2 kolommen. Deze getallen gaan allemaal tot 1 micro graad (ofwel 0,0036''). Als we naar de Vincenty-formule kijken en we geven een 2 identieke longitudes in en een latitude die de kleinst mogelijk verschillende waarde heeft krijgen we de kleinst mogelijke afstand  $\text{MIN}(d_1, 2)$  die de app kan weergeven.

- $latitude_1 = 51,063185 \quad longitude_1 = 3,716138$
- $latitude_2 = 51,063186 \quad longitude_2 = 3,716138$
- $latitude_1 = 51^{\circ}3'47.466'' \quad longitude_1 = 3^{\circ}42'58.0968''$
- $latitude_2 = 51^{\circ}3'47.4696'' \quad longitude_2 = 3^{\circ}42'58.0968''$

Na toepassing van de Vincenty-formule :

- $d_{1,2} = 0.111m$

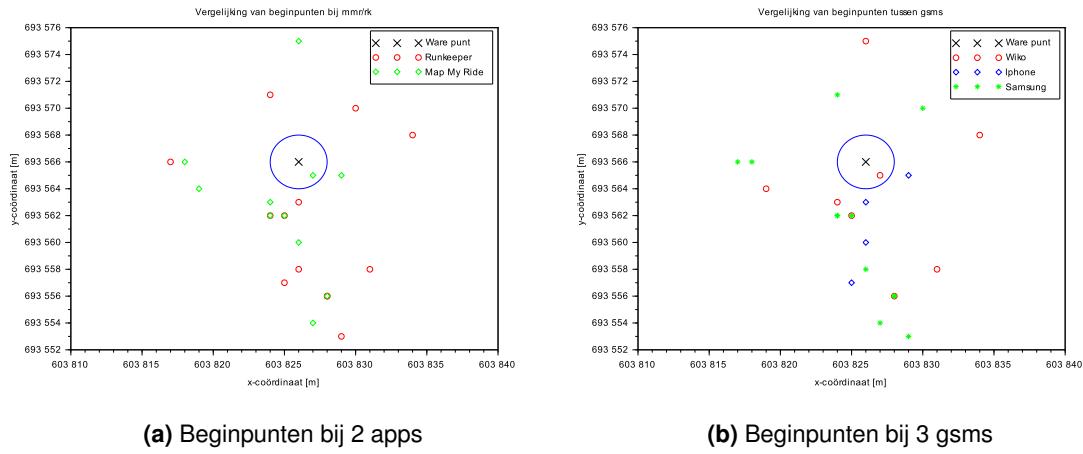
Afhankelijk van de sample tijd kan hier een snelheid mee gelinked worden. Maar een uitspraak doen over de snelheid kan hiermee niet.

## 6.2 Verschil in beginpositie

Om apps onderling te kunnen vergelijken in positie moeten we een punt hebben dat herkenbaar is in de data. Omdat de sample tijd niet uniform is bij de apps is niet mogelijk om bij elk meetpunt de afstand onderling te vergelijken. Waar dit wel mogelijk is, is bij het startpunt. Elke test is van eenzelfde plaats begonnen. Dus wordt de beginpositie vergeleken bij de apps.

### *Waar is de test begonnen?*

De testen zijn niet gebeurd op een een positie waarvan de coördinaten zijn gekend. Omdat alle GPS-data onderling verschilt is het dus moeilijk een uniform punt te kunnen duiden zonder invloedsfactoren die impact hebben op het resultaat van de GPS. Hier kan Google maps tot de redding komen. Google maps maakt gebruik van het WGS84 GIS-systeem. Omdat de visuele representatie een projectie is moet er gekeken worden welke fout er is. Er zijn reeds verschillende onderzoeken gedaan naar de precisie van Google earth's projectie. Omdat Google zijn algoritmes niet wil vrijgeven wordt er enkel op basis van een vergelijkende studie de fouten bepaald. Een studie van ?? heeft bij 436 controlepunten over de wereld een accuraatheid van 39.7 m bepaald. De fouten gingen van 0.4 m tot 171.6 m. Er werd bepaald dat bij Europa, de VS, Canada, Australië, Nieuw Zeeland en Japan de precisie tot 14m naukeriger is dan het gemiddelde en de rest van de wereld. Er zijn 2 studies gevoerd om de precisie te bepalen van de horizontale accuraatheid op kleinere gebieden. In Texas, Amerika ?? waar ze een horizontale nauwkeurigheid van 2.64 m halen. In Rome, Italië ?? werd er zelfs 1 m horizontale nauwkeurigheid gehaald. Helaas is er geen onderzoek gedaan naar deze in België. Dus om deze beginpunten te bepalen moeten we

**Figuur 6.5:** Beginpunten

Gemiddelde afwijking tot het beginpunt [m]	
Runkeeper	6.84
Map My Ride	5.32

**Tabel 6.1:** Gemiddelde afwijking van beginpunten van apps

een zekere marge nemen die een schatting zal zijn. Er zal 2 m genomen worden. Het gaat hierbij vooral om het onderling verschil tussen de punten die de apps aanduiden in beeld te brengen. Om een overzicht te hebben van de initiële punten is het werken met Lambert-coördinaten voldoende en eenvoudig. Gezien Lambert-coördinaten in meter worden uitgedrukt is zo ook de grafiek eenvoudig leesbaar. Rond het beginpunt, bepaald via *Google maps*, is een cirkel met straal 2 m getrokken. Dit geeft zorgt voor een eenvoudige inschatting van de fout door de coördinaten gevonden via Google maps.

- Latitude 51,050462 ; Longitude = 3,710294 ofwel
- x= 603826 y= 693566

We zien bij Figuur 6.5a een puntdiagram, dat de beginpunten buiten de marge vallen van de verwachte fout. Deze bestaat uit 2 m fout via de coördinaten via Google maps en 3 m fout van de satelliet. Omdat er niet voldoende datapunten zijn, valt er geen sluitende conclusie te vormen tussen het verschil in afstand van de apps. Ook bij het verschil tussen de gsms bij Figuur 6.5b

Gemiddelde afwijking tot het beginpunt [m]	
Wiko	5.86
iPhone	4.85
Samsung	6.86

**Tabel 6.2:** verschil tussen gsms

valt dit niet te doen. Wel is de verwachting dat de iPhone het beste scoort ingelost. Deze zou gemiddeld een betere ontvanger hebben dan andere gsms.

Op het eerste zicht lijkt deze denkpiste (het meten van snelheidsprofielen met apps) tot mislukken gedoemd. Maar dit mag nog niet geconcludeerd worden wegens 3 redenen.

- Het eerste datapunt kan een eerste snelle bepaling zijn van de GPS/gsm vooraleer scherper in te stellen.
- Er kan op deze plaats slecht bereik zijn die weggewerkt kan worden door datapunten op de rest van de weg.
- Het kan zijn dat de precisie ligt in het verschil tussen 2 punten bepalen en niet de preciese coördinaten.

Uit metingen is gebleken dat het eerste meetpunt telkens een grove schatting is.

*Besluit :* Het eerste punt bij de ritten zijn een eerste localisatie van de GPS bij stilstand die onnauwkeuriger is dan wanneer de gebruiker in beweging is. Het is dus zinvol om deze te verwijderen gezien de grotere onnauwkeurigheid tov de andere punten.

### 6.3 Aantal datapunten

Zoals reeds gezien is in Tabel 5.6 kan bij eenzelfde rit door verschillende apps meerdere datapunten genomen worden. Dit is afhankelijk van de signaalsterkte, de app en van de gsm. Naar zowel het effect van de app als gsm werd gekeken. Er werd gebruik gemaakt van de iPhone en Wiko. Om een vergelijking te kunnen maken met een GPS was tijdens de metingen ook een Garmin fiets GPS mee.

#### 6.3.1 Effect gsm op de sample tijd

Hier werd gekeken naar het effect van de gsm. Daarvoor dienden de twee andere variabelen gelijk gehouden te worden. Deze metingen werden daardoor gelijktijdig uitgevoerd met dezelfde app *Map My Ride*. Er werden 2 metingen uitgevoerd met een totale duur van ongeveer 1000 seconden. De resultaten van deze testen zijn te zien in Tabel 6.3 :

Gemiddelde sample tijd [seconden/meting]		
	Map my ride	Runkeeper
Garmin	1,08	1,01
Wiko	2,53	2,03
iPhone	1,12	2,03

**Tabel 6.3:** Aantal samples voor 3 apps en 2 gsms

De Garmin GPS had een bijna constante sample tijd van 1 s. De iPhone scoorde veel beter dan de Wiko-gsm. Deze heeft maar de helft zoveel meetpunten genomen als de iPhone. Het voor de hand liggende besluit zou zijn te zeggen dat de iPhone een veel betere transmitter heeft dan de

	mmr	mbc	ct	rk
hoogteverschil begin-eindpunt[m]	25,78-25,79	0	37,6-54,1	20,6-21,5
hoogteverschl	2,77-3,48	7,43	4,15-5,11	1,70-2,68

**Tabel 6.4:** resultaten apps

Wiko, en daardoor meer metingen kan uitvoeren. Maar het kan ook liggen aan een wisselwerking tussen het besturingssysteem van de gsm enerzijds en de app anderzijds. Dit valt pas uit te sluiten wanneer eenzelfde test is gedaan met een andere app.

### 6.3.2 Effect app op de sample tijd

Eenzelfde experiment werd gedaan maar dan met de app *Runkeeper*. De resultaten hiervan zijn bijgevoegd aan Tabel 6.3. De duur van de 2 ritten was wederom een geheel van ongeveer 1000 seconden.

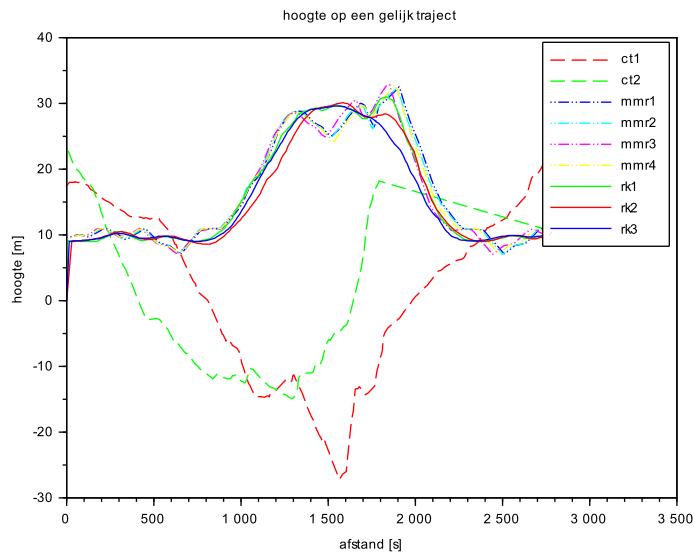
Op beide ritten gaf de Garmin GPS wederom een consistente sample tijd aan van ongeveer 1 s. Het opmerkelijkste was bij deze meting het verschuiven van de sample tijd op de iPhone. Deze was verdubbeld ten opzichte van de vorige app. De sample tijd was gelijk met deze van de Wiko-gsm. Bij de vorige meting kon dus niet besloten worden dat de ‘betere’ gsm automatisch meer metingen kon uitvoeren door een betere transmitter/ontvanger. De wisselwerking tussen het besturingssysteem van de gsm en lag dus aan de basis van de sample tijd.

*Conclusie :* De neiging zou zijn om voor de app te gaan waarbij de sample tijd gemiddeld lager ligt. Maar het uiteindelijke doel was een programma te ontwikkelen voor gebruikers waarvan de gsm niet is gekend. Daarbij is het beter om een uniforme sample tijd te hebben voor bewerkingen zoals later zal blijken bij de stoptijden en de smoothers.

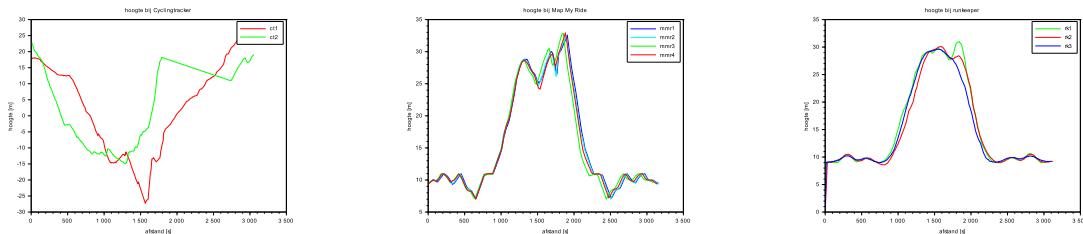
## 6.4 Hoogteverschil bij apps

Wederom kan de veronderstelling gemaakt worden dat bij begin- en eindpunt de waarden gelijk geacht mogen worden. Er wordt in Tabel 6.4 gekeken naar het hoogteverschil bij begin en eindpunt. Deze werd verondersteld 0 te zijn. Ook wordt er gekeken naar de minimale en maximale hoogte tijdens te ritten.

In Figuur 6.6 is te zien hoe de hoogtes variëren bij al deze metingen. Om een duidelijker beeld te geven van het verschil bij de apps onderling wordt in Figuur 6.7 de apps afzonderlijk weergegeven.



Figuur 6.6: hoogte van alle apps



Figuur 6.7: hoogtes apps afzonderlijk resp. ct, mmr, rk

Het is duidelijk dat de hoogtes gemeten bij Cyclingtracker 6.7(links) het zwaarst verschilt in de app zelf. Ook is de hoogte negatief weergegeven. Dit zou logisch zijn bij punten onder de zeespiegel, of indien de hoogte in absolute waarden zou kloppen. Ook de beginhoogte van 43m bij zeespiegel op WGS 84 bij de *afstandsbeperking via Lambert2008* zou deze daling niet kunnen verklaren. Bij gebrek aan een verklaring zal deze app naar de achtergrond verschoven worden door de reeds slecht bekomen resultaten. Bij beide andere apps 6.7(midden) en 6.7(rechts) is het resultaat consistent onderling. Het vreemde hier is dat bij Map My Ride er 3 pieken zich voordoen tijdens de rit, waar er bij Runkeeper maar sprake is van 1.

Er werd gekeken naar hoe apps de hoogte berekeningen deden, omdat ze niet voldeden aan de verwachtingen. Wanneer ze via trilateratie zouden worden bepaald zou de hoogte veel meer pieken moeten vertonen, dan het gekende parcour, door de onnauwkeurigheid van de GPS. O.a. de site van Runkeeper gaf hier verduidelijking over. De datapunten met positie in worden naar een derde partij gestuurd (Topocoding). Zij linken de data met een hoogte die Runkeeper aan de tabel toekent. Strava ging zelfs nog een stap verder. Zij gaven aan dat deze optie pas gebruikt werd wanneer er geen barometrische altitudemeter ter beschikking was. Dit meet op basis van de luchtdruk de hoogte. Dit is bij geen van de apps het geval. Het verschil in pieken kan dus verklaard

worden dat Strava en Runkeeper een andere bron hebben om hun hoogte mee te bepalen.

Bij metingen op de horizontale atletiekpiste werd oa bij Runkeeper een hoogteverschil van 0 - 2,3 m vastgesteld verschillend van meting tot meting. Het blijkt ondanks de by-pass van de onnauwkeurige GPS hoogte bepaling geen betrouwbaar systeem te zijn. Dit is ook duidelijk doordat Strava dit ziet als tweede keuze bij het bepalen van hoogtes.

## 6.5 Verschill in gemiddelde en maximale snelheid

Deze metingen hadden als uiteindelijk doel een zicht te hebben op de snelheid van een fietser. Wanneer een app veel uitschieters heeft geeft dit een vertekend beeld. Dus werd gekeken of de uitschieters ver boven deze van de meetfiets lagen. Indien dit het geval was zou het mogelijk zijn om bij punten die boven een snelheidsgrens lagen te schappen en af te doen als een foute meting. Dit met versnellingen doen is niet aangeraden gezien ook veel metingen springen rond de 'echte' waarde. Deze konden ook onrealistische versnellingen geven maar is daarom geen te verwaarlozen meting. Uit de metingen bleek dat er geen grens te leggen viel bij snelheden die verder van de rest afwijken. Slechts enkele punten met een ratio kleiner dan 1/1000 waren er punten die enorme uitschieters waren. Daarom kon er een snelheidsgrens genomen worden die boven de realistische snelheid van de fietser ging. Bij de speed pedelec lag deze iets hoger dan bij de andere elektrische fietsen. De grens werd arbitrair op 55 km/u gelegd voor speed pedelecs en 45 km/u voor gewone fiets en de pedelecs.

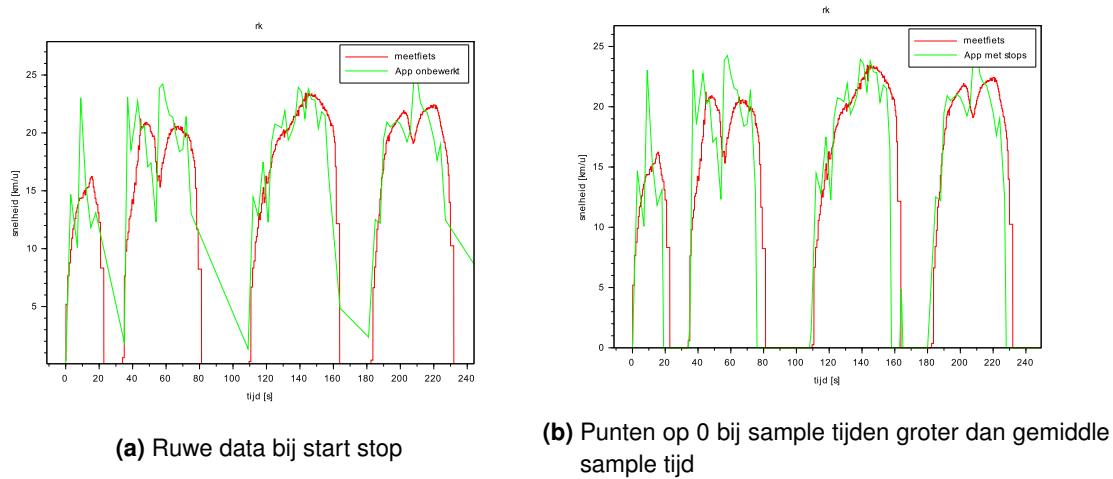
## 6.6 Stops

Er werd reeds iets vreemd gemerkt toen de beginpositie bekeken werd. Die eerste waarde bleek heel onzeker te zijn, maar enkel de eerste waarde werd vernoemd om te schrappen. Dit was niet uit de lucht gegrepen. De apps stoppen vaak met meten wanneer ze geen beweging detecteren. Sommige apps zoals *Sports Tracker*<sup>2</sup> gaven de optie om stilstand al dan niet in de metingen op te nemen. Zo werd de optie tussen constant meten, bij stilstand langer dan 2 seconden of 4 seconden gegeven. Dit bleek in de ruwe data niet te kloppen bij stilstand of lage snelheden stopte deze net als de andere apps met continu te meten. De gemiddelde sample tijd gaat hierdoor de lucht in<sup>3</sup>. Er werd dus gekeken wanneer de apps stoppen met meten.

Deze meting werd bij helder weer gedaan en een route waar weinig hindernissen een rechtstreeks signaal verhinderde. Het werd ook op een recht traject uitgevoerd. Bochten hebben dus geen impact bij het meten van hogere snelheden op de meetfiets dan op bij de app. Dit principe is zichtbaar in BijlageJ.

<sup>2</sup>dit is de enige app uit de geselecteerde apps die deze optie had

<sup>3</sup> Bij de vorige metingen of volgende metingen waar dit voor een vertekend resultaat zou kunnen geven werden deritten gebruikt zonder stilstand.



Figuur 6.8: Start stop rit met Runkeeper

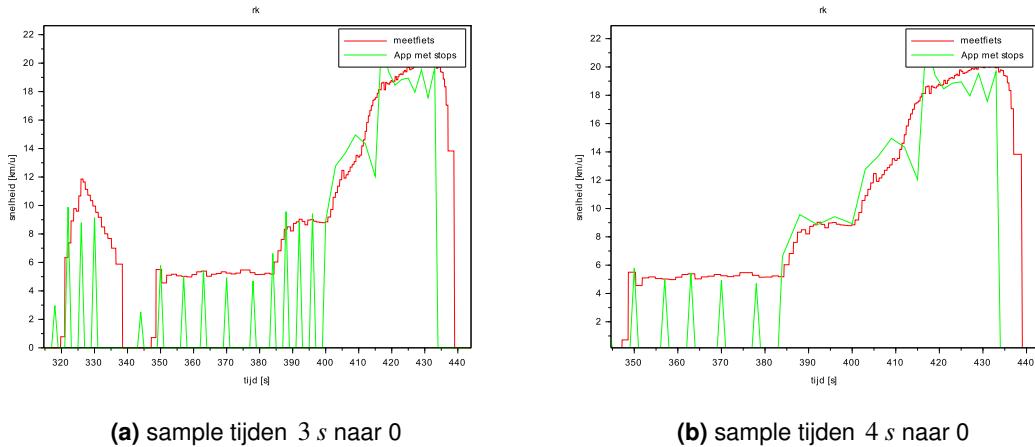
Bij Figuur 6.8a is een deel van een rit te zien waarbij meerdere malen gestart en gestopt is geweest. De rode curve, de meest effen curve, is deze van de meetfiets waarmee de ruwe data van de app (hier Runkeeper) gebruikt werd. Zoals zichtbaar is werden de stops niet op eenzelfde wijze geregistreerd bij de apps als bij de meetfiets. De schuine rechten bij de apps bij stilstand kan misleidend zijn. Deze rechten zijn geen punten die gegeven waren door de ruwe data, maar was een verbinding door Scilab van 2 punten waartussen geen data was. Het leek eenvoudig te zeggen dat tussen deze punten de teller eenvoudigweg op nul gezet moest worden. Bij een sample tijd die groter was dan het gemiddelde konden we de teller op nul zetten. Het resultaat is zichtbaar in Figuur ???. Ondanks het mooie resultaat stelden zich nog enkele problemen. Er werd naar verschillende snelheden gekeken om te weten wanneer de app besloot te stoppen met te meten. De sample tijd in de app lag telkens op gehele seconden. De gemiddelde sample tijd lag bij 2 seconden. In Figuur 6.9a is gekozen voor een stoptijd van 3 s. Het valt duidelijk te zien dat bij beweging tot 9 km/u de sample tijden groter zijn dan 3 seconden en op 0 terecht komen. Om te weten te komen of dit ook bij 4s sample tijd het geval was werd dit ook bekeken. In Figuur 6.9b is dit weergegeven. Het viel direct op dat de tijd nu bleef steken op 5 tot 6 km/u. Men zou kunnen verwachten dat bij 5 s ook deze snelheden zouden herkend worden. Dit bleek niet het geval. De sample tijd bij deze snelheden bleek rond 7 s te liggen. Dit was teveel tijd om korte stilstanden nog te herkennen.

Het is giswerk om te achterhalen wat de reden is achter deze tijden. Een mogelijke verklaring is dat wanneer de afstand tussen 2 punten bij 2 metingen niet ver genoeg uit elkaar liggen de app dit ziet als stilstand met een fout in de nieuwe meting.<sup>3</sup> HDOP geeft de korste afstand om buiten de straal van onzekerheid te komen rondom 1 positie. Als we kijken naar de gemiddelde sample tijd van 2s en de onzekerheid op de positiebepaling door de GPS HDOP 3. Kunnen we zien in de berekening van vlg (6.3)

$$\frac{3 \text{ m}}{2 \text{ s}} = 1,5 \text{ m/s} = 5,4 \text{ km/u} \quad (6.3)$$

Dit zou kunnen verklaren waarom er zo lang gedaan wordt over een nieuwe meting. Deze redenering kan ook doorgetrokken worden naar de sample tijd bij 3s zoals in berekening(6.4).

$$\frac{3 \text{ m}}{1 \text{ s}} = 3 \text{ m/s} = 10,8 \text{ km/u} \quad (6.4)$$



**Figuur 6.9:** Experimenten met stoptijden

*Conclusie :* Bij een sample tijden die 4 of meer seconden duren werden de snelheden op nul gezet. De snelheden onder 6km/u worden als onbetrouwbaar aanschouwd omdat niet met zekerheid kan gezegd worden of het al dan niet om stilstand ging.

