DE HAAGSE HOGESCHOOL

Slagdetectie bij het schaatsen

Machiel Smits
Opleiding Bewegingstechnologie, Haagse Hogeschool

J. Koopman
1e Begeleider

H. Faber
2e Begeleider

R. Mulder
Externe begeleider

Juni 2015

Inhoudsopgave

1. Samenvatting

Uit onderzoek is gebleken dat de verschillende snelheden bij topschaatsers vooral tot stand komen door het aanpassen van de slagfrequentie. Men is geïnteresseerd in de slagfrequentie bij het starten en het verschil in slagfrequentie tussen de bochten en de rechte stukken evenals de relatie tussen de slagfrequentie en de rondetijden. De Vrije Universiteit Amsterdam is begonnen met het detecteren van schaatsslagen aan de hand van sensoren op de schaats. De mogelijkheden van het programma waren beperkt. Aan de hand van de vraag uit de schaatssport en de beperkingen van het programma was de volgende onderzoeksvraag opgesteld: Is het mogelijk om op basis van data uit gyroscopen of accelerometers een algoritme te schrijven dat zelfstandig schaatsslagen kan detecteren op het rechte stuk, in de bocht en tijdens het starten?
Het algoritme moet 98% van de slagen detecteren en slagen in de bochten en op de rechte stukken scheiden.
Er zijn in totaal vier algoritmes geschreven. Twee voor starts en twee voor het rijden van rondes. Beide hebben een versie gebaseerd op een rompsensor en een versie gebaseerd op schaatssensoren. Alle algoritmes zijn gebaseerd op piekdetectie en berekenen slagtijd, slagfrequentie en het aantal slagen. De algoritmes voor de rondes berekenen tevens de rondetijden. De algoritmes zijn geïmplementeerd in een GUI.
De vier algoritmes zijn eerst getest op een skeelerbaan. De algoritmes voor de rondes zijn later ook nog getest op een ijsbaan. De slagtijden en het aantal slagen zijn vergeleken met resultaten uit videoanalyse. De rondetijden zijn vergeleken met een stopwatch.
Voor de slagtijden is een correlatie berekend tussen de algoritmes en high-speed video en er zijn regressievergelijkingen opgesteld. Er zijn Bland-Altman plots gemaakt om de resultaten van de correlaties en de regressievergelijkingen te controleren.
Beide versies van het algoritme voor de rondes scoorden bij de skeelertest lager dan de gewenste 98%. De algoritmes, geschreven voor rondes, vertoonden lage correlaties met de high-speed video. De algoritmes, geschreven voor starts, vertoonden hoge correlaties met de high-speed video. De procentuele deviaties uit de Bland-Altman plots toonden aan dat de hoge correlaties onterecht zijn. De slagtijden van de algoritmes moeten nog een keer gevalideerd worden met schaatsmetingen om een betere conclusie te kunnen trekken.
Bij de schaatstest behaalde de rompsensor 99% en de schaatssensoren 96,3 %. Het percentage van de schaatssensoren zal zeer waarschijnlijk hoger worden als er een correctie wordt geschreven voor de overgang van het rechte stuk naar de bocht.

2. Inleiding

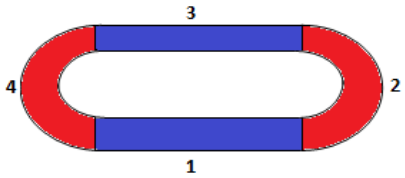
Schaatsen is een populaire sport in Nederland, met als gevolg dat er al jaren onderzoek wordt gedaan naar de schaatsbeweging. 'Real-time feedback voor een betere schaatsprestatie' is een project, gefinancierd door STW (Stichting voor de Technische Wetenschappen), dat loopt tot 1 september 2016. Wetenschappers van de Vrije Universiteit Amsterdam en de Technische Universiteit Delft werken in dit project samen met als doel het verbeteren van de individuele schaatstechniek door middel van real-time feedback over de afzet tijdens het schaatsen. Bij het langebaanschaatsen worden doorgaans zes afstanden gereden: 500m, 1000m, 1500m, 3000m (vrouwen), 5000m en 10000m (mannen). De snelheden die bereikt worden tijdens deze afstanden nemen toe naarmate de afstand afneemt en worden voor een groot deel bepaald door het geleverde vermogen van de schaatser. Het vermogen dat een schaatser levert is het product van de arbeid per slag en de slagfrequentie (Van Ingen Schenau & Bakker, 1980). Uit onderzoek bij vrouwelijke topschaatsters bleek dat de arbeid per slag een constante factor is en dat de verschillende snelheden op de afstanden vooral tot stand komen door het aanpassen van de slagfrequentie (Van Ingen Schenau, De Groot & De Boer, 1985). Het kan daarom nuttig zijn om de slagfrequentie van atleten te bepalen tijdens trainingen en wedstrijden. Schaatscoaches zijn geïnteresseerd in de slagfrequentie, met name voor het analyseren van de start. Daarnaast is men geïnteresseerd in het verschil in slagfrequentie tussen de bochten en de rechte stukken evenals de relatie tussen de slagfrequentie en de rondetijden.
De laatste jaren is het gebruik van micro-elektromechanische systemen (MEMS) sterk toegenomen. Met name accelerometers en gyroscopen worden in allerlei toepassingen gebruikt, zoals mobiele telefoons. In de gezondheidszorg worden accelerometers gebruikt om statische en dynamische activiteiten tijdens het dagelijks leven te monitoren (Veltink, Bussmann, De Vries, Martens & Van Lummel, 1996). Daarnaast zijn op basis van accelerometer data verscheidene algoritmes geschreven die stappen kunnen detecteren tijdens het lopen (Ying, Silex, Schnitzer, Leonhardt & Schiek, 2007). Accelerometers worden in de sport onder andere gebruikt voor het monitoren van atleten tijdens trainingen en wedstrijden (Neville, Wixted, Rowlands & James, 2010). In de schaatssport wordt echter nog geen gebruik gemaakt van MEMS.
De Vrije Universiteit Amsterdam is begonnen met het detecteren van schaatsslagen aan de hand van data uit een sensor, de naam van deze sensor is 'FBW Sensor'. FBW sensoren zijn zelf samengestelde MEMS van de Faculteit der Bewegingswetenschappen, bestaande uit een gyroscoop en een accelerometer. Deze sensoren worden bevestigd op de schaatsschoen of de onderrug. Er is een eenvoudige MATLAB GUI (Graphical User Interface) geschreven die slagen kan detecteren uit gyroscoop data aan de hand van piekdetectie. Uit de gedetecteerde slagen kan de slagtijd en slagfrequentie worden bepaald. Het programma is geschreven voor sensoren op de schaats en is alleen te gebruiken voor rondes. Er wordt geen onderscheid gemaakt tussen bochten en rechte stukken.
Aan de hand van de vraag uit de schaatssport en de beperkingen van het huidige programma is de volgende onderzoeksvraag opgesteld:
- Is het mogelijk om op basis van data uit gyroscopen of accelerometers een algoritme te schrijven dat zelfstandig schaatsslagen kan detecteren op het rechte stuk, in de bocht en tijdens het starten?

Het doel is om minimaal 98% van de slagen te detecteren.