## 6.7 Schommelingen snelheid

Wanneer de curve van de app met deze van de meetfiets vergeleken werd was duidelijk dat deze curve grilliger was. De oorzaak lag voor de hand : de accuraatheid van de app. Dwz zowel de nauwkeurigheid als precisie hebben hier invloed op. Er is geen eenheid om de grilligheid te duiden. De afgelegde afstand was geen juiste indicatie, want wanneer de snelheid schommelt zal op sommige punten de afstand kleiner, en andere groter zijn dan deze van de meetfiets. Wanneer het pad van de snelheidscurve in lengte vergeleken werd was er wel een indicatie over hoeveel de curve schommelt. Deze werd berekend met vgl (6.5).

$$\xi = \frac{\sum_{i=2}^n v_{i,app} - v_{i-1,app}}{\sum_{k=2}^n v_{k,meetfiets} - v_{k-1,meetfiets}} \quad (6.5)$$

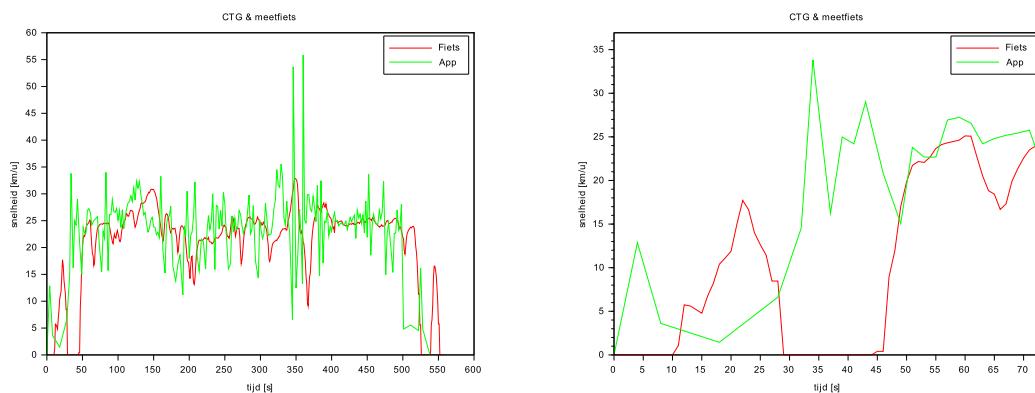
Hierbij is i het aantal metingen bij de app, en k dat bij de meetfiets. Er werd begonnen vanaf het 2e punt gezien telkens 2 punten dienen vergeleken te worden. De verhouding tussen app en meetfiets werd weergegeven door  $\xi$  en heeft geen eenheid. Het dient nogmaal herhaald te worden dat deze formule niet uit de literatuur komt, maar juist is ontworpen en gebruikt om een beeld te krijgen van de grilligheid met een getal dat tussen apps te vergelijken valt.

## 6.8 Gelijk leggen curves

### 6.8.1 Het gelijk leggen van het beginpunt van een rit

Het tijdstip waarop een meting begon bij de meetfiets en app waren natuurlijk niet gelijk. Maar om ze toch te kunnen vergelijken moest er dus een tijdstip zijn die voor beide curves herkenbaar was. Het eerste meetpunt was niet bruikbaar omdat deze al weggefilterd was. Voor deze metingen is bij stilstand de app en meettoestel op de fiets aangestoken en enkele seconden gewacht op een warm start. Er werd dan ongeveer 10 m gefiets en terug stilgestaan voor een 5-tal seconden. Dit is zichtbaar in Figuur 6.10 Pas dan begon de meting van de rit. Bij deze snelheid was er een grote stijging die herkenbaar was voor app en meetfiets. Deze werden handmatig op elkaar geplaatst. Dws de rijen in de matrices die voor dit tijdstip kwamen werden verwijderd. De tijdstippen waarop beide metingen begonnen werden op 0 gezet en ook het einde van de ritten werd gelijktijdig afgesneden.

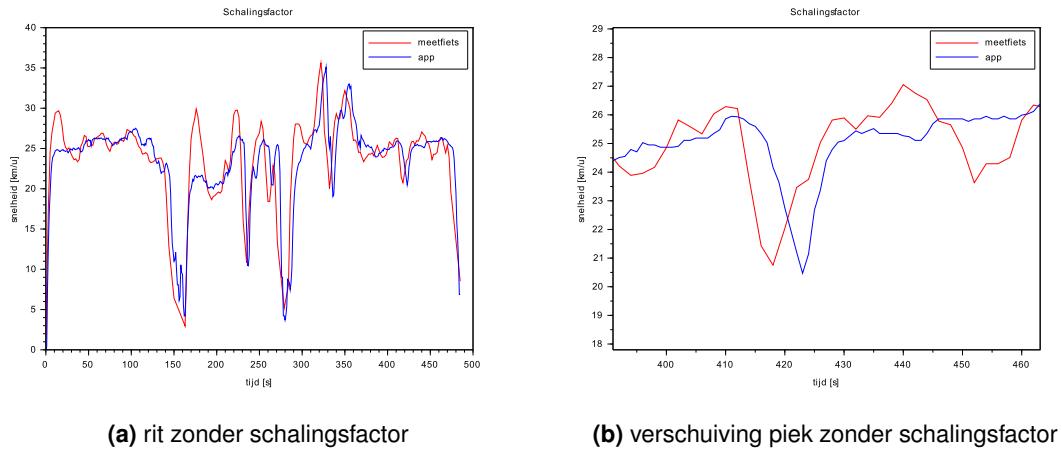
Een gelijk tijdstip vinden was niet zo vanzelfsprekend. Gezien er verschillende sample tijden waren voor de meetfiets kon zelden een rij gevonden worden met een gelijk tijdstip waarop app en meetfiets een meting deden. Om gelijke tijdstippen te hebben moest er geïnterpolleerd worden. Dit werd gedaan met een functie die in Scilab aanwezig is onder de naam *interp1*. Daar kon gekozen worden tussen linear, spline en nearest. Er is uitgegaan van linear omdat deze relatie tussen 2 meetpunten bij de snelheid ook gebruikt werd. Alle meetpunten werden geplaatst op 1 seconde van elkaar. Dwz dat de sample tijd kunstmatig is aangepast. En de andere waarden g Door het *linspace* commando kon de beginwaarde 0 en het laatste tijdstip waarop gelogd werd op te delen in stappen van 1 seconde. Dit zorgde dat er ook in de toekomst de snelheden op bepaalde tijdstippen konder vergeleken worden<sup>4</sup>. Het resultaat is zichtbaar in Figuur 6.11.



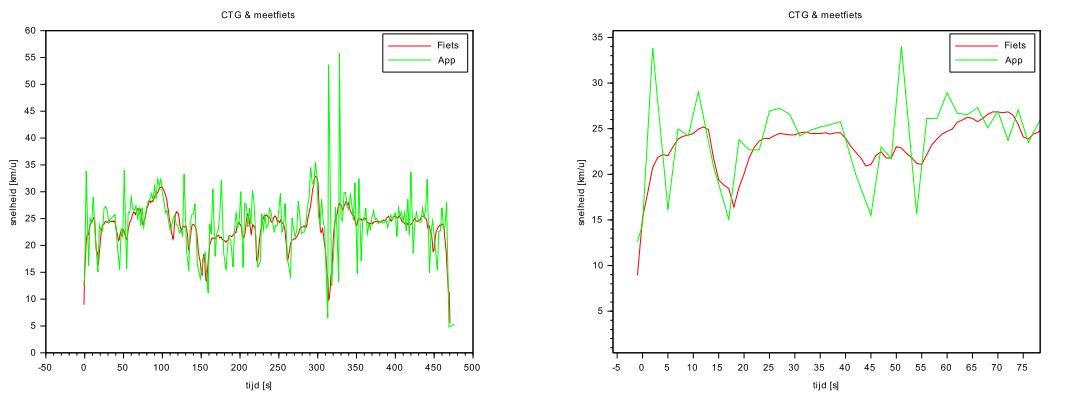
(a) totale rit zonder bijsnijden aanvang en einde      (b) rit zonder bijsnijden aanvang en stop uitvergroot

**Figuur 6.10:** Rit zonder bijsnijden aanvang en einde

<sup>4</sup> Dit werd nog niet gedaan bij de keuze tussen de apps omdat er nog teveel verschillende parameters werkzaam waren. Bij 1 app met 1 gsm op 1 rit was de vergelijking tussen alle punten een stuk doenbaarder en minder tijdrovend



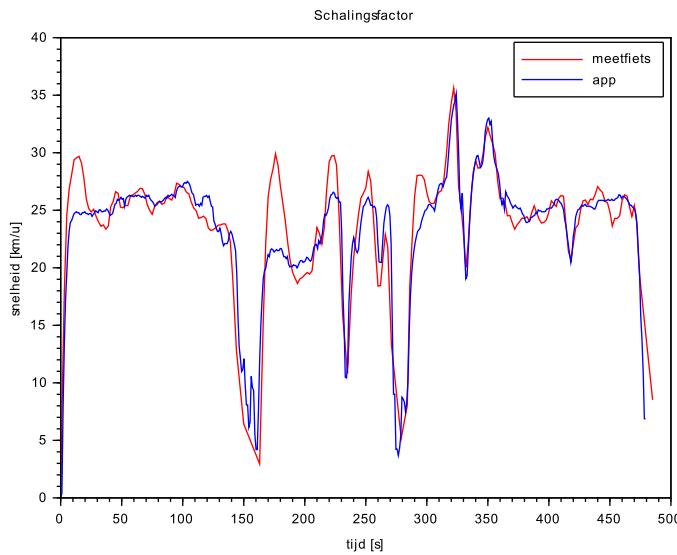
Figuur 6.12: Rit zonder schalingsfactor



Figuur 6.11: Rit met bijsnijden aanvang en einde

### 6.8.2 Schalingsfactor in de tijd

Maar dit gaf nog steeds een vreemd beeld (Figuur 6.12a). De pieken bij aanvang lager dichter bij elkaar dan op het einde van de rit. Figuur 6.12 vergroot een deel uit van deze rit op een van de laatste pieken. Hoe kwam dit? De oorzaak lag bij een afwijking in de tijd bij een van de meettoestellen. Door op het einde van enkele ritten te kijken naar het verschil in pieken konden via de duidelijk afgeronde getallen (door interpolatie) de factor bepaald worden waarmee die pieken uit elkaar lagen. In Figuur 6.12 liggen de 2 punten op tijdstip  $t_{piek,meetfiets} = 336$  en  $t_{piek,app} = 332$ . Dit komt neer op een factor van ongeveer 1,2 %. Deze factor is een fout die bij de interne klok van de gsm ligt of het gevolg van het afronden van de tijdstippen naar gehele seconden in de data. Wanneer de factor bij de tijd van de app in rekening werd gebracht werd een bevredigender resultaat verkregen (zie Figuur 6.13).



Figuur 6.13: Rit met schalingsfactor

## 6.9 Resultaten apps

Van elke app staat er een afbeelding van een willekeurige rit in Bijlage K. Zo blijven de getallen minder abstract. De testen die hierboven werden besproken werden op de ritten van de apps gebruikt en samengevat in Tabel 6.5. Elke app heeft meer dan 1000 datapunten gemeten. Al deze testen werden met de Samsung gsm uitgevoerd. Er is telkens hetzelfde traject Bijlage C afgelegd bij gelijke weersomstandigheden. Het is belangrijk bij deze tabel te weten dat ctg en st enkel door de Windows Phone konden worden afgespeeld en de Windows Phone enkel deze apps kon afspelen. Er werd eerst gedacht om meerdere apps op 1 gsm af te spelen. Bij de Winsows Phone is het niet gelukt 2 apps op hetzelfde moment te kunnen draaien omdat de apps dat niet toelieten. Bij de testen bij de Samsung zijn de ritten verworpen omdat ze halverwege een defect hadden of wegvielen. Ze zijn bijgevoegd in Tabel 6.5 zodat ze ook met de andere apps konder vergeleken worden.

	mmr	mbc	ct	rk	ctg	st
sample tijd [s]	2,136635	1	2,8	2,45	2,46	1,08
procentueel afstandsverschil [%]	1,83	13,38	61,38	1,2	2,12	3,12
$v_{gem,app}$ [km/u]	27,55	17,32	139,05	25,06	23,19	23,83
$v_{gem,meet fiets}$ [km/u]	25,62	24,76	20,5	24,93	22,78	23,1
$v_{max,app}$ [km/u]	57,69345	32,87	931,86	60,1	47,63	64,16
$v_{max,meet fiets}$ [km/u]	38,84	32,87	33,71	32,83	31,17	32,8
Maximaal hoogteverschil [m]	25,79	65,1	45,85	27,2	36	37
Begin-eind hoogteverschil [m]	0,17	16,7	20,35	0,64	4,4	7
$\xi$	2,9	0,866	27,4	3,83	3,9	8,45

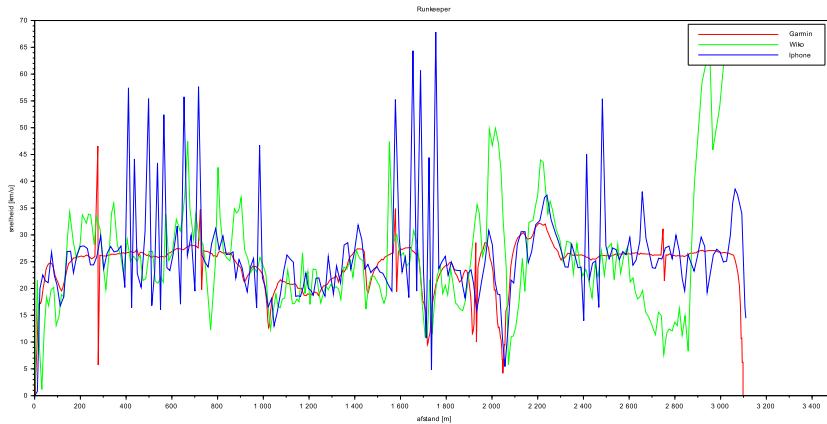
Tabel 6.5: resultaten apps

- *sample tijd* : De sample tijd ging bij geen enkele app boven 3 s. mbc had een constante sample tijd. Dit bleek geen verbetering te zijn voor de andere waarden.
- *Procentuele afwijking in afstand* : Deze leek op 2 apps na bijna verwarloosbaar klein te zijn (onder 3,5%) . Meer dan 10% is dan weer te groot om mee verder te werken.
- $v_{gem}$  : De gemiddelde snelheden van de apps gaven bij mmr, rk, ctg en st een waarde met een verschil naar de meetfiets toe die kleiner was dan 1,5 km/u. Dit zijn goede resultaten wanneer we naar snelheden planden te kijken binnen een bepaalde zone. Het geeft een beeld over de nauwkeurigheid van de metingen.
- $v_{max}$  : Deze waarden geven aan welke hoogtes de pieken bereiken. De meeste waarden liggen tussen de 40-60 km/u. Wanneer dit in combinatie met de  $\xi$ -waarde bekeken werd kon gezien worden of het om veel kleine schommelingen ging of enkele grote.
- *Hoogte verschil*: Bij hetzelfde traject geven de apps waarden die niet overeenstemmen met elkaar. mmr en rk liggen rond 25m in hoogteverschil op hetzelfde traject en ctg en st rond 36m. Op basis hiervan kan geen enkele meest waarschijnlijke waarde bepaald worden. Het verschil in begin-eindhoogte ligt enkel bij mmr en rk in de buurt (<1) van 0. De andere apps geven waarden van 4-46 m.
- $\xi$  : Deze waarden was een maat voor de grilligheid van de curve en gaf een indirect beeld over de precisie (spreiding) van de metingen. 3 van de 6 apps hebben een waarde die binnen 2,5-4 ligt. Dit is een waarde waarbij verwacht kan worden met smoothing bijna gelijke waarden te krijgen.

De invloed van deze resultaten op het kiezen van apps werd besproken bij het onderdeel 6.11.1 Keuze apps.

## 6.10 Metingen met 1 app en meerdere gsms

We zagen dat er een verschil zat tussen verschillende apps. Om de invloed van de gsm te weten te komen werd 1 app op 2 gsms tegelijk gebruikt op eenzelfde parcours. Om een referentie te hebben werd er gebruik gemaakt van een fietsGPS van het merk Garmin deze was niet voorzien van een magneetsensor. Wat hier werd gewenst bekeken te worden was het verschil een gsm kan hebben op de ruwe data. Het resultaat van deze test is te zien in Figuur 6.14. De rode cuve is de meetfiets, de blauwe de iPhone en de groene de wiko gsm.



**Figuur 6.14:** Meerdere gsms die gebruik maken van Runkeeper

Er was duidelijk te zien dat gsms andere waarden opvingen, ondanks dat ze ongeveer een gelijk multipath error hadden. De sterkte van de ontvanger mag dus niet onderschat worden. In Tabel 7.1 worden de gegevens van Figuur 6.14 geven. Het heeft hier geen zin om meerdere metingen onderling te vergelijken. Ze zijn niet te vergelijken met andere ritten gezien de omstandigheden verschillend zijn telkens opnieuw. De ephemeris, multi-path ed. wijzigen. Deze data is opnieuw gedaan met ongeveer 1000 datapunten.

	Garmin	iPhone	wiko
sample tijd [s]	1	2,54	2,13
$V_{gem}$ [km/u]	23,84	26	26,6
$V_{max}$ [km/u]	50,14	67,24	63,22
Maximaal hoogteverschil [m]	0	36,7	20,6
Begin-eind hoogteverschil [m]	0	9,55	1,1
$\xi$ tov Garmin	0	2,45	1,97

**Tabel 6.6:** resultaten Runkeeper op 2 gsms

Het valt op dat de Garmin bij de sample tijd het beste scoort. Dit valt te verklaren doordat deze GPS een beter model is, en de zender sterker is dat deze op de gsms. De gemiddelde snelheden bij beide gsms ligt dicht bij elkaar, maar het hoogteverschil niet. Dit kan zijn door een uitschietier die een verkeerde hoogte gaf, of een fout in het plakken van de hoogte bij een bepaalde coördinaat door de app retroactief. De verwachtingen waren dat het duurdere model (de iPhone) beter zou scoren. Maar er zijn meer pieken bij de iPhone. Er viel dus geen verwachtingspatroon aan het type gsm te koppelen. Bij de kandidaten waarvan de gsm onbekend zijn zal dit dus erkend moeten worden en de foutmarge groter genomen moeten worden.

## 6.11 Besluiten apps

### 6.11.1 Keuze apps

De hoogtes verschilden sterk bij alle apps op 2 na. Dit waren mmr en rk voor zowel het maximale hoogteverschil als het verschil in eind- en beginhoogte. Voor ctg en st was dit enkel het maximaal hoogteverschil dat onderling gelijk lag. Dit was een indicatie dat de hoogtes ondanks de extra hulpmiddelen die werden ingeroepen door de apps zelf te verwerpen. De fout door dit te negeren werd erkend, maar achterwege gelaten in de rest van dit onderzoek. Gezien het Vlaamse landschap niet erg heuvelachtig is, zal dit weinig effect hebben op de resultaten. De  $\xi$  waarde geeft direct aan dat 2 apps buiten beschouwing kunnen gelaten worden. ct & st scoorden hier met een factor die minstens 2x zo groot was als de anderen. Maar ook mbc scoort hier slecht op. Deze app overcompenseert en neemt bijna geen wijzigingen in snelheden waar. Voor de Windows Phone is de keuze dan snel naar ctg gegaan. Bij de keuze tussen mmr en rk scoort mmr beter. Maar zoals gezien bij het onderdeel 6.3 aantal datapunten bleek dat dit wijzigde bij de gebruikte gsm of besturingssysteem. Dit en het feit dat de  $\xi$  waarden van ctg en rk dicht bij elkaar liggen maakt het bij de smoothers ook eenvoudiger dezelfde parameters te gebruiken.

Concreet :

- Er werd gekozen voor Runkeeper voor iOS en Android en Cycling Tracker GPS voor Windows Phone
- De stilstand werd bepaald door bij sample tijden die groter waren dan 4 s de snelheid op 0 te zetten.
- De hoogte werd niet in rekening gebracht door de grote onzekerheid en gebrek aan eenzelfde bepaling tussen apps.
- Er kan geen eenduidige precisie en nauwkeurigheid gebruikt geplakt worden op de apps. Wel dat enkele beter scoorden dan andere, maar het effect van de gsm en randfactoren zorgden ervoor dat dit niet rechtstreeks gelindert kon worden met de app.
- De resultaten waren nog niet bruikbaar voor conclusies en finale uitspraken voor de grilligheid deel wegwerk kan worden. De ritten die de apps weergeven werden stemden niet overeen met de ruwe data. Zij bewerkten hun data nog voor ze deze weergaven. Dit werd ook gedaan in het volgende hoofdstuk.

### 6.11.2 Mogelijke verbeteringen

Hier wordt kort de mogelijkheid tot verbetering bij dit proces beschreven.

- *Gyroscoop* : Er is een voorstel door ? om een gyroscoop in gsms te steken die een schatting zou kunnen maken wanneer het aantal satellieten voldoende is voor een goed resultaat. Dit zou ook een stap voorwaarts zijn in het schatten van snelheden.
- *GNSS* : Wanneer de internationale gemeenschap de GNSS samenvoegt voor een verhoogde precisie van minstens 25% ? zullen de resultaten van localisering door gsms verbeteren.

- *DDTW* : Er bestaat een techniek DDTW of Derivative Dynamic Time Warping ?. Gezien de multipath fouten vaak op gelijkaardige plaatsten voorkomen stellen ze in deze paper voor om de fout op eenzelfde route aan te passen. De fouten worden herkend door het verlagen van de signaalsterkte. Dit kan wel enkel gedaan worden bij eenzelfde parcours wat bij dit onderzoek niet het doel is.
- *Bluetooth* : De apps zouden kunnen uitgerust worden met sensoren die via bluetooth de snelheden naar de gsm stuurt. Dit zou het meetfietsprincipe toepassen en de accuraatheid verbeteren. Maar dit onderzoek zoekt een oplossing zonder extra kosten.

# Hoofdstuk 7

## Smoothers

Zoals reeds vermeld zijn smoothers bewerkingen die uitgevoerd worden om uitschieters in data te verwerken zodat de pieken minder uitgesproken zijn en de resultaten accurater. Er zijn verschillende manieren om de ‘werkelijke’ waarde van de piek in te schatten. Er werden hier 3 besproken die in andere literatuur gebruikt werden. De 3 methodes werden onderling vergeleken met een meetfiets om zo te zien welke methode het dichtst kwam bij de realiteit. Dit werd besproken aan de hand van 1 curve die met de 3 methodes bewerkt werd. Er werd gekeken hoe de smoother omgaat met zowel kleine variaties als grote pieken.

- Lokaal gemiddelde smoother
- Gauss-kernel smoother
- Kalman-filter

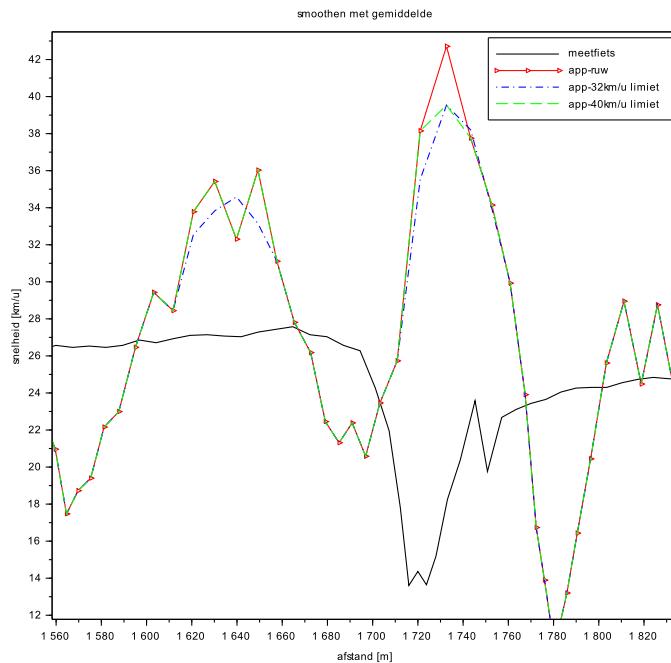
### 7.1 Lokaal gemiddelde smoother

De formule hiervoor wordt gegeven door vergelijking (7.1). Deze methode werd gebruikt door ?. De formule geeft de gesmoothde snelheid  $v_i^\xi$  op tijdstip  $i$  door voor elke waarde  $i$  de 2 aangrenzende waarden erbij te nemen en van deze 3 het gemiddelde te nemen.

$$v_i^\xi = \text{Gemiddelde}(v_{i-1}, v_i, v_{i+1}) \quad (7.1)$$

De eerste vraag die werd gesteld moest worden, was :” wanneer passen we de smoother toe?”. Het grootste probleem in algemene meetresultaten waren de grote uitschieters. Maar wanneer is een punt een uitschiet? Om onrealistische snelheden te achterhalen was het nodig om een grens te bepalen waarbij de snelheid nog realistisch geacht wordt. Bij de paper van ?? werd er gekeken naar welke manier er verplaatst werd. Ze gaan er van uit dat een snelheid groter dan 12 km/u geen wandel, maar een fiets snelheid is, en wanneer de snelheid over 32 km/u gaat een auto is. Bij de testritten waren er momenten waar de snelheid op de fiets boven 32 km/u ging bij stukken die bergaf waren. Om het effect te zien op de curve plotten we beide curves en bekijken het verschil ten opzichte van de meetfiets. De eerste en laatste waarde kunnen volgens deze formule slechts gedeeltelijk uitgemiddeld worden wat al een eerste nadeel is. In Figuur 7.1 zien we het resultaat.

Het viel direct op dat de correcties zelfs niet in de buurt kwamen van meetfiets. Waardoor kwam dit? Omdat de gemiddeldes enkel genomen werden bij de pieken was er aan de algemene ‘foute’



**Figuur 7.1:** Smoother op basis van lokaal gemiddelde

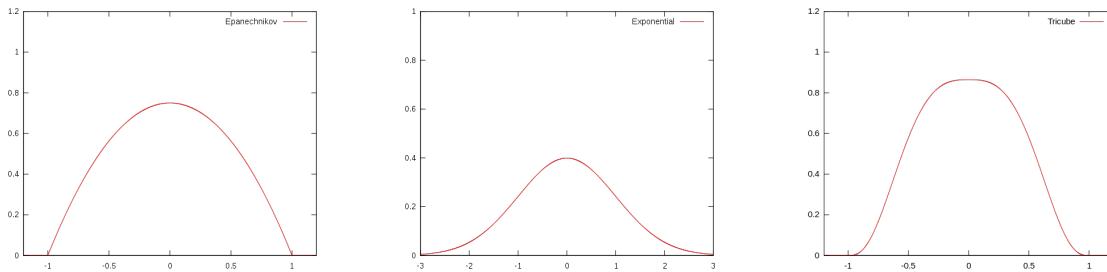
curve van de app niets gewijzigd. Bij de pieken viel op, dat wanneer de curve boven de 40 gecorrigeerd werd ze een gelijke waarde aannamen. De rechterpiek was nauwelijks gewijzigd. Dit komt omdat het datapunt geen geïsoleerde fout is. De omliggende punten zijn ook extreemere waarden. Wanneer dit dus bij pieken gaan die plot 300 km/u aangeven zal de gecorrigeerde versie nog steeds een ireële waarde geven.

Een suggestie om dit probleem te verhelpen zou zijn om meer omliggende datapunten te gebruiken. Maar de grootte van de uitschieter bleef teveel in de schaal leggen waardoor de pieken niet weg werden genomen. Om dit weg te werken zouden teveel punten genomen moeten worden waardoor de ogenblikkelijke waarde niet significant punt was en een gemiddelde waarde bevatte. Dit kon worden verholpen met de Gauss kernel smoother. Deze methode werd verworpen er er werd op zoek gegaan naar een smoother.

## 7.2 Gauss kernel smoother

Deze methode wordt gebruikt bij ?. Er wordt gebruik gemaakt van de curve van Gauss. Deze wordt nog even kort in Bijlage L besproken. Wat deze smoother doet is ipv de snelheid gebruiken om te smoothen, ze werkt met de coördinaten gegeven in de ruwe data. Hierdoor worden fouten niet doorgegeven op de afgeleiden van de afstand naar tijd (snelheid en versnelling). Ook wordt er gewerkt met smoothing over de hele curve en niet enkel bij de pieken.

De formule voor de Gauss kernel smoother is weergegeven in formule (7.2)



Figuur 7.2: Epanechnikov, Gaussische & Tricube kernel

$$\bar{c}(t) = \frac{\sum_j (w(t_j) * c(t_j))}{\sum_j w(t_j)} \quad (7.2a)$$

$$w(t_j) = e^{-\frac{(t-t_j)^2}{2\sigma^2}} \quad (7.2b)$$

Voor elke  $c(t)$  is voor elke coördinaat  $\in x, y, z$  de gesmoothde waarde op tijdstip  $t$ .  $c(t_j)$  is de ruwe waarde van coördinaat  $c$  op tijdstip  $t_j$  en  $w(t_j)$  de Gaussische kernel functie die berekend wordt voor elk punt in de tijd  $t_j$  door vgl (??). Dit wordt ook wel de Nadaraya-Watson kernel-gewogen gemiddelde genoemd. In deze kernel herkennen we de Gaussische curve. De bandbreedte wordt gesteld op 10 seconden wat neerkomt op een gebied van 15 seconden gesmooth wordt. Deze waarde wordt gebruikt door ? omdat ze gezien wordt als een redelijk tijdsframe voor realistische wijzigingen in tegenstelling tot de uitschietende waarden.

### *kernels*

Een kernel is een functie die moet voldoen aan :

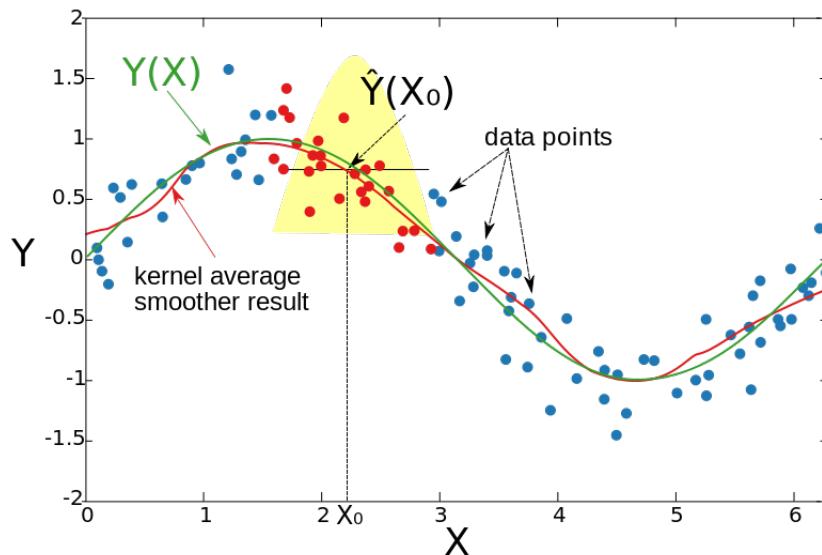
- $\int_{-\infty}^{\infty} K(u)du = 1$
- $K(-u) = K(u)$

Dit houdt in dat de oppervlakte onder de curve gelijk moet zijn aan 1 en dat hij symmetrisch is rond het nulpunt. Er zijn meerdere curves die als kernel gebruikt kunnen worden om te smoothen. De 3 die het meest gebruikt worden zijn te zien in Figuur 7.2.

- Epanechnikov kernel (links)
- Gaussische kernel (midden)
- Tricube kernel (rechts)

De exponentiële curve is voor het schatten bij datapunten geen slechte optie gezien de punten die het dichtst bij zijn het zwaarste doorwegen. Wanneer het lijkt dat de pieken in de grafiek uit meerdere opeenvolgende punten zou bestaan zou een overschakeling naar epanechnikov een valabel alternatief zijn. Om een beeld te geven hoe je deze bewerking kan voorstellen wordt Figuur 7.3.

Je ziet hier op de x-as wat bij de berekeningen hier de tijd voorstelt en op de y-as de coördinaat. De Gauss curve schuift op van links naar recht waar hij telkens de waarden neemt tov  $X_0$ . De



Figuur 7.3: visualisatie smoothing met Gaussische kernel

Gauss-curve loopt in principe verder door, maar door de exponentiële curve zijn deze waarden praktisch verwaarloosbaar.

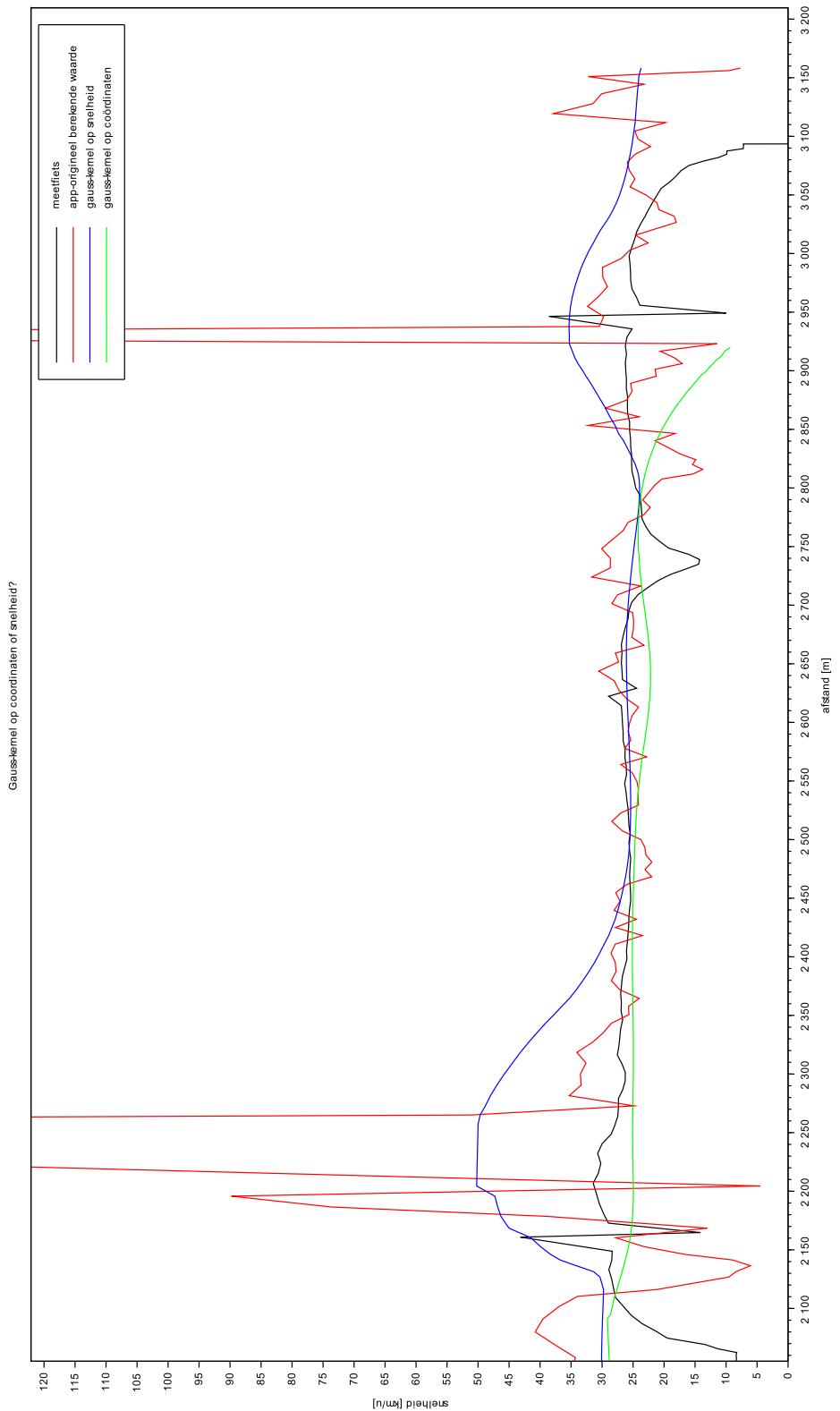
Er wordt direct gekeken welke de verschillen zijn wanneer de smoothing gebeurt op de snelheid en wanneer deze rechtstreeks om de coördinaten gebeurt (zie Figuur 7.4).

Deze Figuur 7.4 geeft slechts het einde van de rit zodat wat besproken wordt duidelijker zichtbaar is. Bij de pieken van de originele data zien we dat de smoother op de snelheid meegaat in deze waarden waar deze op de coördinaten dat niet doet en zo dichter bij de meetfiets komt. Het volgende dat opvalt is de totaal afgelegde weg. Deze is bij de originele waarden en de smoother op de snelheid groter dan de meetfiets en bij de smoother op de coördinaten korter dan de meetfiets. Dit is eenvoudig te verklaren. De pieken in de originele waarden zorgen voor een hogere afstand/snelheid op een korte tijdseenheid waardoor de totale afstand groter wordt dan deze van de meetfiets. De smoother op de snelheid wijzigt enkel de snelheid, maar wijzigt de waarde waar deze uit voortkomt (afstand & tijd) niet. De afgelegde afstand blijft dus gelijk. De smoother op de coördinaten is ongeveer 170m korter. Dit kan komen doordat de spreiding te groot is genomen en hogere snelheden/afstanden gemiddeld worden waar deze werkelijk een grotere snelheid had. Om te zien welke invloed de spreiding heeft op het resultaat bekijken we de smoother op de coördinaten met een spreiding ( $\sigma$ ) die wijzigt.

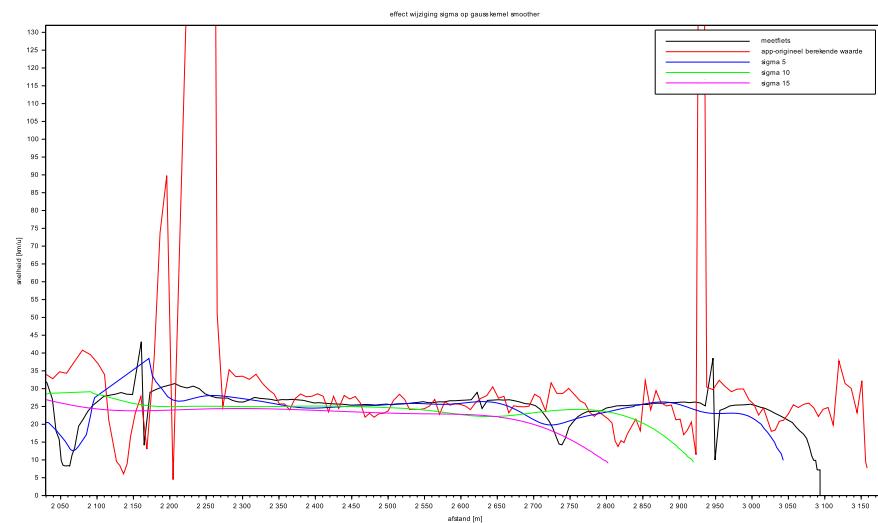
Dezelfde grafiek wordt genomen waarbij de smoothing gebeurt op met een spreiding sigma = 5,10 en 15. Het resultaat is zichtbaar in Figuur 7.5

Deze resultaten vallen binnen de verwachtingen die daarjuist gesteld werden. Bij een kleinere  $\sigma$  gaat de afgelegde afstand naar boven. Het verschil in afstand komt hier neer op ongeveer 1,6% wat binnen een toelaatbare marge ligt in dit onderzoek. In dit onderzoek zal dus verder gegaan worden met  $\sigma = 5$  in tegenstelling tot ?. Andere resultaten die in Tabel 7.1 volgen dit patroon.

Deze resultaten zijn verschillend van app tot app en dus niet uniform. Niet elke app heeft eenzelfde sample tijd zoals reeds vermeld. Het kan zijn dat de app/gsm effect heeft op de sample



**Figuur 7.4:** Gauss-kernel smoother op coördinaten en snelheid



**Figuur 7.5:** wijziging van  $\sigma$  bij de Gauss-kernel smoother

	meetfiets	geen correctie	$\sigma_5$	$\sigma_{10}$	$\sigma_{15}$
afgelegde afstand [m]	3093.53	3158	3042.93	2919.82	2802.39
$v_{gem}$ [km/u]	23.30	26.74	23.27	22.26	21.17

**Tabel 7.1:** resultaten smoothen

tijd waardoor de invloed van  $\sigma$  wijzigt.

### 7.2.1 Bepalen van de spreiding $\sigma_{optimaal}$

Om de spreiding  $\sigma_{optimaal}$  te bepalen moet niet gewerkt worden vanuit het resultaat maar vanuit de theorie. Welke waarde voor  $\sigma$  is een waarde die lateraal goed is voor elke test en type gsm? Volgens het onderzoek door Levenberg-marquardt ? is bij een niet-lineaire vergelijking de minimale fout op het kwadratisch gemiddelde  $\Sigma = 15$ . Wederom een resultaat dat niet strookt met meerdere testen die gevoerd werden bij het onderzoek hier. De gaussische filter vindt zijn toepassing meer in de visuele sector. Door een gebrek aan literatuur over deze filter met deze toepassing werd er gekeken naar een alternatief.

## 7.3 De Kalman-filter

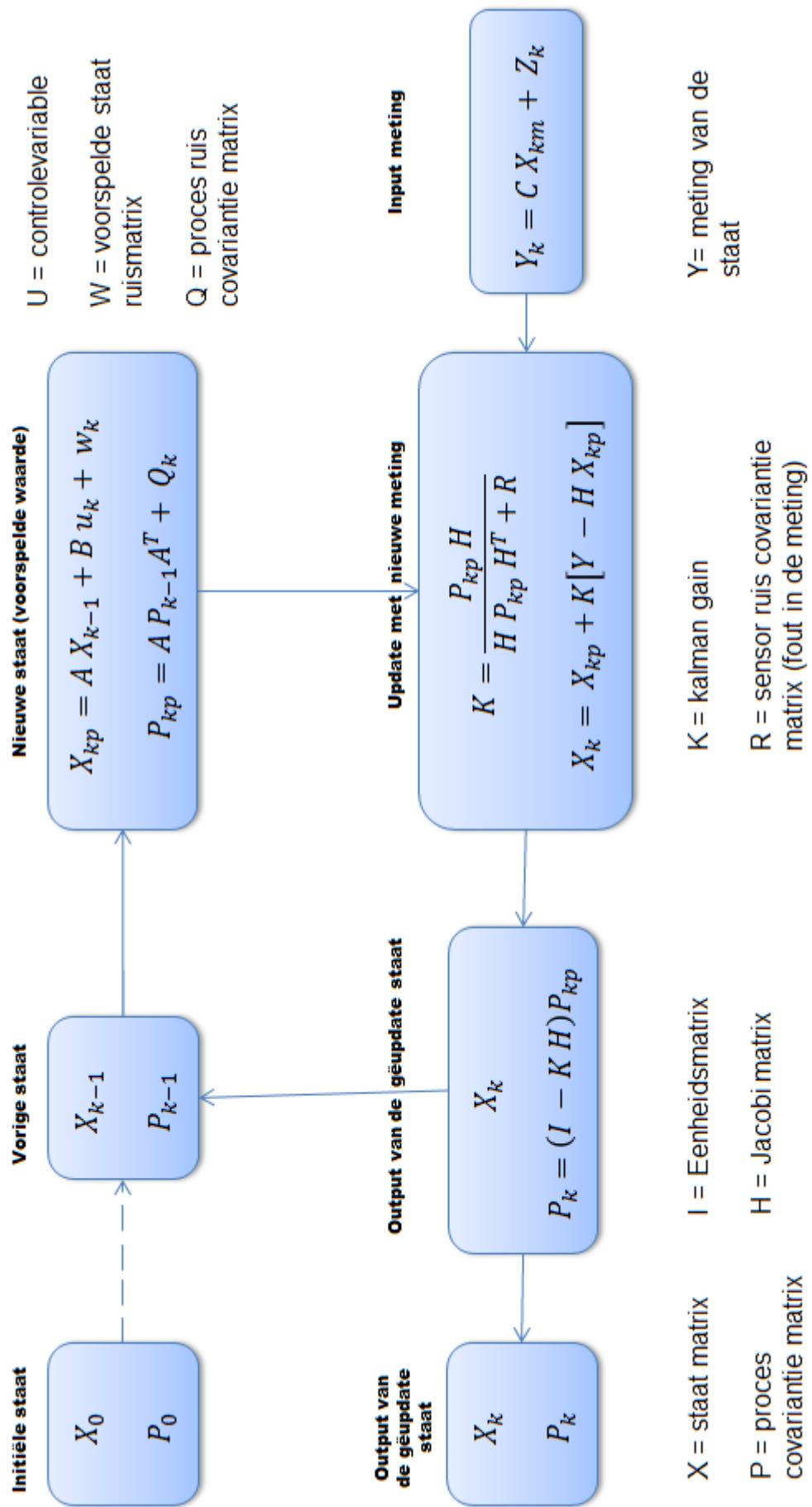
De Kalmanfilter is een filter die op basis van een iteratief proces met behulp van formules en een set metingen een snelle schatting maakt van de waarde, positie, snelheid, etc. van een gemeten object. Tegenwoordig wordt er ook gebruik van gemaakt voor het verwerken van afbeeldingen , robotica, oceanography en biomedisch onderzoek. De metingen bevatten onvoorspelbare fouten en varianties. Ze is ontzettend populair en er zijn vele papers over geschreven. Jammer genoeg verwachten ze een grote wiskundige voorkennis bij de uitleg die ze geven. De uitdaging hieronder is om de filter bekopt en duidelijk te omschrijven zodanig dat we aan het werk kunnen met de benodigde kennis zodat we de filter kunnen gebruiken voor de gemeten data van de GPS.

### 7.3.1 Algemeen overzicht

In het algemeen overzicht wordt het gehele proces kort doorlopen en beschreven a.d.h.v. Figuur 7.6. Dit voorbeeld beslaat zowel de Kalman-filter als de uitgebreide of extended Kalman-filter. Het verschil tussen beiden is het toepassingsgebied. De Kalman-filter maakt gebruik van lineaire functies. De uitgebreide Kalman-filter maakt gebruik van niet-lineaire functies. Wanneer we met wijzigende snelheden werken, werken we ook met versnellingen. We maken dus gebruik van de uitgebreide Kalman-filter. Door de curve op te splitsen in kleine lineare stukjes via wiskundige technieken wijzigen de formules niet veel.