3. Analyse

3.1 Literatuuronderzoek

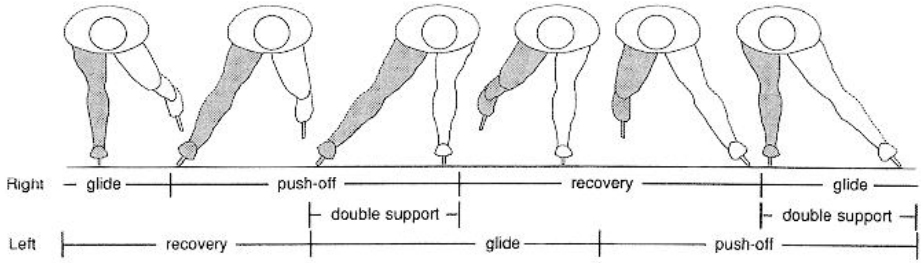
De schaatsbaan is 400 meter lang en kan worden opgedeeld in vier delen, zie Figuur1. Afgezien van de start is schaatsen een cyclische sport. Om slagdetectie mogelijk te maken, is het eerst nodig om de schaatscyclus beter te begrijpen.



Figuur 1. **Delen van een ijsbaan**

3.1.1 Schaatstechniek

Schaatsers bewegen zich voort via de 'gliding technique'. In Figuur 2 (Allinger & Van den Bogert, 1997, p. 279) is de schaatscyclus op het rechte eind weergegeven. Elk been doorloopt in de cyclus drie fases: glijfase (glide), afzet (push-off) en zwaaifase (recovery). In Bijlage 1 is de schaatscyclus uitgebreider beschreven. De afzet eindigt zodra de schaats loskomt van het ijs. De periode tussen twee opeenvolgende momenten van loskomen wordt in dit project gedefinieerd als 'slag'.



Figuur 2. **Frontal view of a speed skater showing the three phases of a skating stroke. Herdrukt van "Skating technique for the straights, based on the optimization of a simulation model," door T.L. Allinger & A.J. Bogert, 1997, *Medicine & Science in Sports & Exercise, 29*(2), p. 279.**

3.1.2 Verschil recht stuk, bocht en start

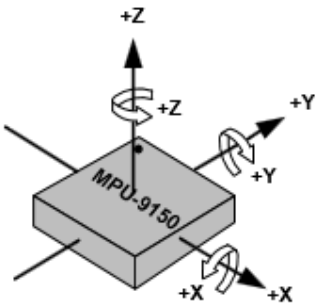
De techniek in de bocht verschilt op een aantal punten van het rechte stuk. Schaatsers leunen gedurende de hele bocht naar links (De Koning, De Groot & Van Ingen Schenau, 1991). In Figuur 3 (De koning et al., 1991, p. 349) is een schaatscyclus in de bocht te zien. De positie van de romp volgt nagenoeg de lijn van de bocht. Op het rechte stuk zwaait de romp heen en weer. Het grote verschil tussen het schaatsen van de bochten en de rechte stukken is dat in de bochten beide benen naar dezelfde kant afzetten (De Boer, Ettema, Van Gorkum, De Groot & Van Ingen Schenau, 1988). Op het rechte stuk zorgt een afzet van het linkerbeen voor een verandering in richting naar rechts. In de bocht zorgt een afzet van het linkerbeen daarentegen voor een verandering in richting naar links. In de bochten duurt de linkerslag korter dan de rechterslag door een kortere glijfase (De Boer, Ettema, Van Gorkum, De Groot & Van Ingen Schenau, 1987). De slagfrequentie in de bochten ligt doorgaans hoger dan op de rechte stukken. De verhouding is afhankelijk van de snelheid. Uitzonderingen hierop zijn de start en de rechte stukken op sprintafstanden. De bovengenoemde verschillen kunnen gebruikt worden om slagen van de vier

Figuur 3. **Position of the body when skating the curve. Herdrukt van "Speed Skating the Curves: A study of Muscle Coordination and Power Production," door J.J. Koning, G. Groot & G.J. Ingen Schenau, 1991, *International Journal of Sport Biomechanics, 7*(4), p.349.**

delen te scheiden.
De start vergt een andere techniek dan het schaatsen van rondes. De eerste slagen van de start worden uitgevoerd met een techniek die veel weg heeft van het lopen (De Koning, De Groot & Van Ingen Schenau, 1989). Zodra de snelheid hoog genoeg is, gaat men in een vloeiende beweging over op de gliding technique. De slagfrequentie bij de start ligt hoger dan gedurende de rondes. Bij het trainen van starts wordt vaak 40 tot 50m gereden. Uit videobeelden is vastgesteld dat dit gaat om 16 tot 20 slagen. Een start is in dit geval gedefinieerd als de eerste 16 slagen.

3.1.3 Accelerometers en gyroscopen

De FBW sensoren bevatten de MPU-9150 MEMS van InvenSense. Een aantal specificaties van de MPU-9150 sensoren zijn weergegeven in Tabel 1. Het meetbereik en de samplefrequentie zijn instelbaar. De 3-assige accelerometer en de 3-assige gyroscoop zitten in één chip, zie Figuur 4 (InvenSense Inc., 2013, p.20). De accelerometer en de gyroscoop hebben daarom hetzelfde assenstelsel. De chip is gemonteerd op een Arduino Pro Micro board. De data kan via een micro-USB poort worden uitgelezen.

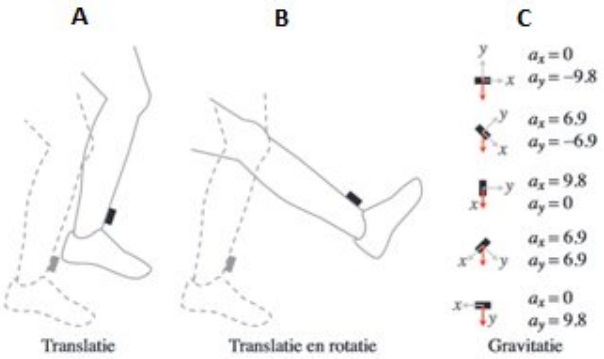


Figuur 4. **Orientation of Axes of Sensitivity and Polarity of Rotation for Accel & Gyro. Herdrukt van "MPU-9150 Product Specification Revision 4.3", van InvenSense Inc., 2013, p.20.**

Het gemeten signaal van een accelerometer op een lichaamsdeel zal vaak bestaan uit een translatie-, rotatie- en een gravitatiecomponent (Van Beers & Verheij, 2015). In Figuur 5 (Van Beers & Verheij, 2015, p. 193) zijn voorbeelden van de drie componenten te zien. Situatie A laat een translatie zien. Hierbij verplaatst het onderbeen terwijl de oriëntatie ongewijzigd blijft. Situatie B is een combinatie van translatie en rotatie. De rotatie zorgt voor een verandering van de oriëntatie van de sensor. In afbeelding C is de gravitatie component (g) te zien. De oriëntatie van de sensor zal bepalen in welke signalen van de accelerometer de gravitatiecomponent zichtbaar is.