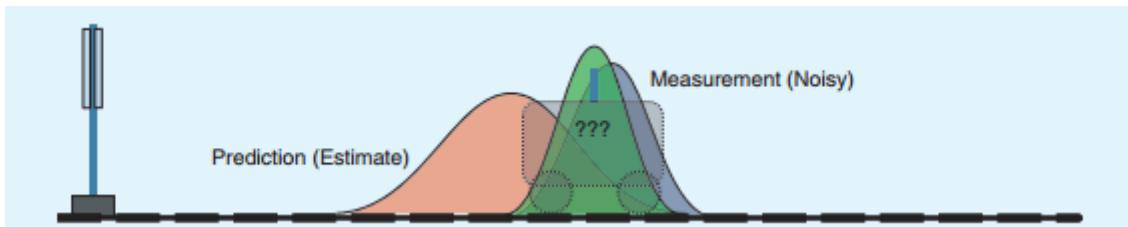
We beginnen met een initiële staat  $X_0$ . In het voorbeeld is dit een positie en een snelheid. Deze waarden zijn een schatting van de grootheid bij aanvang. $P_0$  is de covariantie matrix. Dit is de afwijking op de waarden van  $X_0$ . Om de lus in gang te zetten worden de waarden eenmalig de waarden van de vorige staat. De waarden van de vorige staat worden weergegeven door subscript  $k-1$ . Het subscript  $p$  staat voor het woord previous.

De nieuwe staat wordt voorspeld door middel van de bewegingsvergelijking en de waarden van de vorige staat. Samen met de input van de nieuwe meting gaat de voorspelde waarde met deze van de meting vergeleken worden. Om een gewicht/ mate van invloed aan beiden te geven wordt er gebruik gemaakt van de *Kalman gain*. Deze bepaald welke van beiden meer zal doorwegen en naar welke waarde van beide de uiteindelijke waarde meer zal neigen. Het resultaat hiervan wordt gegeven in de *Output van de gëupdate staat*. Deze waarden zullen gebruikt worden als resultaat en een invloed spelen op de volgende voorspelde staat.



Figuur 7.6: Schema Kalman-filter

Het concept kan ook geschatst worden a.d.h.v. Figuur 7.7. In de rode Gaussische curve zien we de voorspelde positie van het object dat we volgen en de onzekerheid  $w_k$  daarop. De blauwe Gaussische curve is de gemeten waarde met onzekerheid  $Z_k$ . Door de voorspelling en de meting te combineren a.d.h.v. de Kalman Gain gaan we een nieuwe onzekerheid krijgen. Deze is voorgesteld door de groene Gaussische curve en kan in de formules onder de vorm  $P$  gevonden worden. Hieronder zal wat dieper ingegaan worden op elke stap in de cyclus.



Figuur 7.7: Onzekerheden bij de Kalmanfilter

### 7.3.2 De initiële/vorige staat

Hier wordt kort besproken hoe de staatsmatrix en de covariantie matrix in elkaar zitten en hoe ze er uit zien in de initiele of vorige staat.

De *staatsmatrix* bestaat uit een positie, snelheid en versnelling afhankelijk van de doeleinden waarmee de Kalmanfilter gebruikt wordt. Bij wijzigende snelheden maken we gebruik van alle drie. Dit kan in 1 dimensie of meerdere, maar om het voorbeeld eenvoudig te maken gebruiken we 1 dimensie. In vgl (7.3) zien we de staatsmatrix. Hierbij is  $x$  de positie is en  $\dot{x}$  de snelheid. In vgl (7.4) zien we een matrix voor een 2D beweging in het  $(x,y)$ -vlak. Om een versnelling toe te voegen aan de vergelijking kan dit eenvoudig door vgl(7.3) aan te passen tot (7.5). Er zal gewerkt worden met vgl. (7.3). De reden daarvoor zal later verduidelijkt worden.

$$X = \begin{bmatrix} x \\ \dot{x} \end{bmatrix} \quad (7.3)$$

$$X = \begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \end{bmatrix} \quad (7.4)$$

$$X = \begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \end{bmatrix} \quad (7.5)$$

De *covariantie matrix* is een matrix waarin de variantie van een of meerdere dimensies gevormd wordt. De variantie  $= (\sigma)$  van 1 groep metingen wordt gegeven in vgl (7.6) waarbij  $n$  het aantal metingen zijn,  $x$  de waarde van 1 meting en  $\bar{x}$  is het gemiddelde van de metingen.

$$\sigma_x^2 = \frac{\sum_{i=1}^n (\bar{x} - x_i)^2}{n} \quad (7.6)$$

In vergelijking (7.7) wordt dit voor 1 dimensie gegeven. Bij 2 dimensies krijgen we een  $4 \times 4$  matrix zoals in vgl (7.8).

$$\left[ \frac{\sum_{i=1}^{n-1} (\bar{x}-x_i)^2}{n} \right] \quad (7.7)$$

$$\begin{bmatrix} \frac{\sum_{i=1}^{n-1} (\bar{x}-x_i)^2}{n} & \frac{\sum_{i=1}^{n-1} (\bar{x}-x_i)(\bar{v}-v_i)}{n} \\ \frac{\sum_{i=1}^{n-1} (\bar{v}-v_i)(\bar{x}-x_i)}{n} & \frac{\sum_{i=1}^{n-1} (\bar{v}-v_i)^2}{n} \end{bmatrix} \quad (7.8)$$

$$\begin{bmatrix} \sigma_x^2 & \sigma_x \sigma_v \\ \sigma_v \sigma_x & \sigma_v^2 \end{bmatrix} \quad (7.9)$$

De covariantiematrix voor 2D wordt algemeen gegeven door vgl (7.9). Wanneer de versnelling erbij komt wordt deze nog uitgebreider.

### 7.3.3 Nieuwe staat / voorspelde waarde

Hier wordt de staatmatrix voorspeld. In deze context zal de algemene vergelijking (7.10)<sup>1</sup> ingevuld worden met de bewegingsvergelijking (7.11)<sup>2</sup>.

$$X_k = AX_{k-1} + Bu_k + w_k \quad w_k \sim \mathcal{N}(0, Q_k) \quad (7.10)$$

Onder het  $AX_k$  gedeelte vallen de lineaire stukken en  $Bu_k$  de variabelen die de toestand wijzigen. Bij de bewegingsvergelijking beslaat de toestand die we hebben de positie en snelheid, en is de wijzigende factor de versnelling. In vgl (7.12) zien we bij de vermenigvuldiging van  $AX_k$  en  $Bu_k$  de bewegingsvergelijkingen van vgl (7.11) verschijnen.

$$\begin{cases} x_k = x_{k-1} + \dot{x}_{k-1} \Delta t + \frac{1}{2} \ddot{x}_{k-1} \Delta t^2 \\ v_k = v_{k-1} + \dot{x}_t \end{cases} \quad (7.11)$$

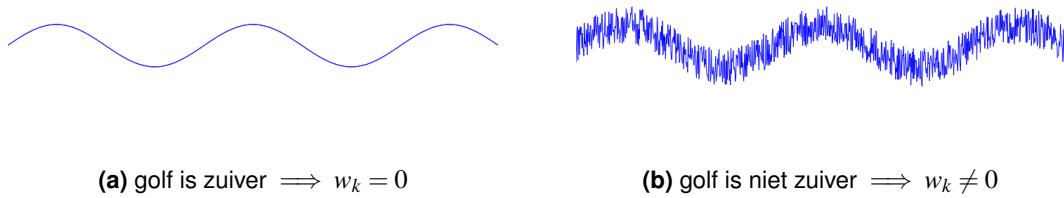
Deze vorm krijgen we slechts wanneer we enkel positie en snelheid als resultaat willen. Wanneer we versnelling willen in de staatmatrix zoals in vgl(7.5) wijzigt de opstelling en krijgen we 1 grote matrix waarin beide verwerkt zijn zoals in vgl (7.13). In de literatuur zullen de meeste vormen wel verwijzen naar de eerste voorstelling. Omdat de versnelling niet rechtstreeks gemeten wordt en om een vorm van eenhouder te behouden in de matrices die anders allemaal aangepast moeten worden te vermijden volgen we de versie van vgl (7.13).

De  $\Delta t$  is een constante waarde. Deze wordt gehaald uit een interpolatie van het GPS-signal. Dit maakt de verwerking sneller aangzien deze waarde niet telkens opgevraagd en aangepast dient te worden.

In de berekeningen zal worden gewerkt met een 1D voorstelling. De aangelegde weg  $s$  wordt niet in 2 richtingsvectoren opgesplitst. Dit heeft verschillende redenen. Ten eerste voor de eenvoud in de berekeningen. De afstands bepaling na de Vincenty zo ofwel opgesplitst moeten worden in 2 arbitraire loodrechte vectoren of de afstand bij elke coördinaat zo telkens afzonderlijk van latitude en longitude gedaan moeten worden. Ten tweede is de onzekerheid die bij een plaatsbepaling met GPS'en gebeurt de plaats tussen intersectie van 3 bollen. De onzekerheid kan niet toegepast

<sup>1</sup>in de literatuur wordt gesproken van state-space model, dit slaat op de wijze om een zo alomvattende formule te hebben voor het wiskundig modelleren van en fysisch systeem die input, output en toestandsvariabelen heeft

<sup>2</sup>hier kan bvb een vallend voorwerp, water niveau in een vat en dergelijke ingevuld worden mits kleine aanpassingen



**Figuur 7.8:** Verduidelijking  $w_k$

worden op een latitude en longitude afonderlijk, zonder extra aanpassingen wat het nodeloos ingewikkeld maakt. En ten derde zou het runtime van de berekeningen in het berekeningsprogramma verlengen.

$$AX_k = \begin{bmatrix} 1 & \Delta T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} \quad Bu = \begin{bmatrix} \frac{1}{2}\Delta T^2 \\ \Delta T \end{bmatrix} \begin{bmatrix} \ddot{x} \end{bmatrix} \quad (7.12)$$

$$AX_k = \begin{bmatrix} 1 & \Delta T & \frac{1}{2}\Delta T \\ 0 & 1 & \Delta T \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \end{bmatrix} \quad (7.13)$$

De laatste term  $w_k$  is het ruis op de eigenlijke toestand. Het is belangrijk om een duidelijk onderscheid te maken tussen de onzekerheden in deze cyclus. Hier wordt bij de implementatie in het Scilab-script op teruggekomen. Omdat dit in termen met GPS moeilijk te vertalen valt wordt in Figuur 7.8 een golf weergegeven. Wanneer we een golf proberen te schatten a.d.h.v. een signaal gaan we uit van een golf met constante amplitude en frequentie. Deze valt te berekenen met de golfvergelijking. Maar wanneer de ware waarde/positie van deze golf ruis bevat valt dit buiten de te voorspellen formule. Dus wordt er een onzekerheid bijgevoegd/verondersteld. Een visueel eenvoudiger voorbeeld is een vliegtuig dat gaat landen. Men zou dit kunnen voorstellen met een mooie dalende curve. Maar de reële hoogte van het vliegtuig kan met kleine schokken werken door turbulentie.  $w_k$  is dus een marge die het ideale model uitbreidt tot een realistischer model. In vgl (7.10) wordt ook vermeld dat het om een normale verdeling gaat. Dit wordt aangetoond met het symbool  $\mathcal{N}$  en  $(0, Q_k)$  wat staat voor een verdeling rond 0 en covariantie  $Q_k$ .

Vervolgens wordt nog naar de covariantiematrix van de voorspelde staat gekeken. Deze bestaat uit  $A$  die reeds werd besproken,  $P_{k-1}$  de covariantie van de vorige staat en  $Q_k$ .  $Q_k$  is de variantie op  $w_k$ , een term dewelke reeds besproken werd. Ze wordt berekend met vgl (7.14). Deze term zorgt er ook voor dat de covariantie matrix niet te klein zal worden. Zo zal de Kalman-gain (7.19), die hierna zal besproken worden, niet te klein worden. Om een duidelijk beeld te hebben van hoe elke matrix er uit ziet is dit van vgl (7.15) gegeven in vgl (7.16)

$$Q = BB^T \sigma_a^2 = \begin{bmatrix} \frac{\Delta t^4}{4} & \frac{\Delta t^3}{2} \\ \frac{\Delta t^3}{2} & \Delta t^2 \end{bmatrix} \sigma_a^2 \quad (7.14)$$

$$P_{pk} = AP_{k-1}A^T + Q_k \quad (7.15)$$

$$P_{pk} = \begin{bmatrix} 1 & \Delta T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_x^2 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \Delta T & 1 \end{bmatrix} + \begin{bmatrix} \frac{\Delta t^4}{4} & \frac{\Delta t^3}{2} \\ \frac{\Delta t^3}{2} & \Delta t^2 \end{bmatrix} \sigma_a^2 \quad (7.16)$$

### 7.3.4 Nieuwe meting

De eerste meting na initialisatie van de lus, of eerstvolgende  $k$  wanneer de cyclus reeds 1x doorgaan is wordt gegeven door  $Y_k$ . De formule hiervoor is gegeven in vgl (7.17) Deze input is het enige dat bij deze lus ingevoerd moet worden nadat de cycli van start is gegaan. Deze komt in dezelfde vorm als de staat matrix en de vorm is vergelijkbaar met  $AX_{k-1}$  in vgl (7.10). is wederom een aanpassing aan de ideale vorm, zodat rekening gehouden kan worden met de mogelijks foutieve waarde. Dit is de ruis in het proces. Dit is verschillend van de foutmarge die gegeven wordt op de verpakking van een meettoestel. Deze komt pas ter sprake bij de update van de staat matrix met de nieuwe meting.  $Z_k$  wordt echter niet toegevoegd bij de berekeningen. Het staat er ter volledigheid als onbekende onzekerheidsvariabele. Bij simulaties kan hier een random fout binnen de marge toegevoegd worden.

$$Y_k = CX_{km} + Z_k \quad Z_k \sim \mathcal{N}(0, R_k) \quad (7.17)$$

$X_{km}$  is de staatmatrix. Gezien we enkel een positie meten en niet rechtstreek een snelheid zal hierin slechts de bovenste waarde verschillen. Dit zou hoe dan ook er uit gehaald kunnen worden met  $C$ . De matrix zorgt ervoor dat dit automatisch gedaan wordt. Zo blijft de vorm van de staatmatrix consistent. Dit is zichtbaar in vgl (7.18)

$$Y_k = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} + [Z_k] \quad (7.18)$$

### 7.3.5 Update met nieuwe meting

Bij de update van de nieuwe staat matrix (7.20) wordt gebruik gemaakt van de Kalman-gain (7.19).

Deze Kalman-gain is de spil van het hele systeem. Ze bepaalt welk gewicht aan het verschil tussen de onzekerheid bij voorspelling van de staatmatrix en deze bij de meting (7.20). De Kalman-gain wordt eerst besproken vooraleer naar de geüpdate staatmatrix gegaan wordt zoals het ook in de berekeningen gebeurt.

$$\begin{cases} K = \frac{P_{kp}H^T}{HP_{kp}H^T + R} \\ X_k = X_{kp} + K[Y_k - HX_{kp}] \end{cases} \quad (7.19) \quad (7.20)$$

Hierbij is  $H$  de Jacobi-matrix is van de sensor. Dit is een matrix van eerste-orde partiële afgeleiden. Een voorbeeld hiervan is gegeven in de matrix (7.21) waarbij  $f = f(x_i)$ .

$$J(f) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \dots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \dots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \dots & \frac{\partial f_m}{\partial x_n} \end{bmatrix} \quad (7.21)$$

Bij een implementatie van de Jacobi die toegepast wordt op  $C$  komt men een gelijke matrix uit bij onze toepassing waarbij enkel een positie en geen snelheid gegeven wordt aan het systeem.

We zijn eindelijk gekomen bij de onzekerheid van de meetsensor  $R$ . Dit is een diagonaal matrix waarbij de onzekerheden in het kwadraat gegeven worden. Doordat we maar 1 term invoegen is dit een  $[1x1]$  matrix zoals gegeven in vgl (??) Wanneer er meerdere sensoren zijn die afhankelijk zijn van elkaar moeten de onzekerheden vermenigvuldigd worden. De matrix blijft steeds symmetrisch langst de diagonaal, of gelijk aan zijn getransponeerde vorm.

We kunnen 3 duidelijke invloeden onderscheiden bij de Kalman-gain :

- $R \rightarrow 0$  : daardoor zal  $K \rightarrow 1$  en dan zal de aanpassing van de staatmatrix voornamelijk gedaan worden met de meting.
- $R \gg 0$  dan zal  $K \rightarrow 0$  waardoor de aanpassing van de staatmatrix voornamelijk gedaan wordt met de voorspelde staat.
- $P \rightarrow 0$  zal de update van de metingen voor het merendeel verwaarloosd worden.

De staatmatrix wordt geüpdate in vgl (7.20). De termen die gebruikt worden zijn reeds besproken. Hier wordt bij de voorspelde waarde van  $X$  bij het verschil tussen de gemeten en voorspelde waarde die een bepaalde grootte krijgt met behulp van de Kalman-gain gevoegd.

### 7.3.6 Output van de geüpdate staat

Om de waarden die de kalmanfilter zal uitsturen te verkrijgen moet juist nog de covariantiematrix aangepast worden. Deze wordt verkregen door van een eenheidsmatrix het veelvoud van de Kalman-gain met de Jacobi matrix af te trekken en het resultaat te vermenigvuldigen met de voorspelde covariantiematrix.

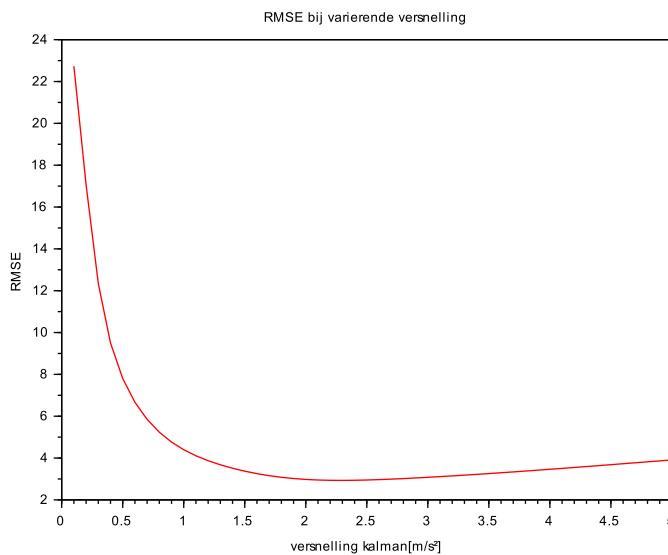
$$\begin{cases} X_p \\ P_k = (I - KH)P_{kp} \end{cases} \quad (7.22)$$

## 7.4 Resultaten met de Kalman-filter

De kalmanfilter is uitgeprobeerd op enkeleritten gemaakt met de meetfiets. Het probleem bij de kalmanfilter is dat de onzekerheid op de fout niet constant is. Bij een grote mogelijke fout op de versnelling van de data zal de gemeten data waarschijnlijker zijn, maar dus ook meer de grillen van de ruwe data volgen. Bij een kleine fout op de versnelling van de gemeten data (de ruwe data van de app) zal de app nauwelijks rekening houden met de gemeten data en slechts weinig variëren omdat de snelheid van de vorige data waarschijnlijker geacht wordt dat de nieuwe gemeten data.

### 7.4.1 RMSE op Kalmanresultaat

Er is een lus over 5 verschillende metingen gegaan met een totaal van meer dan 2500 datapunten die versnellingen van 0,1-5 m/s in stappen van 0,1 m/s uitprobeerde. De data hierna werd punt per punt vergeleken, tussen deze van de meetfiets en de gefilterde data. Hetzelfde werd gedaan tussen de meetfiets en de ruwe data zodat de verbetering tussen de ruwe data en de gefilterde kon beschouwd worden. Bij elk punt van de app werd het kwadratisch gemiddelde van de fout(=RMSE)



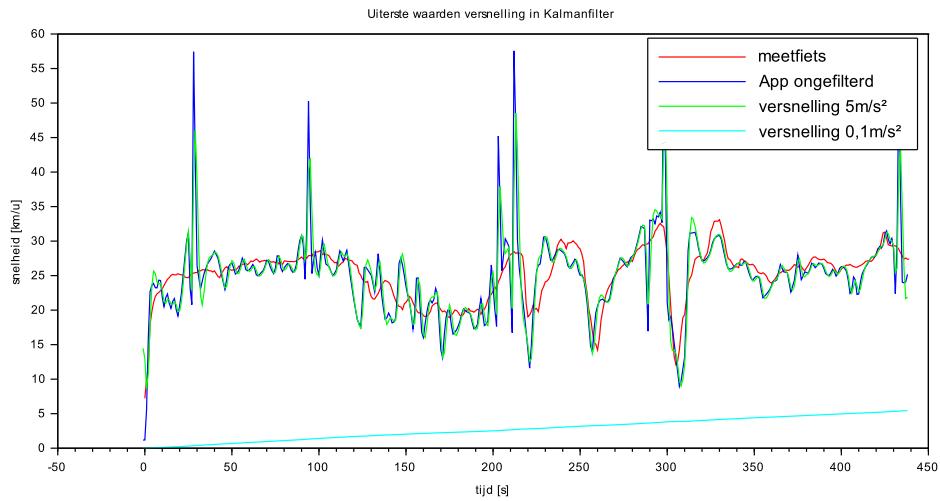
Figuur 7.9: RMSE ifv versnelling in kalmanfilter

bepaald ten opzichte van de meetfiets. Daar werd een gemiddelde van genomen. De RMSE staat dus voor de fout op de ogenblikkelijke snelheid van de app tov de meetfiets. De formule voor RMSE wordt gegeven in (7.23)

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (x_{i,app} - x_{i,meetfiets})^2} \quad (7.23)$$

Het verschil in RMSE ifv de snelheid ingegeven in de kalmanfilter is zichtbaar in Figuur 7.9. Het is duidelijk dat bij te kleine waarden van de versnelling zeer slechte waarden voorkomen. Dit kwam doordat de versnellingen die de app kon nemen te klein waren. Zo werd een curve verkregen die slechts heel gestaag klom. Dit is zichtbaar in Figuur 7.10. De gemeten waarden van de app werden dus als extreem onwaarschijnlijk aanschouwd en de volgende waarde werd bepaald op basis van de berekende waarde. Die dan een zeer kleine versnelling kende aan het begin van de meting. Er komt een minimum bij de RMSE waarna deze wederom de lucht in gaat. Dit gaf een tegengesteld probleem. Bij meetwaarde waar een unrealistisch grote versnellingen als realistisch werd beschouwd door de Kalman-filter ging de snelheid simpelweg mee met de grilligheid van de ruwe data. Ook dit is te zien in Figuur 7.10. De curve van de ruwe data en deze van de gefilterde met hoge versnelling zijn nauwelijks van elkaar te onderscheiden. Slecht bij versnellingen groter dan 5 m/s volgde gefilterde data de curve van de ruwe data niet. Deze zijn zichtbaar bij de hoogste pieken van de snelheid.