Tabel 1 Specificaties MPU-9150

	Accelerometer	Gyroscoop
Aantal assen	3	3
Meetbereik	±2g, ±4g, ±8g en ±16g	±250, ±500, ±1000 en ±2000°/sec
ADC	16-bit	16-bit
samplefrequentie	3,9 – 8000 Hz	3,9 – 8000 Hz



Figuur 5. **Translatie, rotatie en gravitatie. Het been wordt vanuit stilstand (gestreepte lijnen) bewogen naar een andere positie en in het geval van rotatie ook naar een andere oriëntatie (ononderbroken lijnen). Boven 'Gravitatie' is een biaxiale versnellingsopnemer in vijf oriëntaties getekend. De versnellingsopnemer versnelt niet. Toch is het gemeten signaal (α_x en α_y in m/) niet nul. Dit komt door de zwaartekracht (aangegeven met de rode pijlen). Herdrukt van "Meten aan beweging," van R. Beers & R. Verheij, 2015, p. 193.**

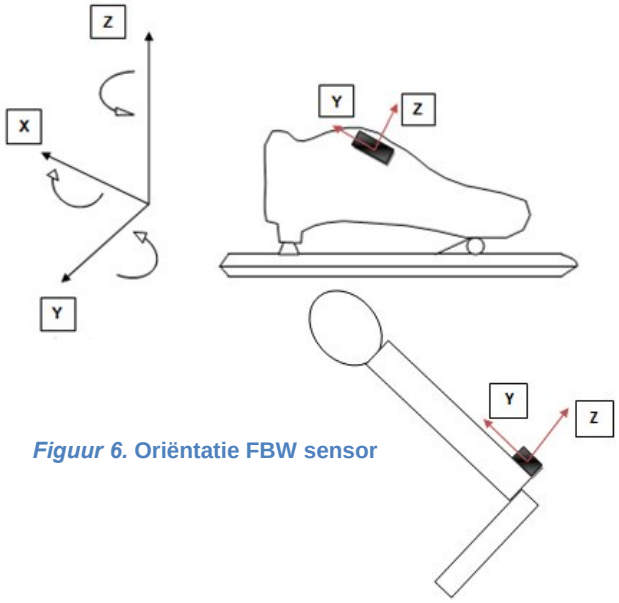
4. Methode

Het algoritme wordt geschreven aan de hand van reeds verworven data. Het gaat om twee metingen, waarbij de proefpersoon een aantal rondes heeft gereden op een 400 meter ijsbaan. Bij beide metingen zijn datasets verkregen van FBW sensoren op de schaats. Bij één meting is ook met een FBW sensor op de onderrug gereden. Daarnaast zijn er ook een drietal datasets van starts gebruikt van sensoren op zowel de romp als de schaats. De samplefrequentie van de sensoren was ingesteld op 500 Hz.

4.1 Plaatsing en oriëntatie FBW sensor

De FBW sensoren zijn op de wreven van beide voeten geplaatst, onder de veters van de schaats. De datasets van de romp zijn verkregen door een FBW sensor op de onderrug te plaatsen. De FBW sensor werd in dit geval in de achterzak van een schaatspak geplaatst. In Figuur 6 is de oriëntatie van de FBW sensor te zien. De rode pijlen in de twee rechterplaatjes geven de positieve assen van de accelerometer aan. De positieve x-as wijst in beide gevallen naar links (het papier in). Het linkerplaatje geeft een overzicht van het assenstelsel van de FBW sensoren. Hierin zijn de pijlen de positieve assen van de accelerometer. De rotatiepijlen geven de positieve richtingen van de gyroscoop aan.

Door de oriëntatie van de FBW sensor zal de gravitatiecomponent bij stilstand voornamelijk te zien zijn in het y- en z-signaal van de accelerometer. Verticale en voor-achterwaartse versnellingen zijn eveneens te verwachten in de y- en z-signalen van de accelerometer. Zijwaartse versnellingen worden voornamelijk zichtbaar in het x-signaal van de accelerometer.



Figuur 6. Oriëntatie FBW sensor

4.2 Ontwikkeling van de algoritmes

De stapfrequentie is al in verscheidene studies bepaald aan de hand van MEMS, met name accelerometers. Ying et al. (2007) hebben drie methodes geschreven die allen gebaseerd zijn op piekdetectie. Kuznietsov (2012) heeft een piekdetectie algoritme geschreven om de stapfrequentie te bepalen bij sprinters. Uit bovenstaande onderzoeken is gebleken dat piekdetectie een snelle en effectieve manier is om de stapfrequentie te bepalen. Om deze redenen is geprobeerd om een algoritme voor slagdetectie op een soortgelijke wijze te ontwikkelen. Er zijn in totaal 4 algoritmes geschreven, zie tabel 2. Het is niet gelukt om een algoritme te schrijven dat zowel starts als rondes kan analyseren. Startslagen waren beter te detecteren uit andere signalen dan de slagen van de rondes. De vier algoritmes zijn in een GUI geplaatst. Een overzicht van de GUI is te vinden in Bijlage 2.

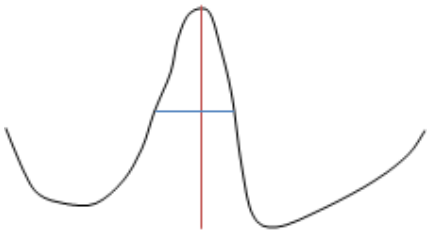
Tabel 2. Overzicht algoritmes

	Gebaseerd op	Type sensor
Algoritme 1 (rondes)	Rompsensor	accelerometer
Algoritme 2 (rondes)	Schaatssensoren	gyroscoop
Algoritme 3 (starts)	Rompsensor	accelerometer
Algoritme 4 (starts)	Schaatssensoren	accelerometer

De algoritmes zijn geschreven in MATLAB R2014b (The Mathworks, USA). Bij het detecteren van de pieken is gebruik gemaakt van de reeds in MATLAB ingebouwde functie 'findpeaks'. MATLAB verstaat onder een 'peak' een datapunt dat groter is dan de twee aangrenzende datapunten of een datapunt dat een oneindige waarde heeft. Deze definitie maakt het alleen mogelijk om positieve pieken te detecteren. De mogelijkheden van findpeaks wordt uitgelegd aan de hand van onderstaande code:

```
[pks,locs,w,p] = findpeaks(x,'MinPeakDistance',0.2*fs,'MinPeakHeight',40);
```

De functie kan maximaal vier outputvariabelen geven: *pks* (y-waardes piek), *locs* (x-waardes piek), *w* (width) en *p* (prominence). In Figuur 7 is een fictieve piek getekend. De rode lijn is de prominence van de piek en de blauwe lijn is de width. De width van de piek wordt door MATLAB automatisch berekend op de halve hoogte van de prominence. De input die aan de functie wordt gegeven is het signaal waarop piekdetectie uitgevoerd moet worden (*x*). Achter de komma kunnen verschillende voorwaarden worden gegeven waaraan de pieken moeten voldoen. Bij de vier algoritmes zijn dat een minimale afstand (MinPeakDistance) tussen de pieken en een minimale piekhoogte (MinPeakHeight).



Figuur 7. **Prominence en width**

In Tabel 3 zijn de waardes weergegeven die zijn gebruikt in de algoritmes.

Tabel 3. **MinPeakDistance en MinPeakHeight van de algoritmes**

	MinPeakDistance	MinPeakHeight
Algoritme 1 (rompsensor)	0.4 (s)	10 m/s²
Algoritme 2 (schaatssensoren)	0.4 (s)	100 °/s
Algoritme 3 (rompsensor)	0.2 (s)	10 m/s²
Algoritme 4 (schaatssensoren)	0.2 (s)	40 m/s²

De outputvariabelen van findpeaks zijn gebruikt om de volgende parameters te berekenen:

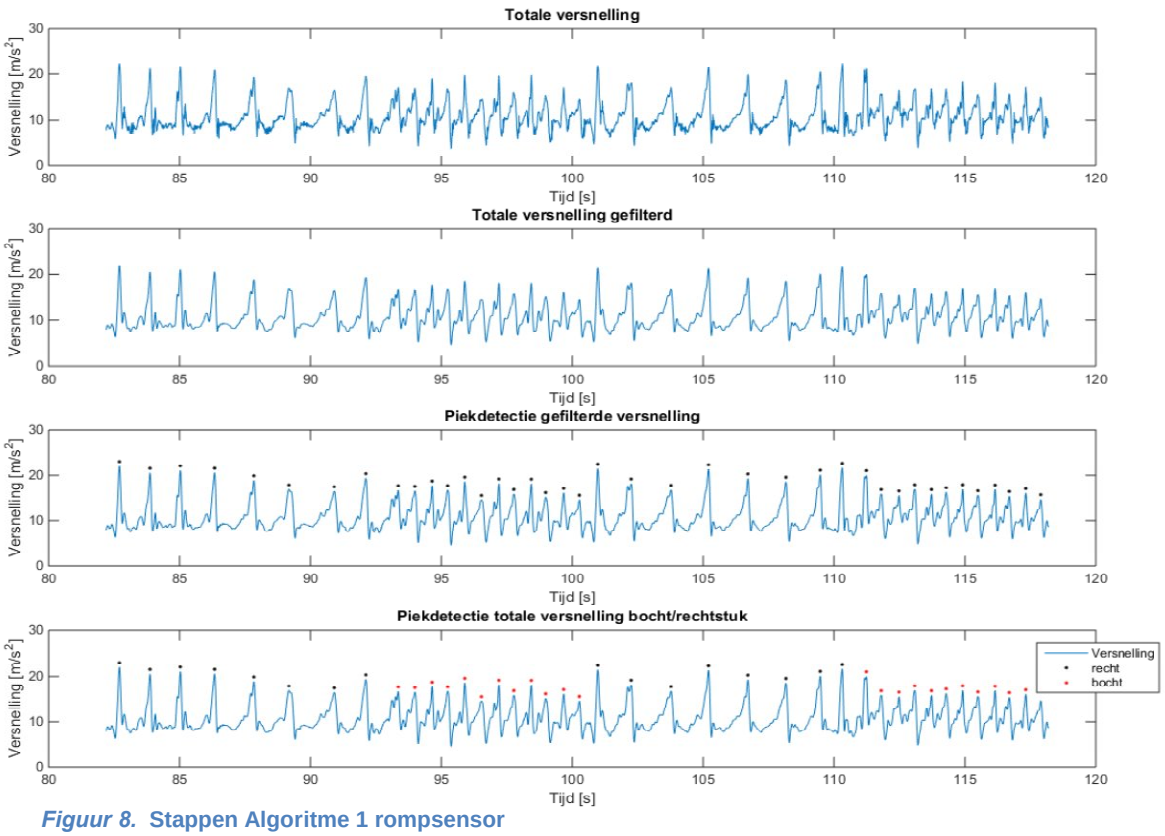
- Aantal slagen (n) = aantal gevonden pieken
- Slagtijd (s) = de tijd tussen twee opeenvolgende pieken
- Slagfrequentie (n/min) = 60/slagtijd

4.2.1 Algoritme 1 (rondes rompsensor)

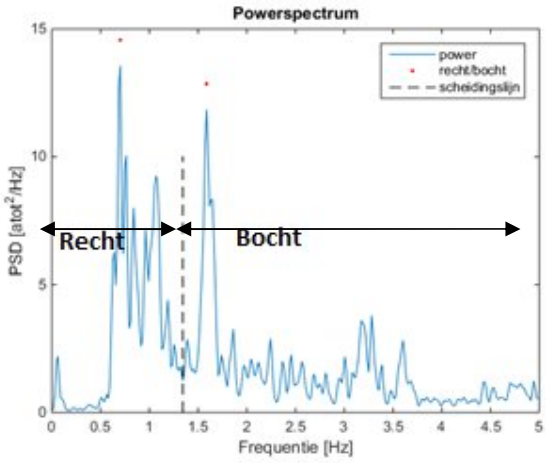
Dit algoritme is gebaseerd op rompversnellingen en maakt gebruik van de totale versnelling. De formule van de totale versnelling is gegeven in vergelijking 1. Waarbij , en respectievelijk de versnellingen zijn in de x-, y- en z-richting.

(1)

Het voordeel van de totale versnelling is dat de oriëntatie van de sensor geen invloed heeft op het gevormde signaal. Door het kwadrateren is de versnelling op elk moment positief. Daarnaast maakt het niet uit in welke richtingen de verschillende assen staan, omdat de versnellingen in de drie richtingen bij elkaar worden opgeteld. In Figuur 8 zijn de belangrijkste stappen van het algoritme weergegeven.



Figuur 8. **Stappen Algoritme 1 rompsensor**



Figuur 9. **Scheidingslijn bepalen uit het powerspectrum**

Er is gekozen om de slagen van de bochten en de rechte stukken te scheiden op basis van frequentie.De computer maakt gebruik van de discrete Fourier-transformatie (DFT) om het signaal van een tijddomein om te zetten naar een frequentiedomein. Dit kan via de Fast Fourier Transform (FFT). Een power spectral density (PSD) geeft het genormaliseerde powerspectrum en is nauwkeuriger dan de FFT (Van Beers & Verheij, 2015). Er is daarom gekozen om het PSD te gebruiken.

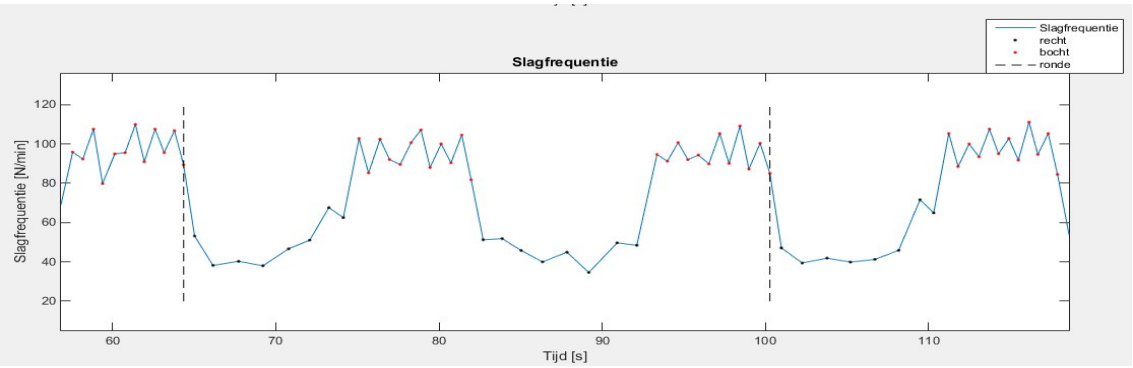
Op basis van het powerspectrum wordt bepaald of een piek een slag is in de bocht of op het rechte stuk. Als een persoon rondes rijdt op een tamelijk constant tempo zijn twee duidelijke pieken te zien in het powerspectrum. De piek met de laagste frequentie representeert de slagen op het rechte stuk. De piek met de hogere frequentie is het gevolg van de bochtslagen. Het algoritme selecteert de datapunten tussen de twee pieken en zoekt het datapunt met de laagste power tussen deze twee pieken. Ter hoogte van dit punt wordt een verticale lijn getrokken. Deze lijn wordt gedefinieerd als de scheidingslijn, zie Figuur 9. De frequentie van het datapunt met de laagste power wordt omgerekend naar een tijdswaarde volgens formule 2. Hierin is de periodetijd en de frequentie van het datapunt.

(2)

Een slag wordt gezien als een bochtslag als de tijdswaarde tussen twee pieken lager is dan de periodetijd. Dit komt overeen met frequenties rechts van de scheidingslijn. Slagen met een

tijdswaarde hoger dan de periodetijd worden gezien als slagen op het rechte stuk. De frequenties liggen in dit geval links van de scheidingslijn.