De resultaten van deritten en hun optimale versnelling werden weergegeven in Tabel ???. Er werd gezocht naar de minima in RMSE bij elke curve ifv van de versnelling die de kalmanfilter aanvaard van de ruwe data. Het bleek dat dit verschil was per curve. Dit was geen verassing gezien de versnellingen bij elke curve verschillend waren en de externe parameters die het verschil tov de meetfiets bepalen elke rit verschilden. Omdat niet elke rit met een meetfiets later zou kunnen vergeleken worden moest er dus een waarde gekozen worden voor alle komenderitten. Het gemiddelde uit de optimale versnellingen werd gekozen. Met deze versnelling werd dan wederom



**Figuur 7.10:** Extreme versnellingen in Kalmanfilter

Runkeeper	$RMSE_{ruwedata}$	$RMSE_{optim}$ [km/u]	$a_{optim}$ [m/s]	$RMSE$ bij $a_{gem} = 2,7$
rit 1	4,43	2,99	2,3	2,99
rit 2	4,54	3,37	1,7	3,71
rit 3	5,09	4,4	3,3	4,48
rit 4	4,61	3,6	2,9	3,61
rit 5	2,9	2,19	3,3	2,21
gemiddelde	4,31	3,31	2,7	3,40

**Tabel 7.2:** Optimale versnelling volgens RMSE bij rk

bekijken wat de RMSE waarden waren tov de fiets. Deze bleek in alle gevallen beter dan de ruwe data van de app. Wat kon uiteindelijk afgelezen worden uit deze tabel? De meest optimale RMSE uit deze ritten was 3,31 km/u. Gezien er een algemene waarde moest gekozen worden voor de versnelling in de kalmenfilter werd deze als gemiddelde gezet op 2,7 m/s<sup>2</sup>. Wanneer dit op alle ritten werd toegepast kwam de gemiddelde RMSE over de 5 ritten op 3,4 km/u. Dit was 1 km/u beter dan bij de ruwe data en slechts 0,09 km/u minder dan het gemiddelde van de meest optimale filtering bij alle ritten.

Er werd met 2 app's gewerkt, rk en ctg. Om te zien of dit resultaat ook opging voor ctg werd een gelijke test gedaan. Wederom 5 ritten met een gelijkaardig aantal meetpunten. Bij de keuze van apps is gekozen om rk en ctg te combineren omdat ze een gelijkaardige  $\xi$  waarde hadden. De verwachtingen waren dan ook dat een gelijkaardige versnelling zou gebruikt moeten worden. Dit werd bevestigd door de resultaten in Tabel 7.3. De optimale versnelling  $a_{optim} = 2,2$  ligt hier wel lager. Wanneer de optimale versnelling van bij Runkeeper gebruikt werd gaf de verbetering van de gefilterde data tov de ruwe data een verbetering van 1 km/u. Een beter versnelling zou kunnen gebruikt worden voor ctg, maar dit wordt achterwege gelaten om alle data uit de apps met een gelijke onzekerheid te behandelen. Er wordt in het vervolg van dit werk ook geen onderscheid gemaakt tussen de resultaten van rk en ctg.

Als laatste werd nog gekeken naar de speed pedelec. Is de versnelling hier anders? Het zou

Cycle Tracks GPS	$RMSE_{ruwedata}$	$RMSE_{optim}$ [km/u]	$a_{optim}$ [m/s]	$RMSE$ bij $a_{gem} = 2,7$
rit 1	6,41	3,21	2	3,95
rit 2	5,05	3,92	1,9	4,46
rit 3	2,40	1.98	2,7	1,98
rit 4	4,8	4,16	2,1	4,29
rit 5	3,75	3,07	2,1	3,40
gemiddelde	4,48	3,9	2,2	3,4

**Tabel 7.3:** Optimale versnelling volgens RMSE bij ctg

Speed pedelec	$RMSE_{ruwedata}$	$RMSE_{optim}$ [km/u]	$a_{optim}$ [m/s]	$RMSE$ bij $a_{gem} = 2,7$
rit 1	8,09	5.05	2,5	5,23

**Tabel 7.4:** Optimale versnelling volgens RMSE bij speed pedelec

logisch lijken dat er wat gewijzigd zou moeten worden. De versnelling van de speedpedelec ligt tussen 0,8-1,2 m/s<sup>2</sup> bij versnellen en tot 2,6 m/s<sup>2</sup> bij bruusk afremmen <sup>?</sup>. Er werd 1 rit gefiets met ongeveer 1220 datapunten.

De fout was zoals verwacht groter dan bij de pedelec. 8,09 m/s bij de ruwe data tov 4,4m/s bij de ruwe data bij de pedelec. De optimale snelheid lag wel dicht bij de 2,7 m/s<sup>2</sup>. Maar de uiteindelijke RMSE bij 2,7 m/s<sup>2</sup> ligt 1,8 km/u hoger dan bij de pedelec. Er moet wel bij vermeld worden dat deze rit met de speedpedelec frequent gestopt is in tegenstelling tot de ritten met de pedelec. Ook is er een ander parcour genomen dan bij de ritten met de pedelec. Dit zou invloed gehad kunnen hebben op deze resultaten.

Het spreekt voor zich dat deze resultaten niet zaligmakend zijn. Er zit nog steeds een vertraging op de app waardoor er verschuivingen zijn. Deze kunnen bij versnellingen of vertragingen grote RMSE als gevolg hebben. Hierdoor zal de werkelijke RMSE lager liggen dan hier berekend is. Bij de speed pedelecs zijn deze versnellingen groter waardoor dit effect nogmaals versterkt wordt.

#### 7.4.2 $\chi^2$ op Kalmanresultaat

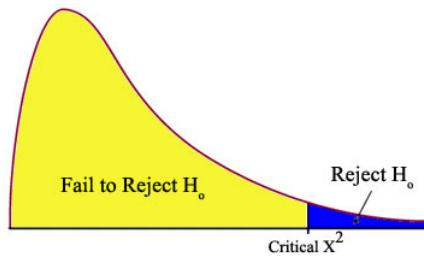
Stel dat de vraag werd gesteld : "Hoe lang werd aan een bepaalde snelheid gereden?", geeft de kalmanfilter hier dan nog een correct beeld van? Om histogrammen te onderling te kunnen vergelijken moest gezocht worden naar een wiskundige methode die dit toestond. Het was geen normale of binominale verdeling wat de opening liet om  $\chi^2$ -test uit te voeren<sup>3</sup>. Er werd maw. gekeken naar de homogeniteit tussen de curves. De waarden die hieronder aan bod komen werden niet in percentages gezet gezien het aantal meetwaarden van de app en meetfiets gelijk lag. Het procentueel uitdrukken gaf niet noodzakelijk een beter inzicht daardoor en zou enkel deze waarden delen door het aantal meetpunten van de rit(ten).

Deze formule wordt gegeven door vgl (7.24). Waarbij  $N_{i,j}$  het aantal waarden i voor een steekproefgrootte j van de metingen is en  $N_{i,j}^*$  de referentiewaarden of anders gezegd het aantal verwachte waarden bij gegeven getoetste waarde. In dit geval waren dit de bemande punten bij een bepaalde snelheid

<sup>3</sup>De Kolmogorov-Smirnov test was een waardig alternatief geweest. Maar er is gekozen voor de meest eenvoudige methode.

<i>f</i>	<i>d</i>											
	.005	.010	.025	.050	.100	.250	.750	.900	.950	.975	.990	.995
1	.000	.000	.001	.004	.016	.102	1.32	2.71	3.84	5.02	6.63	7.88
2	.010	.020	.051	.103	.211	.575	2.77	4.61	5.99	7.38	9.21	10.6
3	.072	.115	.216	.352	.584	1.21	4.11	6.25	7.81	9.35	11.3	12.8
4	.207	.297	.484	.711	1.06	1.92	5.39	7.78	9.49	11.1	13.3	14.9
5	.412	.554	.831	1.15	1.61	2.67	6.63	9.24	11.1	12.8	15.1	16.7
6	.676	.872	1.24	1.64	2.20	3.45	7.84	10.6	12.6	14.4	16.8	18.5
60	35.5	37.5	40.5	43.2	46.5	52.3	67.0	74.4	79.1	83.3	88.4	92.0

Figuur 7.11: kritieke grootheid

Figuur 7.12: Kritieke grootheid op  $\chi^2$ -curve

van de meetfiets.

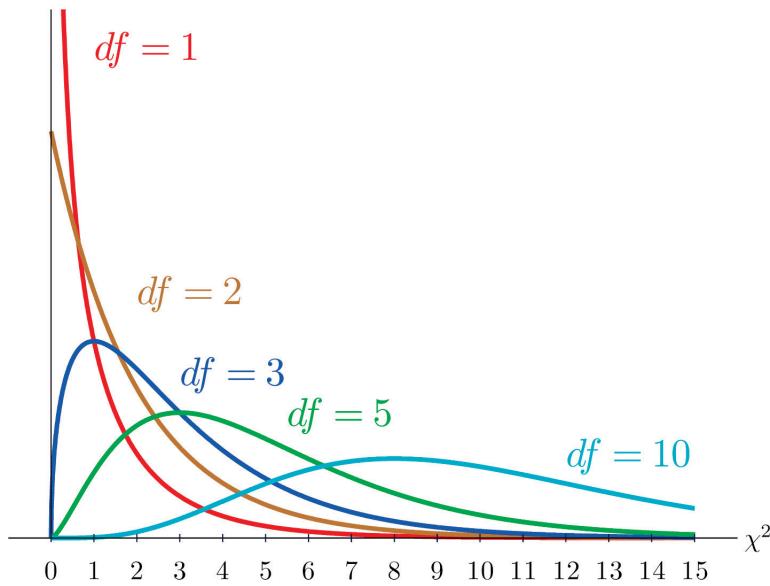
$$\chi^2 = \sum_{i,j} \frac{(N_{i,j} - N_{i,j}^*)^2}{N_{i,j}^*} \quad (7.24)$$

Normaal gezien wordt er bij de  $\chi^2$  een nulhypothese gesteld en wordt ze getoetst aan een grootheid, kritieke grootheid genoemd, die wordt bepaald door het aantal vrijheidsgraden. Een voorbeeld van de tabel met kritieke grootheden is te zien in Figuur 7.11. Wanneer de  $\chi^2$ -waarde de kritieke grootheid overschreidt wordt gesteld dat de nulhypothese mag verworpen worden. Dit principe wordt beter geschat in Figuur 7.12. Wanneer het aantal vrijheidsgraden verhoogd wordt wordt de curve platter en de kritieke waarde dus ook groter. Het effect van het aantal vrijheidsgraden ( $=df=$ degrees of freedom) is duidelijk in Figuur 7.13.

De nulhypothese is hier dan  $H_0 : \text{de app heeft eenzelfde verdeling als de meetfiets}$ .

Er werd eerst gedacht alle gelogde snelheden op te delen binnen bereiken van 1 km/u. Er stelden zich 3 problemen voor.

- *1e probleem* : Om de  $\chi^2$ -waarde te bepalen werd er gedeeld door het aantal verwachte waarden bij het aantal metingen van de testwaarde ( $= N_{i,j}^*$ ). Wanneer deze waarde 0 werd ging de limiet van de uitkomst naar oneindig. Een voorbeeld van de waarden bij een rit zonder stilstand is te vinden in Figuur 7.14. Dit kon niet verrekend worden. Wel was het duidelijk dat de extreme waarden in Figuur 7.14 als het ware naar binnen werden getrokken. Fisher's exacte toest zou kunnen gebruikt worden gezien deze met alle waarden kan werken. Maar deze methode is bedoeld voor kleine steekproeven. Bij grotere steekproeven wordt het rekenwerk immens groter.
- *2e probleem* : Een verdeling maken met een snelheid van 1 km/u terwijl de nauwkeurigheid hoger ligt zoals gezien bij de RMSE. Wanneer deze waarden dan met de meetfiets vergeleken

Figuur 7.13: Het effect van df op de  $\chi^2$ -curve

worden is dit een fout die niet in rekening gebracht kan worden. Dit probleem werd weergegeven in Tabel ???. Hierbij was  $df=18$ . De snelheden van 16 tem. 30 km/u werden ingedeeld in groottes van 1 km/u. De  $\chi^2$ -waarden lagen ver van de kritieke waarden. De verbeteringen waren ook niet noemenswaardig.

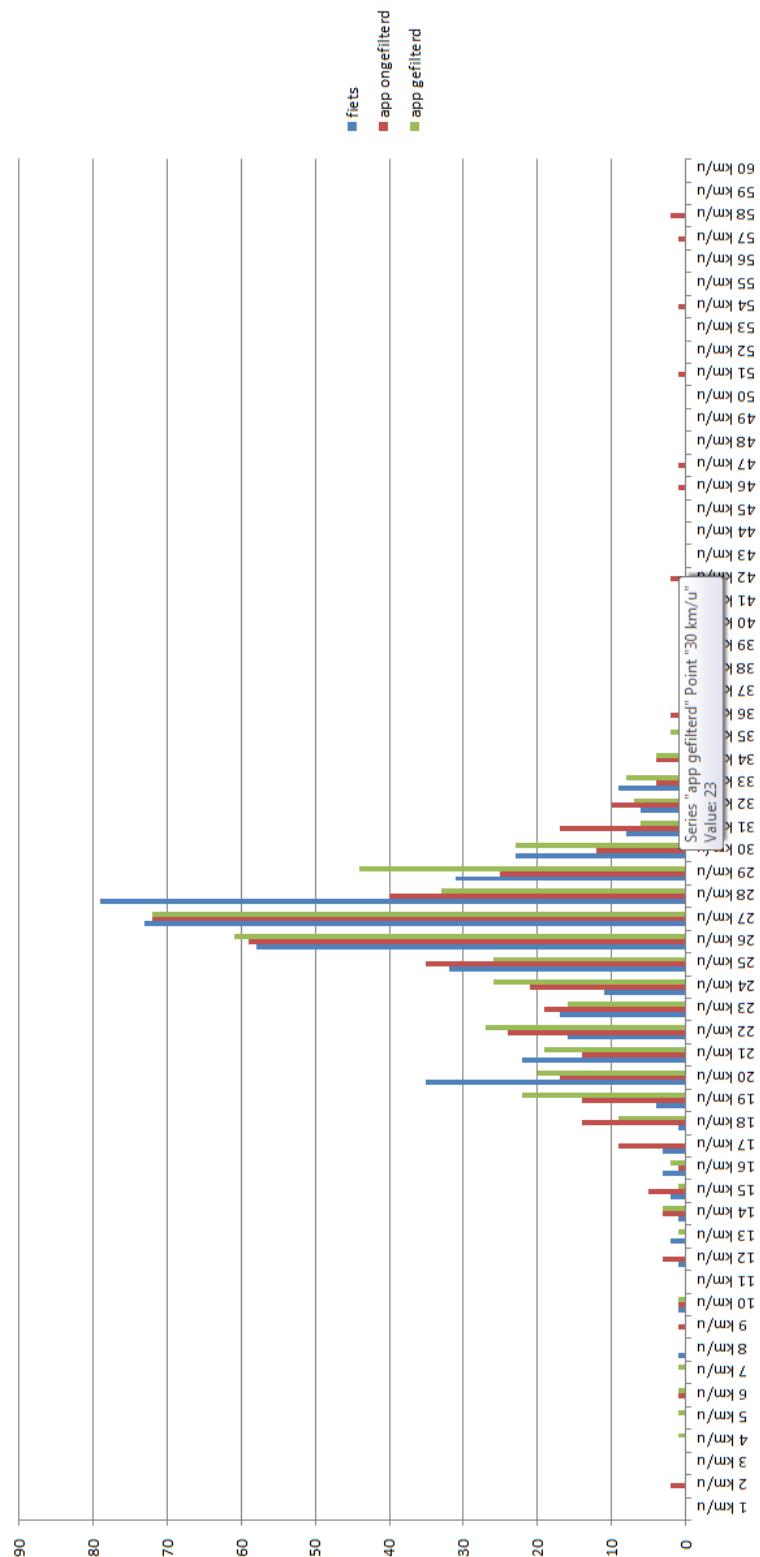
- **3e probleem :** Hoe groot moesten de bereiken dan wel gekozen worden en tussen welke waarden? Bij een gebrek aan betere alternatieven werd er arbitrair voor afgeronde waarden gekozen met een bereik dat groter was dan de RMSE.

Een gelijkaardige procedure als bij RMSE werd gebruikt. Er werd eerst gekeken naar de kleinste  $\chi^2$ -waarde die kan bereikt worden door de versnelling in de Kalman-filter te laten variëren. Daarna werd gekeken welke  $\chi^2$ -waarde er werd bekomen wanneer de  $\chi^2$ -waarde met de versnelling gekozen door de RMSE.

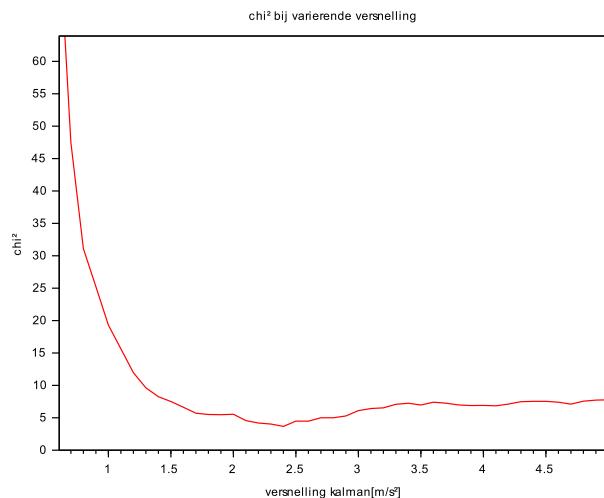
Runkeeper	$\chi^2$ ruwedata	$\chi^2$ optimaal [km/u]	$a_{gem}$ [m/s]	RMSE bij $a_{gem} = 2,7$
rit 1	273,48	133,47	2,3	247,36
rit 2	142,06	60,81	1,8	60,81
rit 3	184,04	86,8	1,3	152,82
rit 4	369,54	160,65	1,3	420,48
rit 5	127,69	109,6	2,3	171,06
gemiddelde	219,3624	110,27	1,8	210,5

Tabel 7.5: Vergelijking histogrammen na kalmanfilter met  $\chi^2$  en  $df=18$ 

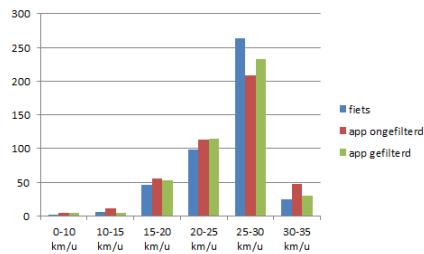
Het viel wederom op dat er een gelijkaardige curve was als bij de RMSE. Dit is zichtbaar in Figuur ???. Deze curve is ietwat grilliger dan bij de RMSE. Dit viel te verklaren doordat de verschuiving van de snelheden niet telkens naar het juiste bereik ging. In Tabel 7.6 werden de waarden ingedeeld volgens bereiken zichtbaar in Figuur7.16. De bereiken waren niet constant 5 km/u. Dit kwam



Figuur 7.14: Histogram van de snelheden in bereiken van 1 km/u



Figuur 7.15: Histogram df=6



Figuur 7.16: Histogram van de snelheden met bereiken ;RMSE 3,4 km/u

doordat er niet werd stilgestaan met de meetfiets en de meetfiets geen waarden boven 35 km/u. Gezien de app wel snelheden had is er gegaan tot 60 km/u. Dit is de grens waarbij reeds automatisch meetwaarden werden verworpen. Hoger kwam dus niet voor in de metingen.

Runkeeper	$\chi^2$ ruwedata	$\chi^2$ optimaal [km/u]	$a_{gem}$ [m/s]	RMSE bij $a_{gem} = 2,7$
rit 1	48,60	13,48	2,4	25,03
rit 2	73,60	4,40	1,5	72,54
rit 3	59,91	43,98	1,3	66,06
rit 4	257,73	31,02	1,3	249,12
rit 5	96,55	36,80	3,4	38,80
gemiddelde	96,55	25,94	1,98	90,31

Tabel 7.6: Vergelijking histogrammen na kalmanfilter met  $\chi^2$  en df=6

Uit de tabel in Figuur 7.11 kon gezien worden dat de kritieke waarde bij 6 vrijheidsgraden die zichtbaar zijn in Figuur 7.16 gelijk is aan 18,5 waarbij er 0,5 % kans is dat de nulhypothese moet verworpen worden maar toch een gelijke verdeling heeft. Zelfs bij de kleinst mogelijke  $\chi^2$ -waarden zijn maar 2 van de 5 ritten die er onder liggen. Dit is bij de algemene versnelling van  $2,7\text{m/s}^2$  nooit het geval. Het is wel zichtbaar dat er een verbetering was. Weliswaar slechts een kleine

verbetering. Dit kon liggen aan het feit dat de bereiken slecht gekozen waren. Voor sommige ritten meer dan andere. Of dat de  $\chi^2$  methode geen sluitend antwoord kan bieden op een absolute verbetering die de nieuwe histogrammen met zich meebrengen. De optimale versnellingen liggen lager dan bij de RMSE-test. Dit kon verklaard worden door naar gemiddelde snelheden te kijken. Gemiddelde snelheden zijn correcter bij apps dan instant-waarden. Wanneer de versnelling lager lag ging de curve de grillen minder volgen en meer neigen naar een gemiddelde. Deze waarden kwamen dichter in de buurt bij deze van de meetfiets die minder varieerde door meetfouten.

Gezien de gemiddelde snelheden nauwkeuriger zijn is het beter naar gemiddelde snelheden te kijken over een bepaald trajectdeel dan histogrammen bij een volledige rit.

## 7.5 Besluit smoothers

De beste methode uit de 3 mogelijke smoothers was de Kalman-filter. De onzekerheid bij nieuwe meting  $i$  van de Kalman-filter hoort wordt uitgedrukt in  $m/s^2$ . Deze bepaalt de mate waarmee curves afgevlakt worden. Om een optimale afvlakking te krijgen werd deze gericht naar de minimale RMSE-waarde tussen de meetfiets en de app. Deze bleek op  $2,7 m/s^2$  te liggen. Dezelfde testen met ctg gaven een heel gelijkaardig resultaat. Waardoor er voor ctg eenzelfde versnelling gekozen werd als voor rk. Om de histogrammen te vergelijken werd gekeken naar de  $\chi^2$  homogeniteits test. De histogrammen van de meetfiets en de app werden niet als metingen van een gelijke verdeling erkend. Wel was er na de Kalman-filtering een verbetering merkbaar. Er werd geconcludeerd dat een juist histogram maken moeilijk was door de grote onnauwkeurigheid gevonden in de RMSE, en dat er beter met gemiddelde snelheden wordt gewerkt dan een weergave van ogenblikkelijke snelheden.

# Hoofdstuk 8

## Snelheidszones

In dit deel werd gekeken waar de fietsers welke snelheid haalden. Hiermee kon bepaald worden of speed pedelec gebruikers zich aan de opgelegde snelheden houden, en bvb of in bebouwde kom hun snelheid lager was.

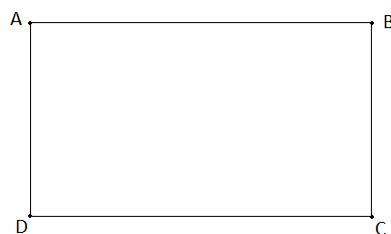
### 8.1 Zones herkennen aan de hand van coördinaten

Het gebruikte verwerkingsprogramma voor alle data was Scilab. Het eenvoudigste zou zijn moesten we hierin kunnen blijven werken. Maar hoe kon geweten worden zonder zicht op de kaart of een fietser zich in een bepaalde snelheidszone bevond?