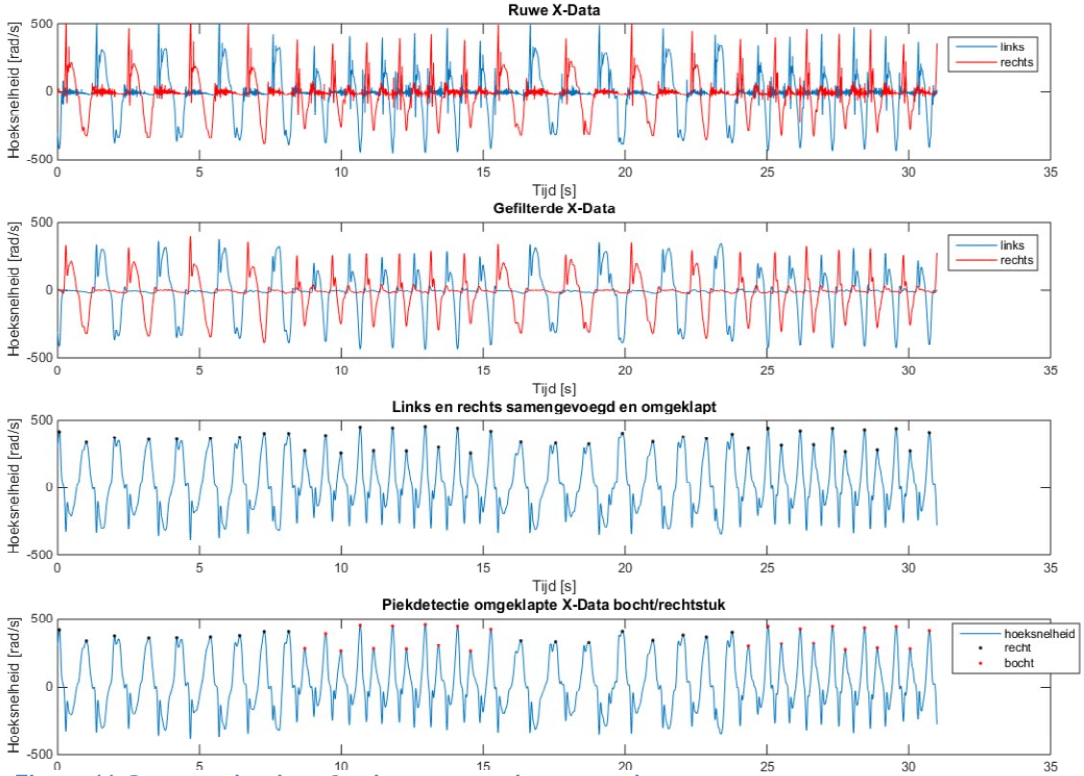
Figuur 10 laat de plot van de slagfrequentie tegen de tijd zien. De rondetijd is vastgesteld als het tijdsverschil tussen de laatste slag van elke tweede bocht. De rondes worden aangegeven met de gestippelde zwarte lijnen.



Figuur 10. Grafiek van de slagfrequentie tegen de tijd

4.2.2 Algoritme 2 (rondes schaatssensoren)

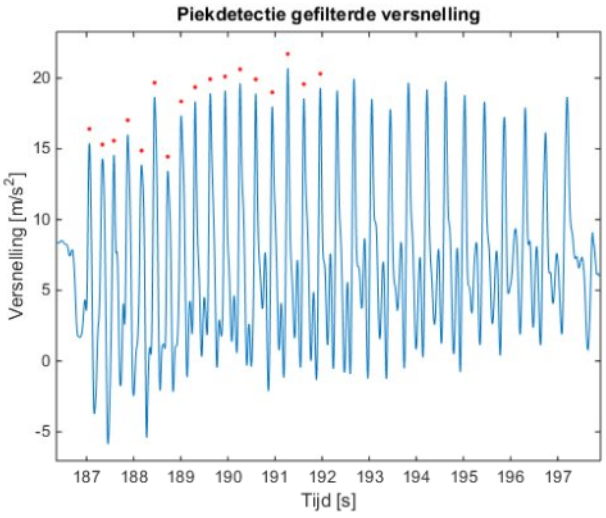
Algoritme 2 detecteert slagen aan de hand van het x-signaal van de gyroscoop. Het x-signaal representeert de plantairflexie en dorsaalflexie bewegingen. In Figuur 11 zijn de verwerkingsstappen weergegeven. De bochten en de rechte stukken worden in dit algoritme net als bij algoritme 1 bepaald aan de hand van het powerspectrum. Het algoritme maakt vervolgens een zelfde grafiek als in Figuur 10.



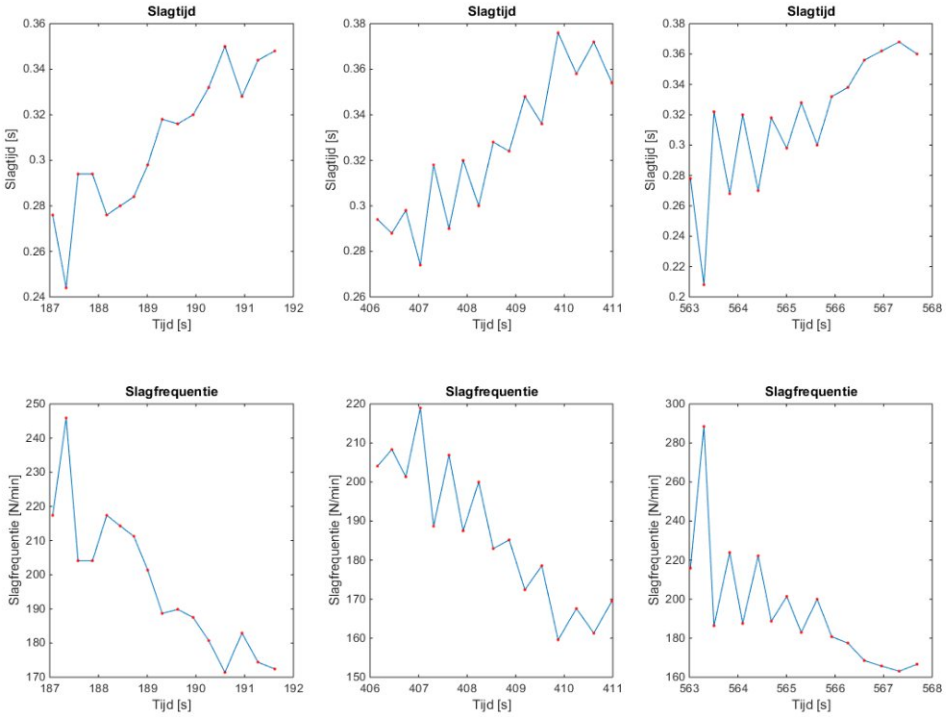
Figuur 11. **Stappen algoritme 2 schaatssensor (gyroscoop)**

4.2.3 Algoritme 3 (start rompsensor)

De starts worden in dit algoritme gedetecteerd aan de hand van het z-signaal. Het gefilterde signaal van één start is te zien in Figuur 12.
Het algoritme zoekt automatisch naar alle starts die in een dataset zitten. Het programma ziet opeenvolgende pieken als een start wanneer de eerste slag en de tweede slag beide korter duren dan 0.4 seconde. Een start is eerder in paragraaf 3.1.2 gedefinieerd als de eerste 16 slagen. Tussen twee opeenvolgende starts moet minimaal 20 seconden zitten. Deze tijdswaarde zorgt ervoor dat de eerste paar slagen na de 16^{e} slag niet worden gezien als een nieuwe start.
Figuur 12 geeft een enkele start weer. Het algoritme plot de slagtijd en de slagfrequentie van elke start in subplots in één figuur, zie Figuur 13.