Bij een rechthoek met een horizontale bovenzijde lijkt het op zich voor de hand te liggen. Bij Figuur 8.1. zou je kunnen zeggen dat punt  $P_{(x,y)}$  binnen de zone ligt wanneer :

- $P_x > A_x \text{ } || P_x > D_x$
- $P_x < B_x \text{ } || P_x < C_x$
- $P_y > D_y \text{ } || P_y > C_y$
- $P_y < A_y \text{ } || P_y < B_y$

Wanneer het ging om een paralelogram of een ruit gingen deze vergelijkingen direct al niet meer op. Dan kon gekeken worden of de afstand tussen het punt en de evenwijdige zijden in beide gevallen kleiner was dan de loodrechte afstand tussen beide zijden. Wanneer dit voor beide zijden gedaan werd, lag het punt ook binnen de vierhoek. Dit werd toegepast in BIJLAGE J. Dit kon



**Figuur 8.1:** Zone afbakening op basis van 4 coordinaten



Figuur 8.2: Zone 50 Gent

```

int i, j, c = 0;
for (i = 0, j = number_of_vertices-1; i < number_of_vertices; j = i++) {
    if ( ((vertices[i].y>p.y) != (vertices[j].y>p.y)) &&
        (p.x < (vertices[j].x-vertices[i].x) * (p.y-vertices[i].y) / (vertices[j].y-vertices[i].y) + vertices[i].x) )
        c = !c;
}
return c;

```

Figuur 8.3: Voorbeeld code raytracing mbv stelling Jordan

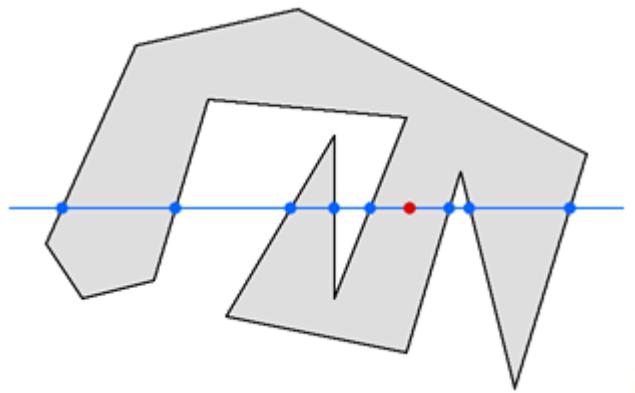
in principe toegepast worden voor het afbakenen van straten. Maar zones worden niet altijd zo weergegeven. Het zou ook voor een lange verwerking zorgen. Vele zones konden gezien worden als polygonen. In Figuur 8.2 zien we een voorbeeld van stad Gent haar zone 30.

Hoe kunnen we dan bepalen of een punt in een polygoon ligt?<sup>1</sup> Er kan met hoeken gewerkt worden, maar de snelste methode maakt gebruik van de techniek *Ray Tracing*. Deze methode maakt gebruik van de stelling van Jordan.

De geïsoleerde code is te zien in Figuur 8.3. Over deze code schrijft de maker( W. Randolph Franklin) van deze verkorte versie : " *The case of the ray going thru a vertex is handled correctly via a careful selection of inequalities. Don't mess with this code unless you're familiar with the idea of Simulation of Simplicity. This pretends to shift the ray infinitesimally down so that it either clearly intersects, or clearly doesn't touch. Since this is merely a conceptual, infinitesimal, shift, it never creates an intersection that didn't exist before, and never destroys an intersection that clearly existed before*"

Maar eenvoudig gezegd komt het hier op neer : Er wordt een oneindig lange horizontale lijn (of ray) getekend op het punt waarvan we willen weten of het al dan niet in de polygoon ligt. De snijpunten met de zijden worden links en recht van het punt bekeken. Wanneer het aantal snijpunten oneven

<sup>1</sup>Naar deze vraag wordt vaak gerefereerd als het PIP probleem of point-in-polygon.



**Figuur 8.4:** Stelling van Jordan

is langst beide zijden ligt het punt in de polygoon. Bij een even nummer langst een van beide zijden is dit niet het geval. Figuur 8.4 schetst dit principe duidelijk. De verklaring van de termen en opbouw in Figuur 8.3, wordt uit de doeken gedaan in BIJLAGE M.  
Het lijkt voor de hand te liggen dat dit dan ook werd toegepast op de coördinaten. Op basis van de coördinaten kon gezegd worden of een fietser zich in een bepaald gebied/polygoon bevond. Maar het afbakenen van de polygoonen lag niet zo voor de hand. De stappen hiervoor werden daarom uitgelegd.

## 8.2 Ingeven van grenzen en gebruiken script

Om aan de data van de snelheidszones verwerkbaar te krijgen moesten enkele stappen genomen worden. Deze stappen worden hier even kort geschetst. Deze methode was omslachtig bij gebrek aan een beter alternatief. Ze is enkel bruikbaar voor proefpersonen vaak eenzelfde route afleggen. a.d.h.v. een proefpersoon X wordt de procedure omschreven in dit hoofdstuk. De resultaten worden besproken in het volgende hoofdstuk. Dit zijn kort de stappen die uitgevoerd werden.

- plot de(vaste) route van de gebruiker van de app
- gemeente/stads-mobiliteitsdienst bereiken om digitale afbakening van de snelheidszones te verkrijgen
- bekijk de zones rondom het traject van de fietser
- ga via Google maps de hoekpunten van de zones af
- geeft de hoekpunten in in een Excel-bestand

### 8.2.1 Plotten route gebruiker app

Om te weten welke gemeentes bekeken moesten worden was het noodzakelijk de route van proefpersoon X te bekijken. Dit ging door het gpx-bestand weer te geven via een site die dit



Figuur 8.5: Weergave traject gpx-bestand

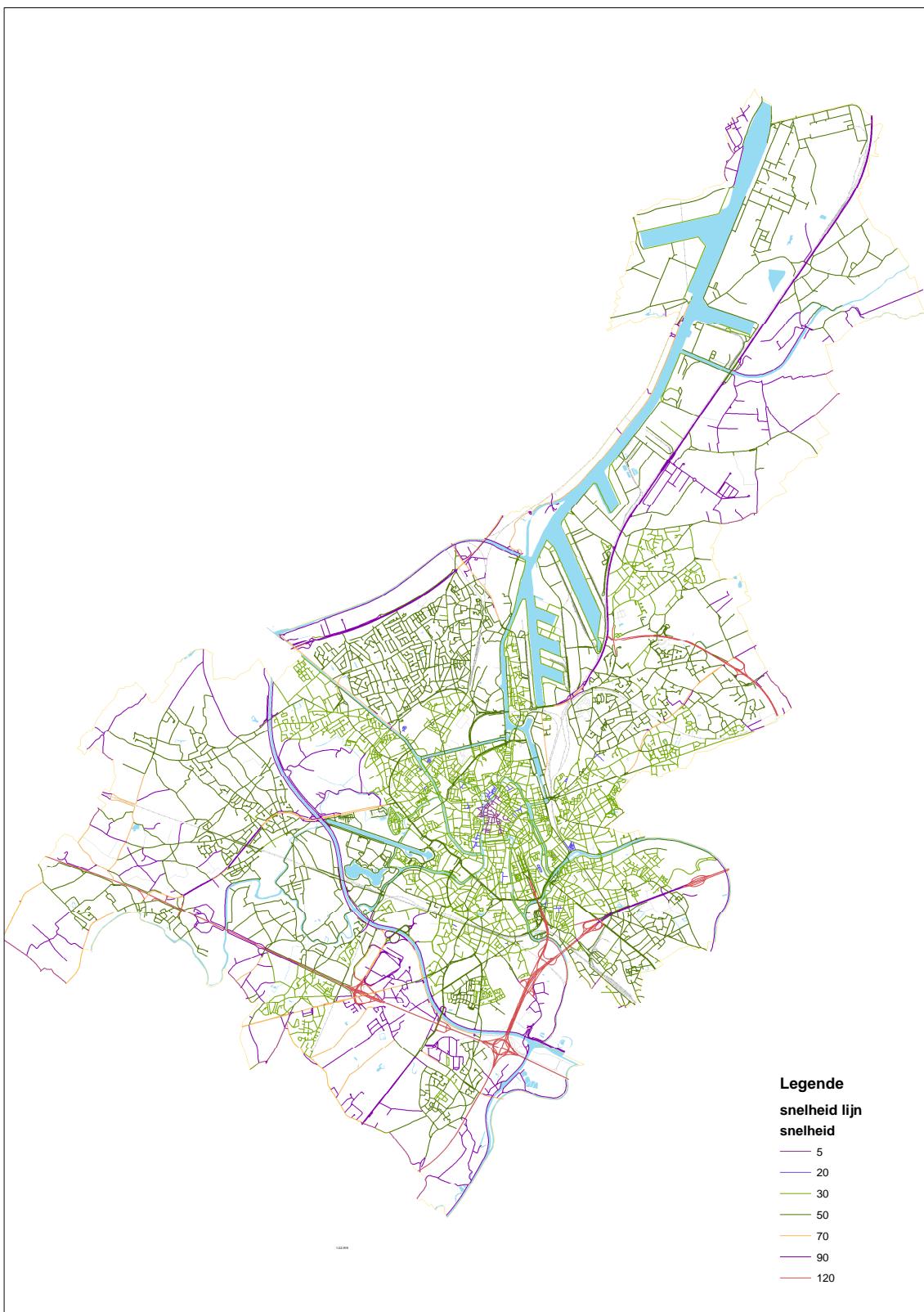
weergaf. gpx-viewer in Google intypen was voldoende om meerdere opties te verkrijgen. Daarin diende het gpx-bestand geüpload te worden. Een voorbeeld van de verkregen weergave is zichtbaar in Figuur 8.5. Dit beeld werd verkregen door de site *GPSvisualiser.com*.

### 8.2.2 Gemeente/stads mobiliteitsdienst bereiken

Waarschuwing :dit onderdeel vergt geduld en een kleine dosis geluk. De gemeenten en steden waarvan je data wilt, hebben steeds een site. Daar waren de departementen zichtbaar en meestal wie te bereiken viel voor mobiliteit. Dit was jammer genoeg niet altijd het geval. Soms moet je doorverwezen worden. Bij StadGent werden er shapefiles die gebruikt werden in ArcGis. Het is aangeraden bij verder onderzoek te vragen aan de dienst in kwestie of ze deze files in hun bezit hebben. Nog belangrijker zijn de referenties, want zonder referenties bleken deze bestanden met veel potentieel waardeloos. Een ander pad werd gezocht en voor gent is een bestand doorgestuurd geweest waar de straten getekend waren, wederom zonder referentie. Er werd besloten te roeien met de riemen die er waren gezien de tijd aan het dringen was. Het bestand is zichtbaar in Figuur 8.6.

### 8.2.3 afbakenen snelheidszones

Nu de snelheidszone verkregen was werd Google maps geopend en ingezoomd op de zones waar fietser X langs was gereden. De zone 30 waar deze fietser doorreed werd afgebakend zoals te zien is op Figuur 8.7 Bij Google maps konden de coördinaten van een punt gevonden worden door simpelweg te klikken op de punt. Zoals reeds gezien is de nauwkeurigheid ongeveer 2 m. Om dit te beperken is steeds gekozen om de coördinaten ongeveer 2m naar het midden van de polygoon te nemen. Zo werd verzekerd dat data die geen zone 30 was, bestempeld werd als 30. Wanneer persoon X meer dan reed in zone 30 was het duidelijk dat dit niet meer in het onzekerheidsgebied lag.



Figuur 8.6: Snelheidszones Gent



Figuur 8.7: Aanduiden zone 30 polygonen

#### 8.2.4 Invoegen in Excel-bestand

Belangrijk bij het Excel-bestand was dat de coördinaten per polygoon wijzerzin of tegenwijzerzin onder elkaar geplaatst werden. Anders zou het Scilab-script een fout maken. Een voorbeeld van het Excel-bestand wordt gegeven in Figuur 8.8. Er werd een naam gegeven bij elke polygoon om voor duidelijkheid te zorgen welke gebieden reeds gedefinieerd waren.

	A	B	C	D	E	F	G	H
1	sint denijs westrem		Henri dunantlaan		Ekkergem		Groenevallei	
2	51,01906	3,657039	51,04842	3,704716	51,04955	3,705412	51,053	3,699757
3	51,0162	3,658364	51,04927	3,702701	51,05058	3,70972	51,05695	3,70932
4	51,0193	3,676792	51,04914	3,702478	51,05575	3,708176	51,58816	3,709597
5	51,02238	3,678971	51,04816	3,704253	51,05255	3,700252	51,06293	3,715115
6	51,026	3,66629			51,05022	3,703647	51,06657	3,712934
7							51,06283	3,695081
8							51,05782	3,693701

Figuur 8.8: Excel-bestand met snelheidszones

Er werd ook een Scilab-script gemaakt die alle polygonen per gemeente/stad in las en ploteert. Op deze wijze kon gezien worden of er fouten gemaakt werden. Een beeld van deze code werd gegeven in Bijlage N.

## **Hoofdstuk 9**

# **Verwerken van ritten van speed pedelec gebruikers**

### **9.1 Algemeen overzicht van te zetten stappen en gebruiken programma's**

Omdat er enkele stappen doorlopen moesten worden om aan de beoogde resultaten te geraken leek het nuttig om ze nog even kort te overlopen. Ze zijn schematisch weergegeven in Figuur 9.1. De stappen werden per vak gezet en de benodigdheden om ze uit te kunnen voeren eronder.

De scripts zelf hebben volgende functies :

*algemene filter :*

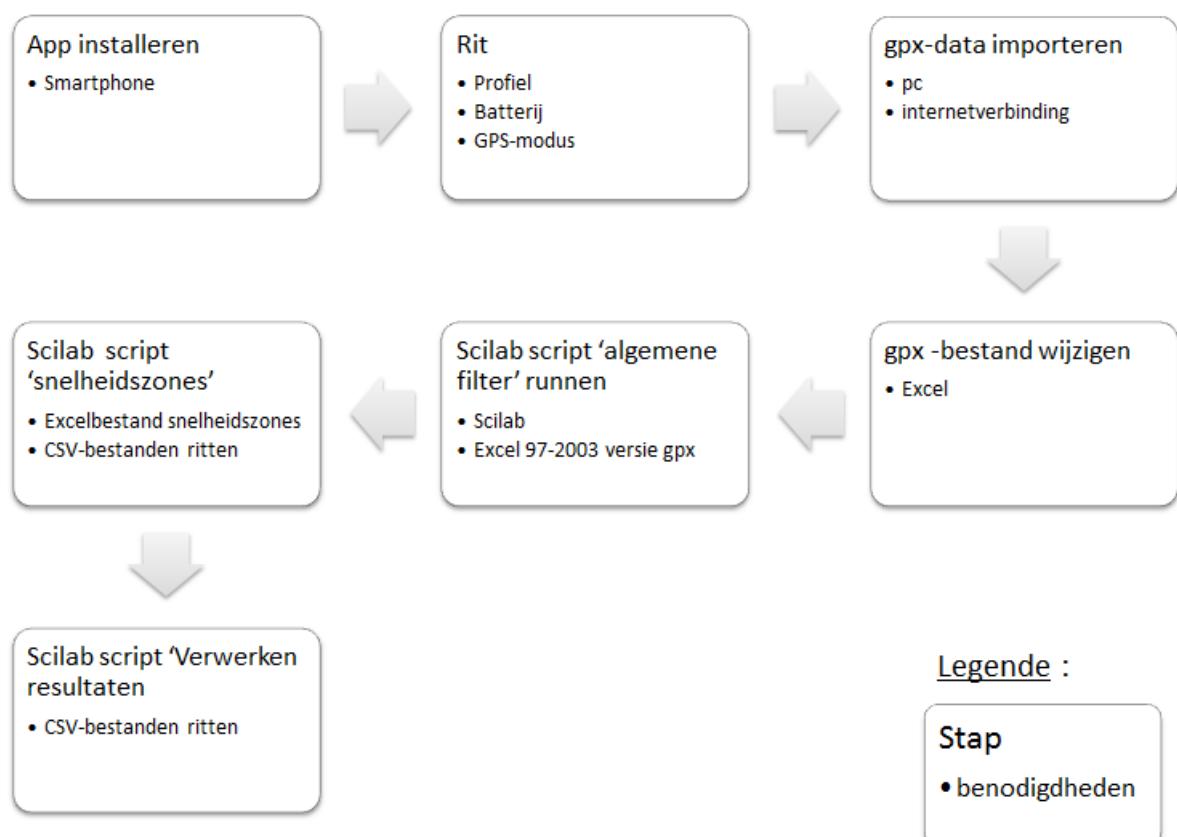
- Excel-bestand inlezen (lus om eventueel meerdere bestanden tegelijk in te lezen)
- Vincenty om van coördinaten naar afstand te gaan
- Stops invoegen ifv sample tijd
- Kalmanfilter
- wegschrijven naar CSV-bestand onder een gelijke naam

*snelheidszones :*

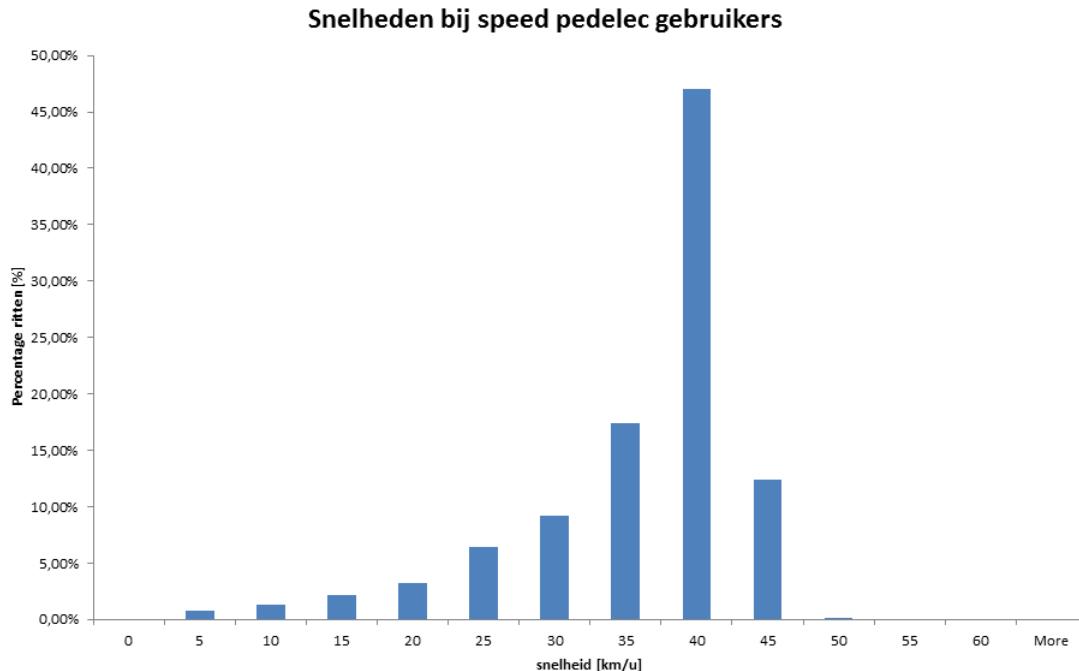
- inlezen CSV-bestand
- inlezen Excel-bestand met snelheidszones
- kijken of punten in polygonen liggen
- kolom toevoegen aan CSV-bestand met snelheidslimiet

*verwerking resultaten*

- inlezen van CSV-bestand(en)
- berekenen van parameters over meerdere bestanden



**Figuur 9.1:** Algemeen overzicht te doorlopen stappen voor verwerking data



**Figuur 9.2:** snelheden bij verschillende voertuigen

- berekenen van parameters in bestand
- grafieken weergeven van 1 bestand
- grafieken weergeven van meerdere bestanden gezamelijk

## 9.2 Resultaten van speed-pedelecgebruikers

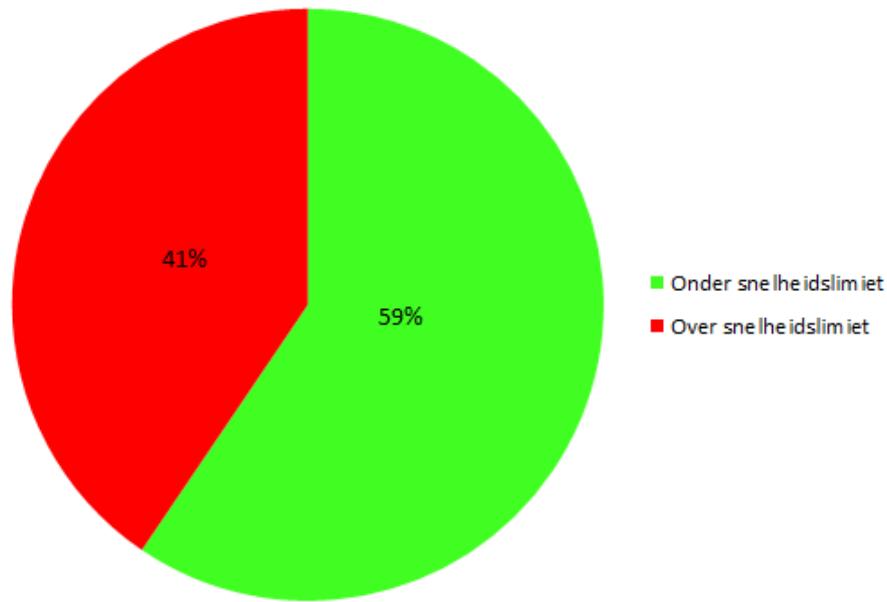
In dit deel zal gekeken worden naar hoe de data van verschillende type gebruikers er uit ziet. Om uitspraken te doen over de gemiddelde afstand en duur van ritten zou er meer naturalistische data nodig zijn van meerdere proefpersonen. Het verzamelen hiervan maakte geen deel uit van dit onderzoek. Wel valt deze data eenvoudig te bekomen via het Scilab-script.

In Figuur 9.2 werd de snelheid over meerdere ritten bij speed pedelecgebruikers bekeken. Hier zijn meer dan 16.000 datapunten in verwerkt, wat neerkomt op ongeveer 5 uur rijdata.

Zo goed als de helft van de rit werd aan 40 km/u afgelegd. Dit is 5 km/u lager dan de maximaal geassisteerde snelheid.

Wanneer er naar ritten gekeken wordt in zone 30 kan, met de RMSE marge afgetrokken, bij speed pedelecs met veel zekerheid gezegd worden dat 41% van het traject in zone 30 is afgelegd aan een snelheid groter dan 30 km/u (zie Figuur 9.3).

Dit is een voorbarige uitspraak als het gaat om alle speed pedelec gebruikers. Dit voorbeeld diende als proof of concept.



Figuur 9.3: Snelheid speed pedelec in zone 30

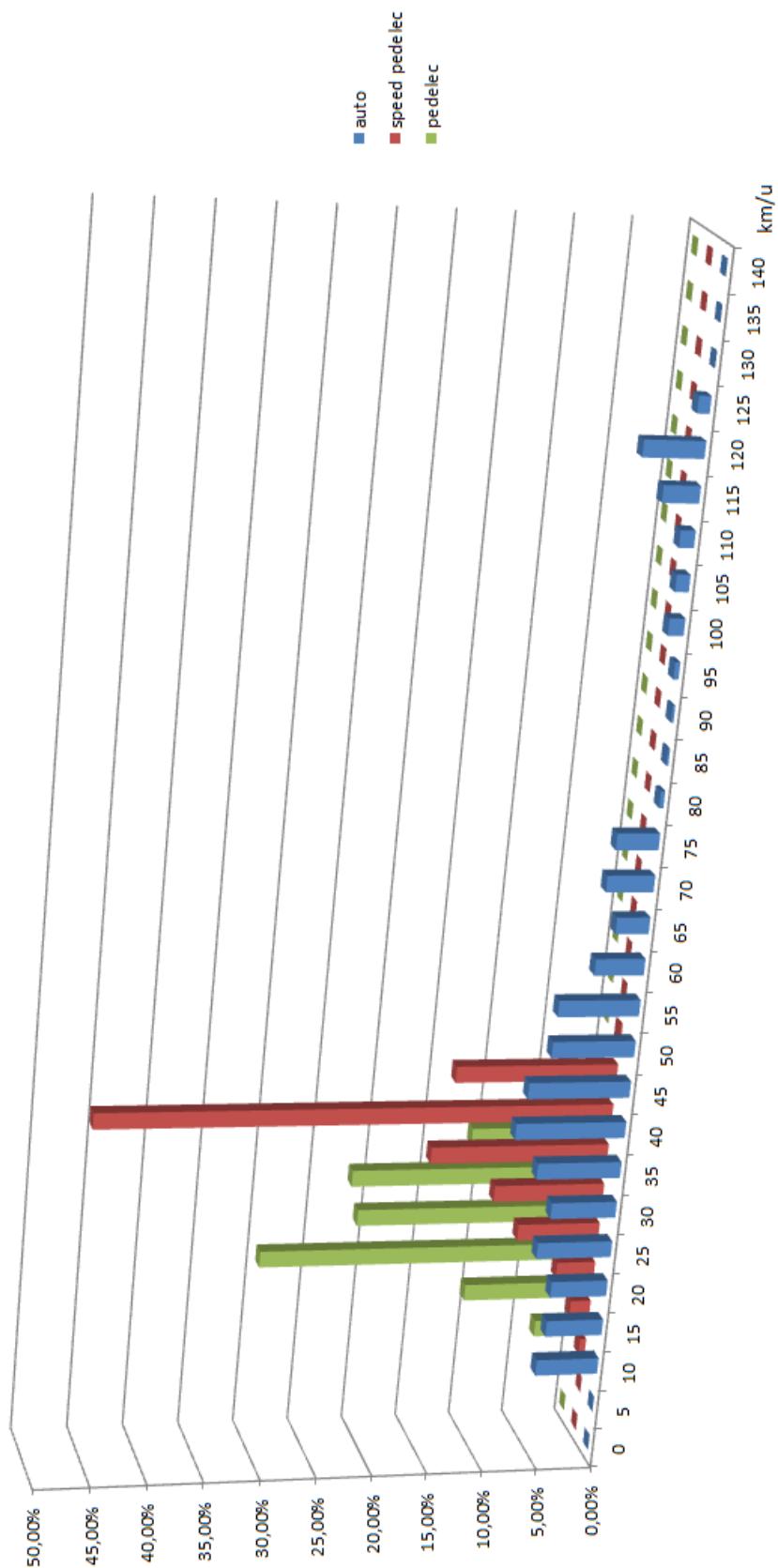
### 9.3 Resultaten bij verschillende voertuigen

Er werd kort even gekeken welke de verschillen waren in resultaten bij de volgende gebruikers :

- de gebruiker van de pedelec
- de gebruiker van de speed pedelec
- een autobestuurder

Wederom wordt hier niet te diep op ingegaan gezien de benodigde hoeveelheid data niet beschikbaar is om conclusies te trekken. In Figuur 9.4 zijn de snelheden te zien van 3 voertuigen. De pedelec, de speed pedelec en een autobestuurder. De snelheden werden uitgezet in een tabel. De stilstanden zijn er vantussen gehaald. Deze zijn te afhankelijk van rit tot rit en vertellen in deze context niet meer. De andere snelheden zijn opgedeeld in bereiken van 5 km/u en gedeeld door het aantal meetpunten exclusief stilstand. Het resultaat geeft het percentage van beweging aan een bepaalde snelheidbereik is gereden.

Het is duidelijk dat de pedelec en speedpedelec pieken hebben bij de snelheden waarbij ze geassisteerd worden. De pedelec-gebruiker ging wel nog vaker over de 25 km/u die geassisteerd werd. De speedpedelec heeft dit minder. De range die over de 45 km/u gaat kan een fout zijn in de RMSE , maar evengoed wind mee of helling af zijn. De auto tot slot heeft geen uitgesproken pieken, maar de snelheidslimieten 30, 50, 70 en 120 km/u zijn wel te herkennen. Tussen 30 en 50 km/u zijn de pieken minder uitgesproken. Dit kan zijn omdat er te snel gereden werd in zone 30 of onder de snelheid bij zone 50.



Figuur 9.4: snelheden bij verschillende voertuigen

# **Hoofdstuk 10**

## **Algemeen besluit**

*Hoe werkt een GPS?* Bij aanvang van dit onderzoek werd naar de werking van de GPS gekeken. Dit leerde ons dat er externe parameters zijn die de gebruiker niet onder controle heeft zoals metereologische of omgeving. Andere parameters als roaming om de posities van satellieten sneller te vinden en het zo min mogelijk bedekken van de GPS ontvanger. Het werd ook duidelijk dat er een verschil was in nauwkeurigheid tussen horizontale en verticale positiebepaling. Deze was meer dan het dubbele van de horizontale wat op 3 m verondersteld werd.

*Kiezen van de app* De keuze van app's werd op voorhand bepaald a.d.h.v. parameters als kostprijs, gebruiksvriendelijkheid en compatibiliteit met besturingssystemen van smartphones. Slecht enkele op de markt voldeden hieraan en gaven hun ruwe data ter beschikking. Het bleek ook dat geen apps voldeden aan deze criteria en werkten op alle besturingssystemen voor smartphones. De Windows phone bleek los te staan van Android en iOS.

*Proeven met apps op smartphones* Bij de metingen werd gekozen om de apps te vergelijken met een meetfiets die was uitgerust met een sensor aan het wiel. Er werd telkens hetzelfde traject gefietst. Maar ook dit weerhield niet dat de omgeving wijzigde (snoeien bomen, werken onderweg waardoor uitgeweken moest worde, etc.)

*Verwerkingswijze van de ruwe data* De verwerkingswijze van de ruwe data werd vervolgens onder bestudeerd. De weergave van de apps bleek onderling te verwisselen. Maar bij de meeste degelijke apps was het enkel de kolom die wijzigde. Zij gaven de coördinaten in latitude en longitude in decimale graden, het tijdstip van de meting met een datum en een hoogte. De afstand zelf werd zelden gegeven. Dus moest er op zoek gegaan worden naar een methode om de coördinaten om te zetten in afgelegde afstand. Het bleek dat kaartprojecties geen goede optie was. Beter was het te werken met afstanden op een bol. Er waren 3 methodes hiervoor. De cosinusregel, haversine-formule en Vincenty-formule. Uit deze 3 bleek de Vincenty-formule de nauwkeurigste en ook degene die Map my Ride gebruikte voor zijn afstandberekening.

*Testen met apps en meetfiets* Om te weten met welke apps de proeven gedaan gingen worden moesten ze onderling vergeleken worden. Aan de hand van enkele proefjes die ook inzicht gaven over de werking van de apps zelf werden 2 apps gekozen. Dit waren Runkeeper en Cycle Tracks GPS. De voornaamste reden was dat ze een gemiddelde snelheid hadden die dicht bij de meetfiets lag en een grilligheid in de curve waarmee gewerkt kon worden. Andere app's vertoonden veel fouten bij deze parameters. Uit de proeven bleek zoals gezegd enkele eigenschappen van de apps.

- Het beginpunten hadden een grotere onzekerheid dan deze eens er bewogen werd. Dit kwam doordat de precise locatie nog niet direct door de app nauwkeurig bepaald werd. Er werd eerst een snelle schatting gedaan van het gebied waar de gebruiker zich in bevond.
- De samptijd is soms afhankelijk van de besturingssysteem.
- Het hoogteverschil werd niet door de app zelf bepaald maar door een ander programma die dat achteraf deed op basis van de coördinaten. Dit bleek nog steeds onnauwkeurig. Er werd besloten in het vlakke Vlaamse landschap dit achterwege te laten.
- De gemiddelde snelheid van de apps is correcter dan de ogenblikkelijke snelheid.
- Bij lage snelheden (<5,4 km/u) werd de samptijd verhoogd. Dit was het geval tot 3 s bij snelheden onder 10,4 km/u.

Er zijn heel veel parameters die zorgen dat het moeilijk was een nauwkeurigheid te bepalen op de gemeten data. Deze werd achterhaald bij de apps die uit het gros naar voren waren gekomen.

*Smoothers* De data vertoonde unrealistische versnellingen en vertragingen. Deze uitschieters waren punten die weggewerkt konden worden mbv. smoothers. Er werd naar 3 smoothers gekeken. De lokaal gemiddelde smoother, de Gauss-kernel smoother en de Kalmanfilter. De lokaal gemiddelde smoother was een slecht systeem en bij de Gauss-kernel kon geen constant optimum bepaald worden. Dit verschilde veel van rit tot rit. De Kalman-filter wordt het meest gebruikt in de context van GPS'en. De onzekerheid op de gemeten data werd bepaald door een maximaal mogelijke versnelling. Deze moet experimenteel bepaald worden. Na onderzoek bleek een snelheid van  $2,7 \text{ m/s}^2$  de beste optie te zijn voor zowel de ctg als rk app. Een verbetering van 1 km/u werd waargenomen bij de RMSE die bij de ruwe data gemiddeld rond 4,4 m lag.

Om de fouten door verschuiving/laattijdige meting te elimineren werd bekeken of er verbeteringen zichtbaar waren bij histogrammen. Het probleem bij de histogrammen lag dat het lag aan de bereiken. Met de  $\chi^2$  waarde werd wel een verbetering waargenomen, maar deze gaf geen duidelijker beeld van een meer optimale versneling om in de Kalman-filter te steken.

*Snelheidszones* Aan de hand van coördinaten werd gekeken of mensen zich in zone 30 bevonden. Deze methode bleek volgens het PIP-systeem te bepalen. Maar er waren enkele nadelen aan de werkwijze om dit te kunnen bepalen.

- Er was niet 1 groot databestand voor Vlaanderen beschikbaar met de snelheidszones.
- Langdurig proces om snelheidszones te achterhalen bij de gemeenten.
- Omslachtig proces om coördinaten in excelbestanden te steken.
- Het traject van de gebruiker moet gekend zijn en liefst ook gelijk bij meerdere ritten.

*Verwerken van ritten van speed pedelec gebruikers* Tot slot werd bepaald dat in 7 stappen van de app installeren tot verwerkte data kan geraken. De snelheden van enkele speed pedelec ritten werd bekeken waaruit bleek dat de snelheid meestal rond de 40 km/u lag. Ook dat er kon vastgesteld worden of een gebruiker zich in zone 30 zich aan de snelheid hield. Het type voertuig kon vastgesteld worden aan de hand van histogrammen. De auto, speed pedelec en pedelec waren duidelijk van elkaar te onderscheiden.

## **Bijlage A**

### **Keuze apps**

Naam	Prijs	Extra betalende features	Android	Iphone	Bluetooth	Helling	Cadence	Hartslag	Grootte [MB]	Beoordeling	Data downloadbaar
Wahoo Fitness	€ -	Nee	Ja	Ja	Nee	Ja		Ja		3,90	?
Cyclemeter	€ -	Ja	Nee	Ja	Ja	Ja	?			4,00	Ja
Google Maps	€ -	Nee	Ja	Ja	Nee	Nee	Nee			4,30	Nee
Map My Ride	€ -	Ja	Ja	Ja	Ja	Ja	Ja	Ja		4,40	?
Strava	€ -	Ja	Ja	Ja	Ja	Ja	?	Ja	26,45	4,60	?
bicycle buddy	€ -	Nee	Ja	Ja	Nee	Nee	Nee	Nee		3,00	Nee
Positive Drive	€ -	Nee	Ja	Ja	Nee	Nee	Nee	Nee		3,10	Nee
Fietstijden.nl	€ -	Ja	Ja	Ja	Nee	Nee	Nee	Ja		4,30	?
Runtastic road bike tracker	€ -	Ja	Ja	Ja	Ja	Ja	?	Ja	17,79	4,50	?
Endomondo Hardlopen & Fiets	€ -	Ja	Ja	Ja	Ja	Nee	?	Ja		4,40	?
BikeComputer	€ -	Nee	Ja	Nee	Ja	Nee	Nee	Nee		4,30	?
Runtastic mountain bike	€ -	Ja	Ja	Ja	Ja	Ja	Ja	Ja	13,42	4,50	?
Bike gear Calculator	€ -	Nee	Ja	Ja	Nee	Nee	Nee	Ja		4,10	Nee
Cycle Tracker Pro	€ 2,99	Nee	Nee	Ja	Nee	Ja	Nee	Ja		3,00	Ja
Ride with gps	€ -	Ja	Ja	Ja	Ja	Ja	Ja	Ja		4,10	Ja
My Tracks	€ -	Nee	Ja	Ja	Nee	Ja	Nee	Nee		4,20	Ja
exclo Gps Fietsen fiest	€ -	Nee	Ja	Ja	Nee	Nee	Nee	Nee		3,90	Nee
Move! Bike computer	€ -	Nee	Ja	Ja	Nee	Ja	Nee	Nee	2,22	4,10	Ja
B.iCycle - GPS-fietscomputer	€ 3,99	Nee	Ja	Ja	Nee	Ja	Nee	Nee		3,40	Nee

Figuur A.1: Lijst van 20 apps

## **Bijlage B**

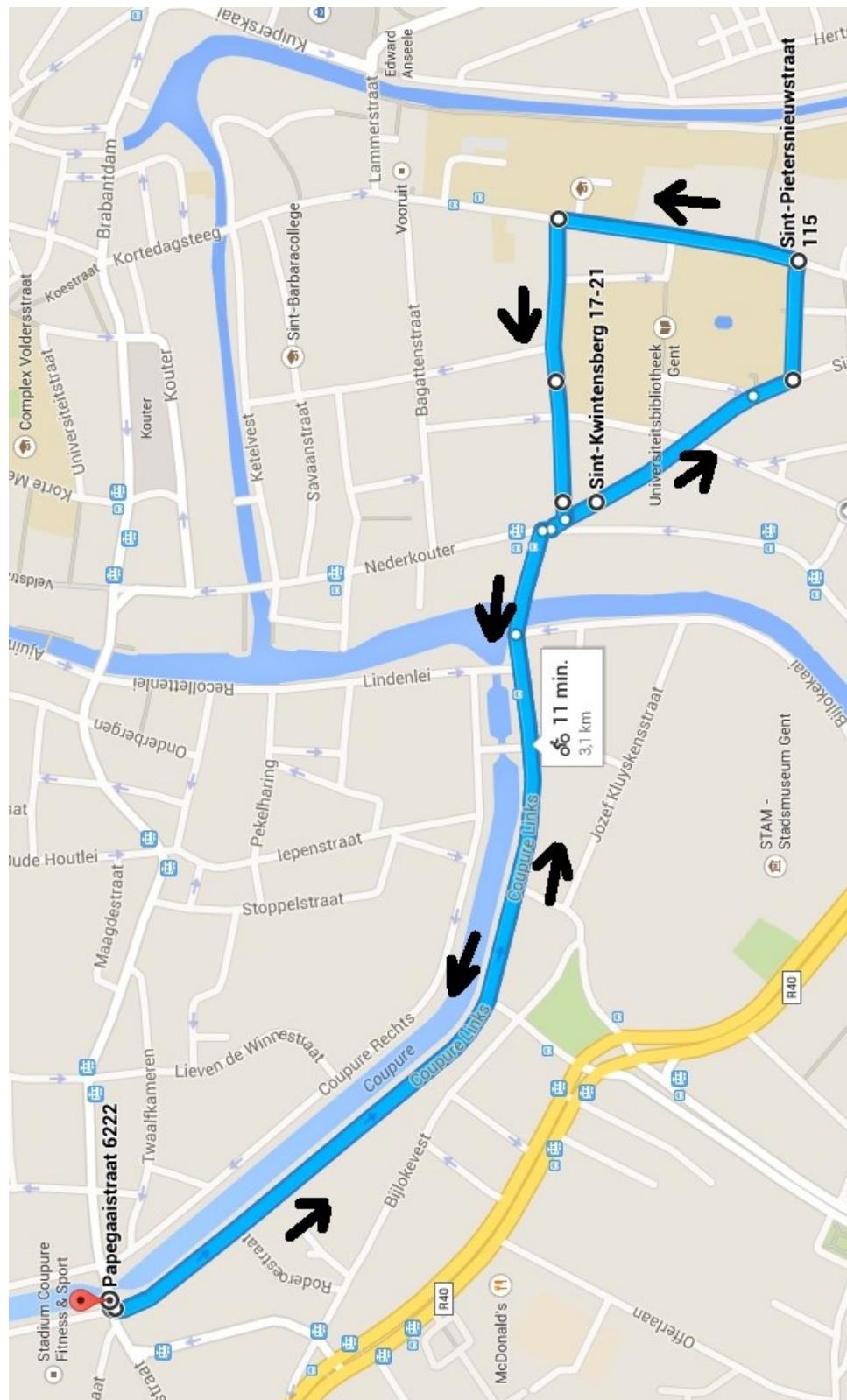
### **Bronnen apps**

Naam	Link
Wahoo Fitness	<a href="https://play.google.com/store/apps/details?id=com.wahoofitness.fitness">https://play.google.com/store/apps/details?id=com.wahoofitness.fitness</a>
Cyclemeter	<a href="https://play.google.com/store/apps/details?id=com.mapmyride.android">https://play.google.com/store/apps/details?id=com.mapmyride.android</a>
Google Maps	<a href="https://play.google.com/store/apps/details?id=com.google.android.apps.maps&amp;hl=en">https://play.google.com/store/apps/details?id=com.google.android.apps.maps&amp;hl=en</a>
Map My Ride	<a href="https://play.google.com/store/apps/details?id=com.mapmyride.android">https://play.google.com/store/apps/details?id=com.mapmyride.android</a>
Strava	<a href="https://play.google.com/store/apps/details?id=com.strava">https://play.google.com/store/apps/details?id=com.strava</a>
bicycle buddy	<a href="https://play.google.com/store/apps/details?id=nl.luminis.bicyclebuddy">https://play.google.com/store/apps/details?id=nl.luminis.bicyclebuddy</a>
Positive Drive	<a href="https://play.google.com/store/apps/details?id=com.pd&amp;hl=nl">https://play.google.com/store/apps/details?id=com.pd&amp;hl=nl</a>
Fietstijden.nl	<a href="https://play.google.com/store/apps/details?id=nl.fietstijden.android">https://play.google.com/store/apps/details?id=nl.fietstijden.android</a>
Runtastic road bike tracker	<a href="https://play.google.com/store/apps/details?id=com.runtastic.android.roadbike.lite">https://play.google.com/store/apps/details?id=com.runtastic.android.roadbike.lite</a>
Endomondo Hardlopen & Fietsten	<a href="https://play.google.com/store/apps/details?id=com.endomondo.android">https://play.google.com/store/apps/details?id=com.endomondo.android</a>
BikeComputer	<a href="https://play.google.com/store/apps/details?id=de.roehrlerbikecomputer">https://play.google.com/store/apps/details?id=de.roehrlerbikecomputer</a>
Runtastic mountain bike	<a href="https://play.google.com/store/apps/details?id=com.runtastic.android.mountainbike.lite">https://play.google.com/store/apps/details?id=com.runtastic.android.mountainbike.lite</a>
Bike gear Calculator	<a href="https://play.google.com/store/apps/details?id=com.gearcalculator&amp;hl=nl_BE">https://play.google.com/store/apps/details?id=com.gearcalculator&amp;hl=nl_BE</a>
Cycle Tracker Pro	<a href="https://play.google.com/store/apps/details?id=com.ridewithgps.mobile&amp;utm_campaign=android_app&amp;utm_source=site&amp;utm_medium=android_page">https://play.google.com/store/apps/details?id=com.ridewithgps.mobile&amp;utm_campaign=android_app&amp;utm_source=site&amp;utm_medium=android_page</a>
Ride with gps	<a href="https://play.google.com/store/apps/details?id=com.google.android.maps.mytracks">https://play.google.com/store/apps/details?id=com.google.android.maps.mytracks</a>
My Tracks	<a href="https://play.google.com/store/apps/details?id=com.lufful.exclo">https://play.google.com/store/apps/details?id=com.lufful.exclo</a>
exclo GPS Fietsen fiest	<a href="https://play.google.com/store/apps/details?id=pl.com.digita.BikeComputer">https://play.google.com/store/apps/details?id=pl.com.digita.BikeComputer</a>
Move Bike computer	<a href="https://play.google.com/store/apps/details?id=com.biycle.android">https://play.google.com/store/apps/details?id=com.biycle.android</a>
B.iCycle - GPS-fietscomputer	<a href="https://play.google.com/store/apps/details?id=com.biycle.android">https://play.google.com/store/apps/details?id=com.biycle.android</a>

Figuur B.1: Bronnen van de 20 pregesteekteerde apps

## **Bijlage C**

### **Testparcour**



Figuur C.1: Parcours afgiefst voor de testen

## **Bijlage D**

# **Stappenplan voor het gebruiken van de apps**

Hieronder worden de stappen besproken die nodig zijn om van installatie van de app tot het bekomen van de rauwe data die bewerkbaar is in Scilab.

### **Installatie van de app :**

1. Ga naar de appstore op de gsm.
2. Type de naam van de app in en ga na of er meerdere apps bestaan onder dezelfde naam.  
Kies in dat geval voor de optie voor fietsers'
3. Na de installatie is de app gebruiksklaar
4. Maak een profiel aan voor de testen van start gaan. Dit profiel zal toegang geven tot het online downloaden van de ruwe data.
5. Ga na of de eenheden op de app ingesteld zijn op metrische eenheden. Dit hoeft slechts een maal gedaan te worden.

### **De meting :**

1. Ga na of GPS / A-GPS aanstaat op de gsm om de app te gebruiken
2. Sta bij de aanvang van de meting enkele seconden stil wanneer de app van start is gegaan. Zo voorkom je een 'cold start'.
3. Zorg dat de gsm optimaal bereik heeft door deze nergens in te steken.
4. Bij het beëindigen van de app sta dan wederom enkele seconden stil om een 'cold stop' te voorkomen.
5. Zorg dat na elke meting de app de meting opslaat.
6. Geef elke app voor de zekerheid een comment zodat ze te bij onduidelijkheid op de site toch te vinden zijn. Anders zal op basis van tijd of afgelegde parcours de meting herkend moeten worden.

**Het downloaden van de data :**

1. Ga naar de site van de gekozen app en log in met het eerder gekozen profiel<sup>1</sup>.
2. elke site heeft zijn eigen leuke zoektocht naar het .gpx bestand wanneer ze dit hebben. Wanneer dit niet direct vindbaar is kijk dan online voor hulp. Wanneer er geen .gpx beschikbaar is zijn er ook nog andere versies van bestanden die mogelijk omzetbaar zijn naar Excel. Dit is een vereiste bij de volgende stap.

**Omzetten van de data zodat deze door Scilab gebruikt kan worden :**

1. rechtermuisklik op het .gpx bestand en open het met Microsoft Excel.
2. Excel zal enkele foutmeldingen geven maar druk telkens op 'OK'
3. de data zal nu in kolommen zichtbaar zijn
4. sla dit document op onder een versie 'Excel 97-2003 Workbook'. Andere versies kan Scilab foutmeldingen geven voor onleesbaarheid bij het 'read xls' commando.
5. Geef een naam zonder spaties of tekens zodat bij het inlezen Scilab wederom geen foutmelding geeft. ' ' ipv een spatie werkt zonder problemen.

---

<sup>1</sup>sommige apps hebben geen site. In dat geval is het nodig om via de app de data te versturen naar een email adres na elke meting.

## Bijlage E

# Van app naar Scilab

De wijze waarop van .gpx bestand naar een verwerkt csv-file wordt hieronder in stappen uitgelegd.

Het is noodzakelijk in het bezit te zijn van het bestand *algemene\_filter.sce* en Scilab om het in te gebruiken<sup>1</sup>

### E.0.1 Omzetten van .gpx naar .xls

1. open het .gpx bestand in Excel
2. Er komen een paar pop-up schermen met vragen. Kort door de bocht komt het neer op 4x te enteren.
  - (a) klik bij de melding *the file you want to open is in a different format than specified by the file extension. Verify that the file is not corrupted and is from a trusted source before opening the file. Do you want to open the file now?* op 'Yes'
  - (b) Het volgende venster dat verschijnt vraagt onder welke vorm het bestand ge-opend moet worden. Vink 'As an XML-table' aan en klik op 'Ok'
  - (c) Het volgende scherm vraag : *The following schema elements and structures cannot be mapped to a worksheet... Do you want to continue adding this schema to your workbook?* Klik wederom op 'Yes'.
  - (d) Het volgende scherm geeft het XML Import Error. Deze misleidende titel geeft enkel weer dat de data geïmporteerd is. Druk voor de laatste keer op 'Ok'
3. Ga naar 'Save As' en wijzig het type van bestand van *Excel Workbook* naar *Excel 97 -2003 Workbook*. Dit type is bewerkbaar via Scilab.