Figuur 12. **Detectie van de eerste 17 pieken van een start**

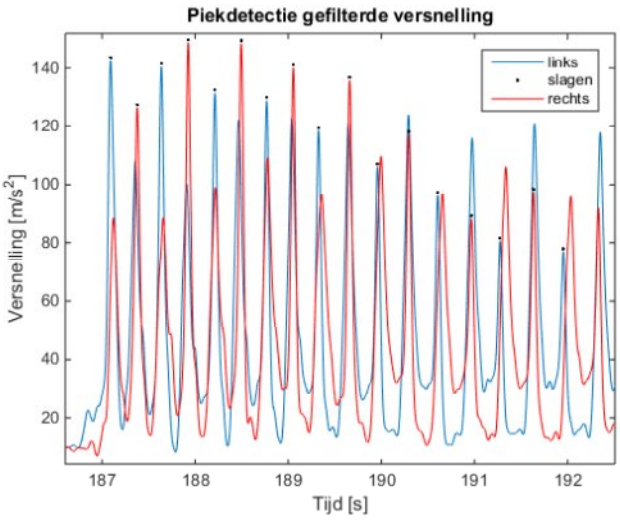


Figuur 13. **Meerdere starts uit één dataset geplot in subplots**

4.2.4 Algoritme 4 (start schaatssensoren)

Het laatste algoritme is geschreven om starts te analyseren met behulp van schaatsversnellingen. In dit geval is, net als bij de het algoritme 1, gebruikt gemaakt van de totale versnelling, zie vergelijking 2. Het algoritme zoekt eveneens automatisch naar alle startjes in een dataset.

In Figuur 14 is te zien dat elke sensor per slag twee pieken genereerd. Hierdoor hebben beide schaatsen op nagenoeg hetzelfde moment een piek. Het algoritme ziet het signaal met de grootste eerste piek als startbeen. De eerste 17 pieken worden daarna systematisch gemarkeerd. De pieken met een oneven nummer worden geselecteerd uit het signaal van het startbeen. De pieken met een even nummer worden geselecteerd uit het signaal van het andere been.



Figuur 14. **Piekdetectie start schaatssensoren**

Het algoritme geeft vervolgens een zelfde output als te zien is in Figuur 13.

4.3 Validatie

De vier algoritmes zijn geschreven op basis van reeds bestaande datasets. Er is videodata nodig om de algoritmes te valideren. Er zijn testen uitgevoerd op twee verschillende locaties: Skeelerbaan en ijsbaan. Op de skeelerbaan zijn zowel rondes als starts uitgevoerd. Op de ijsbaan zijn enkel rondes gereden. De testen op de ijsbaan zijn uitgevoerd tijdens publieke openingstijden. Door de drukte zijn er geen starts uitgevoerd op het ijs. In het starten zit wel een verschil tussen het schaatsen en het skeeleren. Bij het skeeleren gaat de persoon veel sneller over op de gliding technique. Bij de skeelerstarts is daarom gekeken naar de eerste 8 slagen in plaats van 16.
Uit de testen moet blijken:

- Of de algoritmes 98% van de slagen detecteren.

4.3.1 Parameters

In Tabel 4 zijn de parameters gegeven die tijdens de testen zijn gemeten. Een x geeft aan of de parameter is getest. In de rechter kolom wordt per parameter aangegeven welk meetinstrument tijdens de testen zijn gebruikt om de uitkomsten van de algoritmes te valideren. De slagfrequentie wordt berekend uit de slagtijd. Een deviatie in de slagtijd zorgt automatisch voor een deviatie in de slagfrequentie. Daarom wordt in dit geval alleen gekeken naar de slagtijd. De slagtijden zijn wegens tijdgebrek niet gemeten op de ijsbaan. Het aantal slagen en het aantal slagen per bocht/recht stuk is in de meeste gevallen zeer duidelijk af te leiden uit het signaal van de FBW sensoren. De videobeelden zullen vooral dienen als bevestiging. De algoritmes voor de rondes en de starts zijn op een verschillende wijze getest.

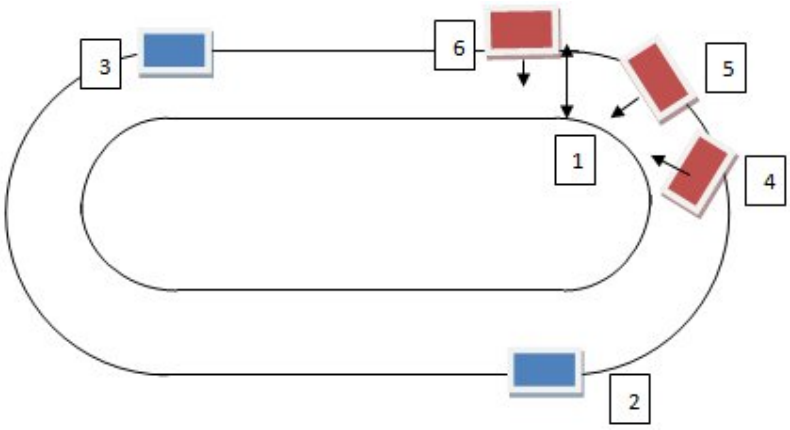
Tabel 4. Te meten parameters

	Skeelerbaan	Ijsbaan	Meetinstrument
Algoritme 1+2 (test1)			
- Totaal aantal slagen	x	x	Video
- Slagen bocht/recht	x	x	Video
- Slagtijd	x	-	High-speed video
- Rondetijd	x	x	Stopwatch
Algoritme 3+4 (test2)			
- Eerste 8 slagen	x	-	Video
- Slagtijd	x	-	High-speed video

4.3.2 Testopzet skeelerbaan

De metingen hebben plaatsgevonden op de ondergrond van de 400 meter kunstijsbaan in Haarlem. Er zijn 4 proefpersonen gemeten met wedstrijdervaring op regionaal/nationaal niveau. Elke proefpersoon heeft 5 rondes gereden. Hierbij is de eerste ronde bedoeld om op gang te komen. De laatste vier rondes zijn op een zo constant mogelijk tempo gereden. Dit is gecontroleerd aan de hand van rondetijden. De rondetijden zijn opgenomen met een stopwatch. De proefpersonen reden de rondes met 3 FBW sensoren. De FBW sensoren zijn op de wreven van beide voeten geplaatst, onder de veters van de schaats. De derde sensor werd in de achterzak van een schaatspak geplaatst. Bij afwezigheid van een achterzak is de FBW sensor, op een vergelijkbare hoogte, op het schaatspak geplakt. De meetopstelling is weergegeven in Figuur 15. Tabel 5 geeft een overzicht van de genummerde onderdelen. De gehele 5 rondes zijn gefilmd met twee videocamera's. De overgang van de tweede bocht naar het eerste rechte stuk is voor elke ronde gefilmd met 3 high-speed camera's op 120 fps. Hs camera 1 is gericht op het einde van de bocht. De proefpersoon reed elke ronde vanzelf het beeld in. Hs camera 2 en 3 stuurden mee met de proefpersoon. De high-speed camera's

werden elke ronde gesynchroniseerd door een persoon te laten springen in het zicht van alle camera's.

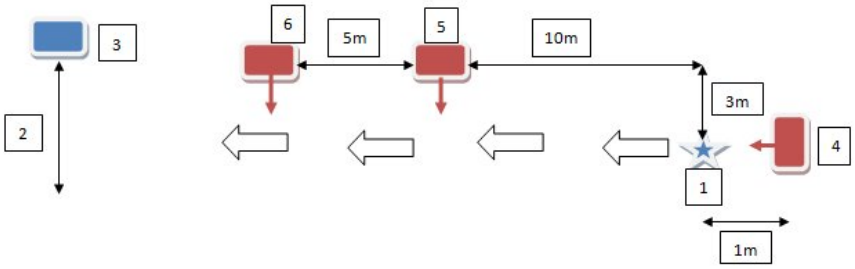


Figuur 15. **Meetopstelling rondes**

Tabel 5. Onderdelen meetopstelling rondes

1	Startlijn + rondetijdmeting
2	Videocamera 1
3	Videocamera 2
4	Hs camera 1
5	Hs camera 2
6	Hs camera 3

De 4 proefpersonen hebben 3 starts uitgevoerd van 30 meter. Voor de starts is dezelfde startlijn gebruikt als in Figuur 15. De FBW sensoren zijn op dezelfde manier bevestigd als bij test 1. Figuur 16 geeft de meetopstelling weer. In Tabel 6 zijn de onderdelen genummerd. Hs camera 1 is achter de proefpersoon geplaatst. De andere twee high speed camera's stuurden mee gedurende de gehele start. De high-speed camera's filmden met 120 fps. De drie high-speed camera's zijn gesynchroniseerd door een persoon voor elke start te laten springen in het zicht van alle camera's. Het testprotocol van beide testen is terug te vinden in Bijlage 3.



Figuur 16. **Meetopstelling starts**

Tabel 6. Onderdelen meetopstelling starts

1	Proefpersoon
2	Finishlijn
3	Video camera
4	HS camera 1
5	HS camera 2
6	HS camera 3

4.3.3 Testopzet ijsbaan

De metingen hebben plaatsgevonden op Ijsbaan Twente in Enschede. Er zijn opnieuw 4 proefpersonen gemeten met wedstrijdervaring op regionaal/nationaal niveau. De opzet van de meting is hetzelfde als beschreven in paragraaf 4.3.2. In dit geval zijn enkel rondes gereden. Tijdens de test is de meetopstelling gebruikt uit Figuur 15, met uitzondering van de high-speed camera's.

4.3.4 Verwerking data FBW sensoren

De data files van de FBW sensoren zijn uitgelezen en omgezet naar .mat files. In het geval van algoritme 1 en 2 zijn de laatste 4 rondes geselecteerd. Uit de signalen en met behulp van de video's zijn het totaal aantal slagen en het aantal slagen per bocht en recht stuk vastgesteld.
De .mat files zijn vervolgens geanalyseerd met de verschillende algoritmes. De resultaten van de algoritmes zijn voor elke proefpersoon naast elkaar gezet in Microsoft Office Excel.

4.3.5 Verwerking videodata

De video's van de rondes zijn bekeken in VLC media player. Het aantal slagen per bocht en recht stuk zijn genoteerd in Microsoft Office Excel. Bij beperkt zicht hebben de signalen uit de FBW sensoren uitsluitsel gegeven.

4.3.6 Verwerking high-speed video

De slagtijden van de starts en de rondes zijn bepaald via videoanalyse in Kinovea. Een slagtijd in de video is gedefinieerd als het tijdsverschil tussen twee opeenvolgende momenten van loskomen van het achterste wiel. De drie high-speed camera's zijn eerst voor elke ronde/start gesynchroniseerd op het moment dat de springende persoon de grond raakt. Van elke proefpersoon zijn de slagtijden van de laatste 4 bochtslagen van 4 rondes berekend. Bij de starts zijn de slagtijden van de eerste 8 slagen bepaald.

4.3.7 Verwerking stopwatch

De rondetijden van de laatste 4 rondes zijn genoteerd in Microsoft Office Excel en afgerond op één decimaal.

4.3.8 Vergelijking algoritmes met videodata

De slagtijden van de algoritmes en de videodata zijn ingelezen in IBM SPSS statistics 21. De slagtijden van de algoritmes zijn in SPSS vergeleken met de gouden standaard (high-speed video). Er zijn regressievergelijkingen opgesteld en tevens zijn de correlaties bepaald tussen de slagtijden van de high-speed video en de verschillende algoritmes. Er zijn Bland-Altman plots gemaakt om de resultaten van de correlaties en de regressievergelijkingen te controleren. Daarnaast is gekeken naar de betrouwbaarheid van de data via de intraclass correlation coefficient (ICC).

5. Resultaten

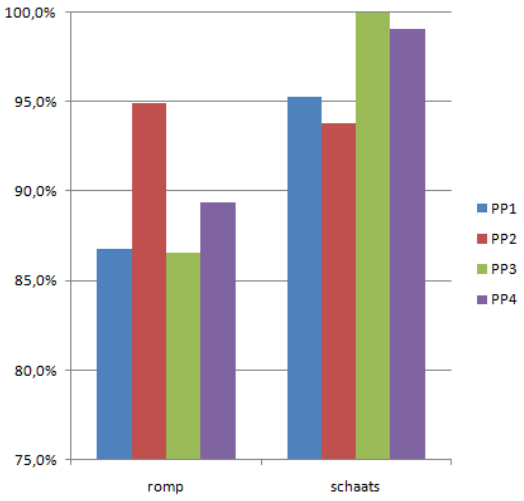
5.1 Skeelerbaan

5.1.1 Algoritme 1 en 2

In Tabel 7 zijn de algemene resultaten van algoritme 1 (romp) en 2 (schaats) weergegeven. Zowel de romp als de schaats scoren hoog, waarbij de schaatssensoren zelfs de gewenste 98% halen. De percentages liggen lager bij het correcte aantal slagen van de bochten en rechte stukken. Het valt op dat het verval bij de rompsensor veel groter is. In Figuur 17 zijn de percentages van elke proefpersoon te zien. Algemeen gezien scoren de schaatssensoren scoren veel beter. Alleen proefpersoon 2 scoort beter met de rompsensor.
In Tabel 7 zijn tevens de resultaten van de rondetijden weergegeven. De rondetijd is in paragraaf 4.2.1 vastgesteld als het tijdsverschil tussen de laatste slag van elke tweede bocht. De rondetijd is goed gekeurd indien deze is berekend over de periode tussen de twee juiste pieken. De schaatssensoren leveren ook in dit geval een beter percentage op. In de onderste twee rijen zijn de gemiddelde absolute deviatie van de rondetijden ten op zichte van de stopwatch en de standaard deviatie van de gemiddelde absolute deviatie gegeven. Deze zijn alleen gebaseerd op de rondetijden die berekend zijn over de juiste periode.

Tabel 7. Resultaten algoritme 1 (romp) en algoritme 2 (schaats)

	Romp	Schaats	Video
Totaal aantal slagen	884	913	910
Totaal goede slagen Percentage	876 *96,3%*	898 *98,7%*	910 -
Totaal goed bocht/recht Percentage	816 *89,7%*	882 *96,9%*	910 -
Gemiste pieken	34	12	-
Foutieve pieken	10	7	-
Rondetijden goed Percentage	6 37,5%	11 68,8%	16 -
Gemiddelde abs deviatie t.o.v stopwatch	0,3 s	0,3 s	-
Standaard deviatie van gemiddelde abs deviatie	0,3 s	0,2 s	-



Figuur 17. **Percentage goede slagen bocht/recht per proefpersoon (skeelertest)**

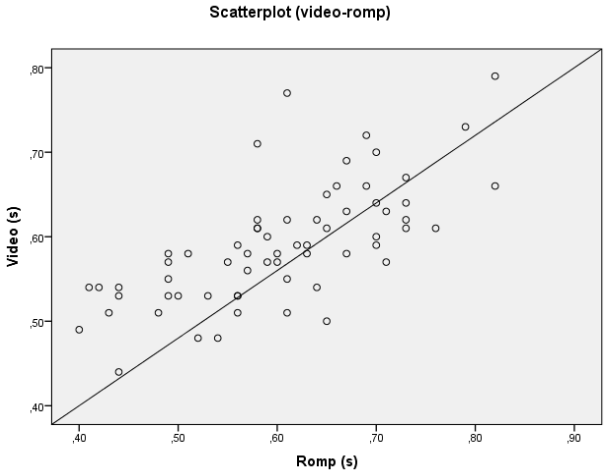
In Tabel 8 zijn de correlaties tussen de slagtijden van algoritme 1 en 2 en de high-speed video te zien. De correlaties zijn voor beide systemen zeer laag. De ICC is een maat voor betrouwbaarheid en geeft aan of bij herhaling dezelfde resultaten worden behaald. Hierbij staat 0 voor geen betrouwbaarheid en 1 voor een hoge betrouwbaarheid. De high-speed video scoort duidelijk hoger dan de twee algoritmes.