Het originele gpx bestand kan nog gebruikt worden om het originele traject te visualiseren op Google maps via <http://www.GPSvisualizer.com/>.

---

<sup>1</sup>Scilab is gratis wiskundig programma dat te downloaden valt via het internet

### E.0.2 Openen en bewerken .xls bestand in Scilab

1. Open Scilab
2. Wijzig de directory naar de map waarin het bestand ‘*algemene\_filter.sce*’ staat
3. Typ `exec('algemene_filter.sce')`
4. Er verschijnt een box waarin je het bestand dat je nodig hebt gaat aanduiden.
5. Het programma zal het xls bestand inlezen, op basis van de coordinaten een afstand geven. Een verstreken tijd berekenen. Een snelheid bepalen en een kalmanfilter gebruiken om het ruis er uit te halen. Dit allemaal zal gezet worden in een CSV-bestand dat automatisch onder dezelfde naam als het bestand in dezelfde map geschreven wordt. Hier staat in de kolommen respectievelijk : verstreken seconden, afgelegde afstand, snelheid op tijdstip t , nauwkeurigheid van snelheid, versnelling, latitude op tijdstip t , longitude op tijdstip t .

## **Bijlage F**

### **De cosinus-regel**

## **Bijlage G**

### **De Haversine formule**

## Bijlage H

# Formule van Vincenty

Er zijn verschillende berekeningsmethodes die op niet iteratieve wijze afstanden bepalen. Maar deze vragen meer rekenwerk en hebben meer verwerkingstijd nodig om via computer te berkekenen. De reden waarom deze berekeningswijze sneller zijn heeft te maken met het feit dat ze slecht gebruik maakt van 3 trigonometrische functies. De sinus, cosinus en tangens. Dit zorgt dat er minder data hoeft opgeslagen en opgeroepen worden vanuit het programma.

Zoals reeds vermeld in de tekst zijn er 2 wijzen waarop de Vincenty formule kan gebruikt worden. De directe en indirecte Vincenty formule. notaties :

- a,b de grote en kleine semeassen van de ellipsoïde.
- f, de excentriciteit =  $a(a - b)/a$
- L, het verschil in longitude (positief oost)
- s, lengte van de geodese
- $\alpha_1, \alpha_2$  de azimuth van de geodese, wijzerzin van het noorden.
- $\alpha_2$  azimuth van de geodese bij de evenaar.
- $u^2 = \cos^2 \alpha (a^2 - b^2) / b^2$
- U, gereduceerde latitude, gedefinieerd door  $\tan U = (1 - f) * \tan \phi$
- $\lambda$ , verschil in longitude op de hulpsfeer. Deze wordt geïttereerd waarbij als eerste waarde L wordt genomen.
- $\sigma$  de gehoekte afstand tussen P1 en P2 op de sfeer
- $\sigma_1$  de gehoekte afstand op de sfeer van de evenaar tot P1
- $\sigma_m$  gehoekte afstand op de sfeer van de evenaar tot het middelpunt van de lijn.

Directe formule :

$$\tan \sigma_1 = \tan U_1 / \cos \alpha_1 \quad (\text{H.1})$$

$$\sin \alpha = \cos U_1 \sin \alpha_1 \quad (\text{H.2})$$

$$A = 1 + \frac{u^2}{16384} \{ 4096 + u^2[-768 + u^2(320 - 175u^2)] \} \quad (\text{H.3})$$

$$B = \frac{u^2}{1024} \{ 256 + u^2[-128 + u^2(74 - 47u^2)] \} \quad (\text{H.4})$$

$$2\sigma_m = 2\sigma_1 + \sigma \quad (\text{H.5})$$

$$\Delta\sigma = B \sin \sigma \left\{ \cos 2\sigma_m + \frac{1}{4} B [\cos \sigma (-1 + 2 \cos^2 2\sigma_m) - \frac{1}{6} B \cos 2\sigma_m (-3 + 4 \sin^2 \sigma) (-3 + 4 \cos^2 2\sigma_m)] \right\} \quad (\text{H.6})$$

$$\sigma = \frac{s}{bA + \Delta\sigma} \quad (\text{H.7})$$

In vgln (H.5), (H.6) & (H.7) worden geïttereerd totdat het verschil in  $\sigma$  verwaarloosbaar is.

$$\tan \Phi_2 = \frac{\sin U_1 \cos \sigma + \cos U_1 \sin \sigma \cos \alpha_1}{(1-f)[\sin^2 \alpha + (\sin U_1 \sin \sigma - \cos U_1 \cos \sigma \cos \alpha_1)^2]^{\frac{1}{2}}} \quad (\text{H.8})$$

$$\tan \lambda = \frac{\sin \sigma \sin \alpha_1}{\cos U_1 \cos \sigma - \sin U_1 \sin \sigma \cos \alpha_1} \quad (\text{H.9})$$

$$C = \frac{f}{16} \cos^2 \alpha [4 + f(4 - 3 \cos \alpha_1)] \quad (\text{H.10})$$

$$L = \lambda - (1-C)f \sin \alpha \{ \sigma + C \sin \sigma [\cos 2\sigma_m + C \cos \sigma (-1 + 2 \cos^2 2\sigma_m)] \} \quad (\text{H.11})$$

$$\tan \alpha_2 = \frac{\sin \alpha}{-\sin U_1 \sin \sigma + \cos U_1 \cos \sigma \cos \alpha_1} \quad (\text{H.12})$$

## **Bijlage I**

### **Standaardafwijking**

## **Bijlage J**

### **Afwijking van afstand door bochten bij apps**

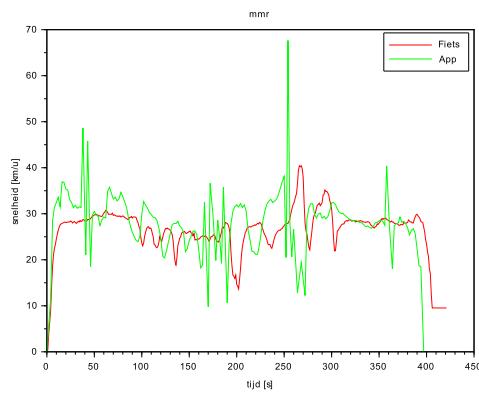
Dit is niet opgenomen onder de testen met apps en meetfiets omdat dit niet in kaart viel te brengen voor de vergelijking van apps onderling.

rechte hoek met 2 zijden van  $1m = 2\text{mafgelegd}$ . bij cirkel  $2 * \pi * 1/4 = 1,57\text{mafgelegd}$  worst case is een recht lijn tussen 2 punten= $\Rightarrow \sqrt{2} = 1,41m$

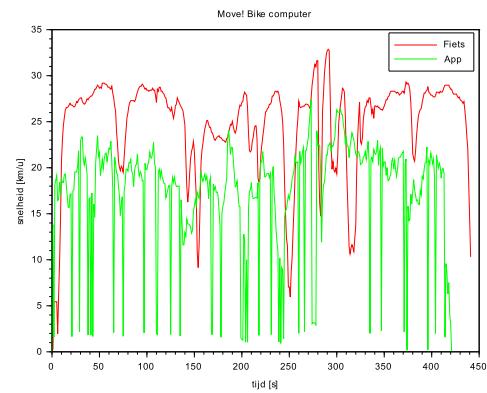
= $\zeta$  worst case onvg 30% van afstand minder afgelegd.

## Bijlage K

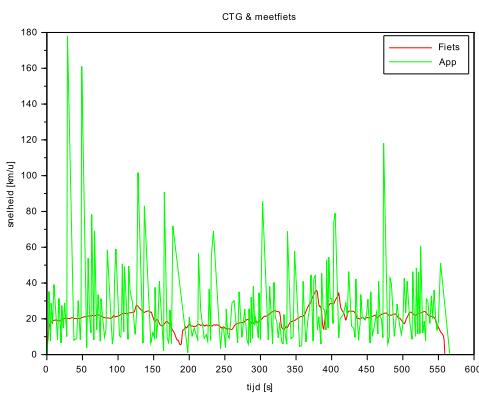
### Voorbeelden ritten apps



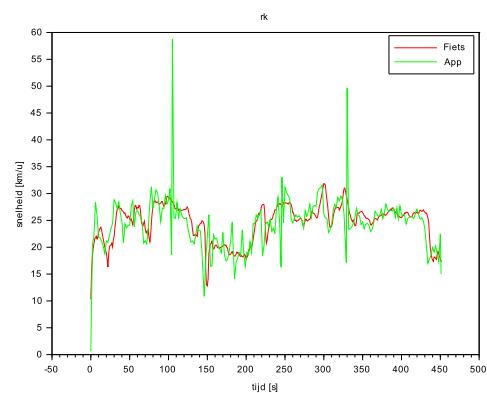
(a) Voorbeeld rit mmr



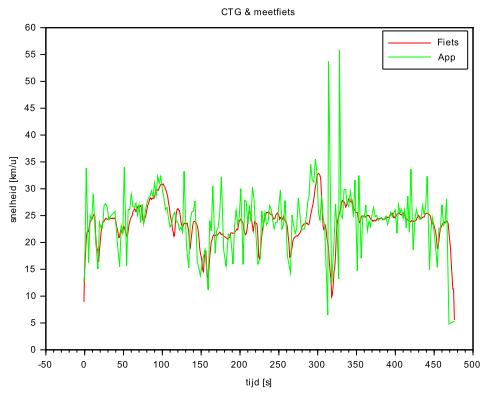
(b) Voorbeeld rit mbc



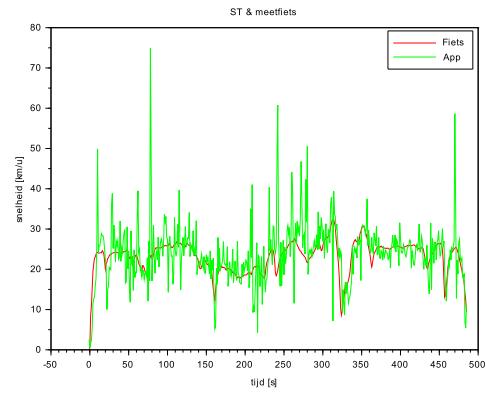
(a) Voorbeeld rit ct



(b) Voorbeeld rit rk



(a) Voorbeeld rit ctg



(b) Voorbeeld rit st

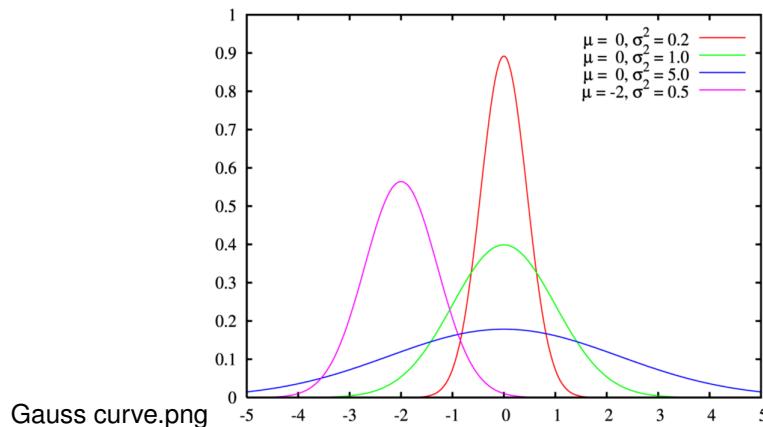
## Bijlage L

# Gauss Curve

De curve van Gauss wordt ook wel een normale verdeling genoemd. Het is een kansverdeling die gegeven wordt door de formule (L.1)

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2} \quad (\text{L.1})$$

De e-macht zorgt voor de curve die zo welbekend is. De term  $\sigma$  staat voor de standaardafwijking en  $\mu$  de verwachtingswaarde. De eerste term  $\frac{1}{\sigma\sqrt{2\pi}}$  is de normalisatieconstante. Deze zorgt dat de oppervlakte onder de curve steeds gelijk is aan 1. Hierdoor krijgt elk oppervlaktedeel een percentage van 1 die zijn kans weergeeft bij een normale verdeling. Dit valt eenvoudig te controleren door de integraal van  $-\infty$  tot  $\infty$  en men krijgt  $\sigma\sqrt{2\pi}$ .



**Figuur L.1:** Standaard Gauss curves

Wat wil dat concreet zeggen?  $\mu$  is de x-waarde waar de curve symmetrisch kan rond gespiegeld worden. Bij een verlegging van  $\mu$  verschuift het middenpunt. De verwachtingswaarde is de waarde die het meeste kans heeft op de curve waardoor deze onder het hoogste punt ligt.  $\sigma$  is de standaardafwijking. Bij een normale verdeling liggen 68,26% van de waarden tussen  $-\sigma$  en  $\sigma$ . 95,44% ligt tussen  $-2\sigma$  en  $2\sigma$ . Hoe groter  $\sigma$  hoe breder en lager de curve zal zijn zoals te zien is in Figuur L.1

## **Bijlage M**

### **Stelling van Jordan**

Kijk eerst naar raytracing voor uitleg. Probleemstellingen die hieronder besproken ook aan bod brengen. <http://alienryderflex.com/polygon/>

## **Bijlage N**

### **Script voor het plotten van de polygonen**

## Bijlage O

### Scilab script ‘Algemene filter’ in detail

Hier wordt de code weergegeven die gebruikt is voor de verwerking van ruwe data naar het snelheidsprofiel. Bij elke stap wordt een Figuur gegeven die verklaard wordt.

```
clear  
clc  
tic()
```

Het *clear* en *clc* commando maken het scherm en geheugen met parameters die gebruikt waren vrij. *tic()* start een teller. Dit staat er enkel om een richtlijn te geven hoe lang het programma moet lopen op het eind van het document. Zo kan een inschatting gemaakt worden voor de duur bij de ander bestanden.

```
bestandsnaam=uigetfile(["*.xls"], "", "Kies het excelbestand van een aangelegd fietstraject");  
Sheets=readxls(bestandsnaam);  
[path, fname, extension] = fileparts(bestandsnaam);  
s3=Sheets(1);
```

```

//1 = Runkeeper
//2 = Strava
//3 = Cycle Tracks GPS
//4 = Map my ride
app_keuze=x_choose(['Runkeeper','Strava','Cycle Tracks GPS','Map My Ride'],[Dubbel klik op de app;'die gebruikt werd voor deze meting:']);
select app_keuze
case 1 then
    lat=85;
    long=86;
    tijd=88;
case 2 then
    lat=85;
    long=86;
    tijd=88;
case 3 then
    lat=6;
    long=7;
    tijd=9;
case 4 then
    lat=9;
    long=10;
    tijd=8;
end

[nijen,kolommen]=size(s3);
app=zeros(nijen-2,5);

app(:,2)=s3(3:$,lat); //lat
app(:,3)=s3(3:$,long); //long

i=4;
t2=[];
t1=[strtod(part(s3(2,tijd),1:4)),strtod(part(s3(2,tijd),6:7)),strtod(part(s3(2,tijd),9:10)),strtod(part(s3(2,tijd),12:13)),strtod(part(s3(2,tijd),15:16)),strtod(part(s3(2,tijd),18:19))];

while(i<=(nijen))
t2=[strtod(part(s3(i,tijd),1:4)),strtod(part(s3(i,tijd),6:7)),strtod(part(s3(i,tijd),9:10)),strtod(part(s3(i,tijd),12:13)),strtod(part(s3(i,tijd),15:16)),strtod(part(s3(i,tijd),18:19))];
app(i-1,1)=etime(t2,t1);
i=i+1;
end

```

```
i=2;
j=1;
rijen_te_verwijderen=0;
while(i<=rijen)
    if(app(i-1,1)==app(i,1))then
        rijen_te_verwijderen(j,1)=i;
        j=j+1;
    end
    i=i+1;
end

app(rijen_te_verwijderen-1,:)=(app(rijen_te_verwijderen-1,:)+app(rijen_te_verwijderen,:))/2;
app(rijen_te_verwijderen,:)=[];
[njen,kolommen]=size(app);
```

```

a=6378137.0;
b=6356752.3;
f=(a-b)/a;
i=2;
while(i<=(njen-1))
    phi1=(app(i-1,2)/180*%pi); //lat1
    y1=(app(i-1,3)/180*%pi); //long 1
    phi2=(app(i,2)/180*%pi); //lat2
    y2=(app(i,3)/180*%pi); //long 2

    if(abs(y2-y1)~=0)then
        L=y2-y1;
    else
        L=10^(-6);
    end
    U1=atan((1-f)*tan(phi1));
    U2=atan((1-f)*tan(phi2));
    lambda=L;
    sigma=0;
    cosalfakwadraat=0;
    costweesigmam=0;
    sinalfa=0;
    C=0;
    cossigma=0;
    sinsigma=0;
    n=1;
    while(n<100)
        sinsigma=sqrt((cos(U2)*sin(lambda))^2+(cos(U1)*sin(U2)-sin(U1)*cos(U2)*cos(lambda))^2);
        cossigma=sin(U1)*sin(U2)+cos(U1)*cos(U2)*cos(lambda);
        sigma=atan(sinsigma/cossigma);
        sinalfa=cos(U1)*cos(U2)*sin(lambda)/sinsigma;
        cosalfakwadraat=1-sinalfa^2;
        costweesigmam=cossigma-2*sin(U1)*sin(U2)/cosalfakwadraat;
        C=f/16*cosalfakwadraat*(4+f*(4-3*cosalfakwadraat));
        lambda=L+(1-C)*f*sinalfa*(sigma+C*sinsigma*(costweesigmam+C*cossigma*(-1+2*costweesigmam^2)));
        n=n+1;
    end
    ukwadraat=cosalfakwadraat*(a^2-b^2)/(b^2);
    A=1+ukwadraat/16384*(4096+ukwadraat*(-768+ukwadraat*(320-175*ukwadraat)));
    B=ukwadraat/1024*(256+ukwadraat*(-128+ukwadraat*(74-47*ukwadraat)));
    deltasigma=B*sinsigma*(costweesigmam+B/4*(cossigma*(-1+2*costweesigmam^2)-B/6*costweesigmam*(-3+4*sinsigma^2)*(-3+4*costweesigmam^2)));
    afstand=b*A*(sigma-deltasigma);
    app(i,4)=afstand;
    i=i+1;
end

i=2;
while(i<=njen-1)
    app(i,4)=app(i-1,4)+app(i,4);
    i=i+1;
end

```

```
i=2;
while(i<=(njen-1))
  if(app(i,1)~=app(i-1,1))then
    app(i,5)=(app(i,4)-app(i-1,4))*3.6/((app(i,1)-app(i-1,1)));
  else app(i,5)=app(i-1,5);
  end
i=i+1;
end
```

```
tesnel=find(app(:,5)>65);
app(tesnel,:)=[];
[njen,kolommen]=size(app);
```

```
totsec=ceil(app($,1));
t_orig_app=app(:,1);
v_orig_app=app(:,5);
t_intp=linspace(0,totsec,(totsec+1));
v_intp=interp1(t_orig_app,v_orig_app,t_intp,'linear');
lat_intp=interp1(t_orig_app,app(:,2),t_intp,'linear');
long_intp=interp1(t_orig_app,app(:,3),t_intp,'linear');
```

```
v_intp(isnan(v_intp))=0;
d_intp=v_intp*3.6;

i=2;
while(i<=length(d_intp))
  d_intp(i,:)=d_intp(i-1,:)+d_intp(i,:);
  i=i+1;
end

i=2;
a_intp=zeros(length(v_intp),1);
while(i<=(length(v_intp)-1))
  a_intp(i)=(v_intp(i)-v_intp(i-1))/(t_intp(i)-t_intp(i-1));
  i=i+1;
end
```

```

i=3;
dt=t_intp(i+1,1)-t_intp(i,1);
onzekerheid=3;
X0=[0;0];
P0 = [onzekerheid^2 0; 0 0];
F=[1 dt; 0 1];
H = [1,0];
R=[onzekerheid^2];
Q=[dt^4/4 dt^3/2; dt^3/2 dt^2]*2;
I=[1,0;0,1];
C=I;
C=C';
i=1;

Xkv=X0;
Pkv=P0;

while(i<=length(d_intp))

Xkp=F*Xkv;
Pkp = F*Pkv*F + Q;
y=d_intp(i,1);
Yk=C*y;

K= Pkp * H'/ (H*Pkp*H'+ R);
X= Xkp+ K*(Yk - H*Xkp);

Pk=(I-K*H)*Pkp;
//kalmanresultaat=cat(2,kalmanresultaat,X);
kalmanresultaat(:,i)=[X,Pk(1,1);Pk(2,2)];
Xkv=X;
Pk=Pk;
i=i+1;
end

kalmanresultaat=kalmanresultaat';
kalmanresultaat(:,2)=kalmanresultaat(:,2)*3.6;

i=2;
while(i<=length(t_intp))
    kalmanresultaat(i,5)=(kalmanresultaat(i,2)-kalmanresultaat(i-1,2))/(t_intp(i)-t_intp(i-1));
    i=i+1;
end

eindresultaat=cat(2,t_intp,kalmanresultaat,lat_intp,long_intp);

write_csv(eindresultaat,path+fname+'.csv');

```

```
clf(2);
scf(2);
plot(t_intp,kalmarresultaat(:,2),b)
plot(t_intp(:,1),v_intp(:,1),-g)
plot(mean(kalmarresultaat(:,2)))
title("Kalmanfilter");
ylabel("snelheid [km/u]");
xlabel("tijd[s]");
legend(["kalmanfilter","niet-gecorrigeerde snelheid"]);
```

```
beep
toc()
```

## **Bijlage P**

### **Scilab script ‘Herkennen snelheidszones’ in detail**



## Bijlage Q

# Poster

KU LEUVEN

# De veiligheid van fietsers op basis van hun snelheidsprofielen

De alledaagse fietser kan tegenwoordig hogere snelheden dan ooit behalen

Snelheid km/u	Percentage kans op een ongeval (%)
0	~10
20	~20
40	~45 (Dodeboting)
45	~50
60	~65 (Verguisboting)
65	~70
70	~75
80	~80

met behulp van een elektrische fiets. De snelheden lopen op tot 45 km/u.

Bestaat manier basis gemeten snelheden te leggen voor veiligheid?

er een om op van een link met Op basis

van gps-data wordt gepoogd een

snelheidsprofiel op te stellen van fietsers en hiermee uitspraken te doen over veiligheid van de nieuwe snelle fietser.

Afstand fm	Garmin (km/h)	Wiko (km/h)	iPhone (km/h)
0	~25	~25	~25
200	~25	~25	~25
300	~25	~25	~10 (peak)
400	~25	~25	~45 (peak)
500	~25	~25	~25
600	~25	~35	~25
700	~25	~30	~25
800	~25	~35	~25
900	~25	~30	~25
1000	~25	~40	~25

door : Olivier Burez

Promotoren : Emilia Motoasca & Bram Rotthier

FACULTEIT INDUSTRIELE INGENIEURSWETENSCHAPPEN  
TECHNOLOGIECAMPUS GENT  
Gebroeders De Smetstraat 1  
9000 GENT, België  
tel. + 32 92 65 86 10  
fax + 32 92 25 62 69  
[iiw.gent@kuleuven.be](mailto:iiw.gent@kuleuven.be)  
[www.iiw.kuleuven.be](http://www.iiw.kuleuven.be)



LID VAN  
**ASSOCIATIE  
KU LEUVEN**