Tabel 8. Correlaties tussen de slagtijden van algoritme 1 en de high-speed video en ICC van de drie meetsystemen

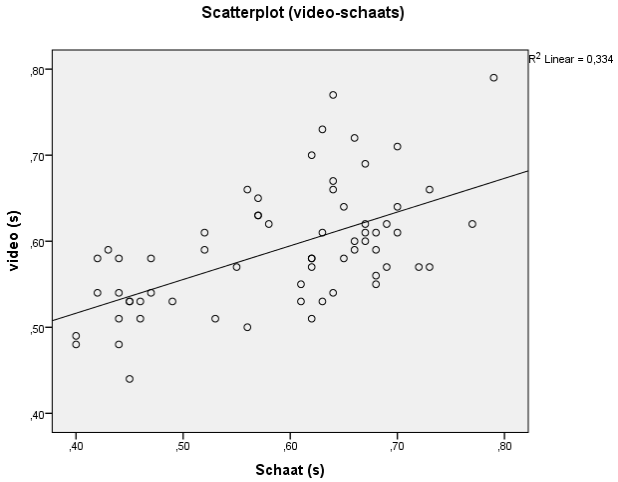
	Correlatie (r)	Sig. (r)	ICC	Sig. (ICC)
Algoritme 1 (romp)	0,689	0,000	0,790	0,006
Algoritme 2 (schaats)	0,578	0,000	0,726	0,019
Video	-	-	0,840	0,001

In Figuur 18 en Figuur 19 zijn twee scatterplots zichtbaar. De algoritmes zijn hier uitgezet tegen de gouden standaard (high-speed video). Formule 3 en 4 geven de regressielijnen.

(3)
(4)



Figuur 18. Scatterplot video/romp

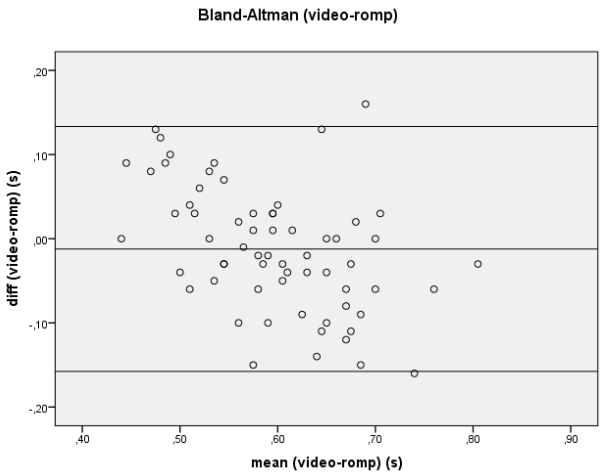


Figuur 19. Scatterplot video/schaats

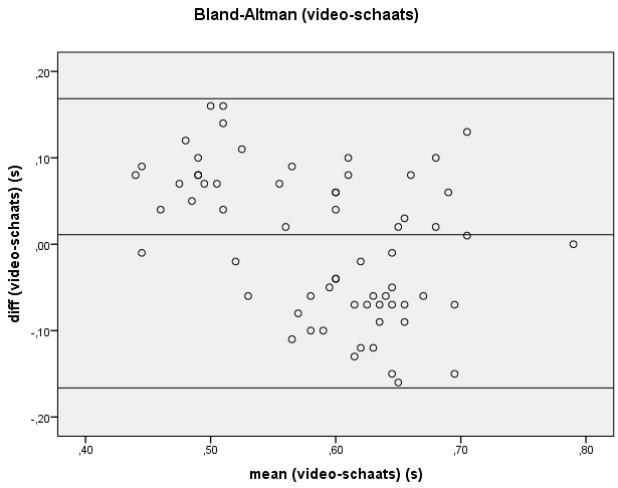
Tabel 9 bevat informatie over de deviaties van de algoritmes ten opzichte van de high-speed video. De gemiddelde deviatie van beide algoritmes is heel laag. De gemiddelde absolute deviatie(Abs gem.) van de algoritmes is veel hoger.

Tabel 9. Deviaties tussen slagtijden van de twee meetsystemen van algoritme 1 (romp), algoritme 2 (schaats) en de high-speed video

	Gemiddelde (s)	Std (s)	Minimum (s)	Maximum (s)	Abs gem. (s)	Std Abs (s)
Diff(video-romp)	-0,0122	0,07423	-0,16	0,16	0,0600	0,04476
Diff(video-schaats)	0,0011	0,08543	-0,16	0,16	0,0748	0,04012



Figuur 20. Bland-Altman video-romp (algoritme 1)



Figuur 21. Bland-Altman video-schaats (algoritme 2)

Figuur 20 en Figuur 21 tonen de Bland-Altman plots. In een Bland-Altman plot wordt de deviatie van twee meetsystemen uitgezet tegen de gemiddeldes van de twee meetsystemen. De middelste horizontale lijn ligt op de waarde van de gemiddelde deviatie uit Tabel 9. De andere twee lijnen geven de grenzen van het 95% betrouwbaarheidsinterval aan. In de scatterplots is te zien dat de slagtijden in de bochten tussen de 0,45 en 0,80 seconde lagen. In de Bland-Altman plots is te zien dat er een hoop slagtijden zijn met een deviatie rond de 0,10 seconde. Een deviatie van 0,10 komt in dat geval overeen met een procentuele afwijking van 12,5-22,2 % ten opzichte van de high-speed video. In tabel 8 is te zien dat de maximale afwijking van beide algoritmes 0,16 seconde was.

5.1.2 Algoritme 3 en 4

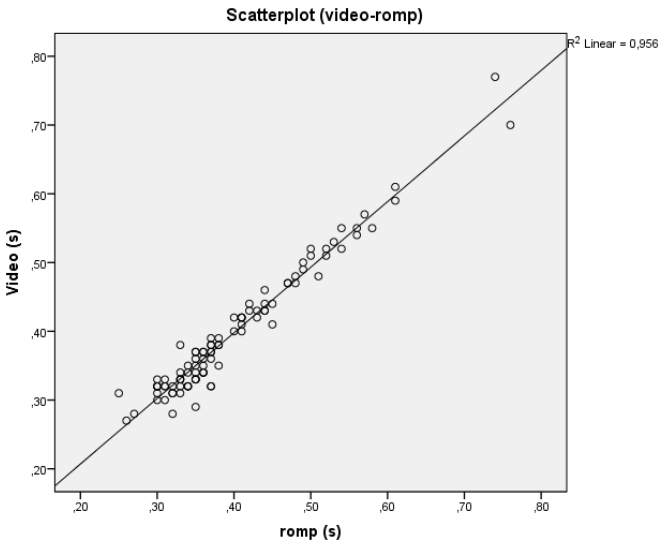
In Tabel 10 zijn de correlaties tussen de slagtijden van algoritme 3 en 4 en de high-speed video te zien. Beide algoritmes hebben een hoge correlatie met de high-speed video. De ICC waardes van de drie meetsystemen liggen dicht bij elkaar.

Tabel 10. **Correlatie tussen de slagtijden van algoritme 3, algoritme 4 en de high-speed video en ICC van de drie meetsystemen.**

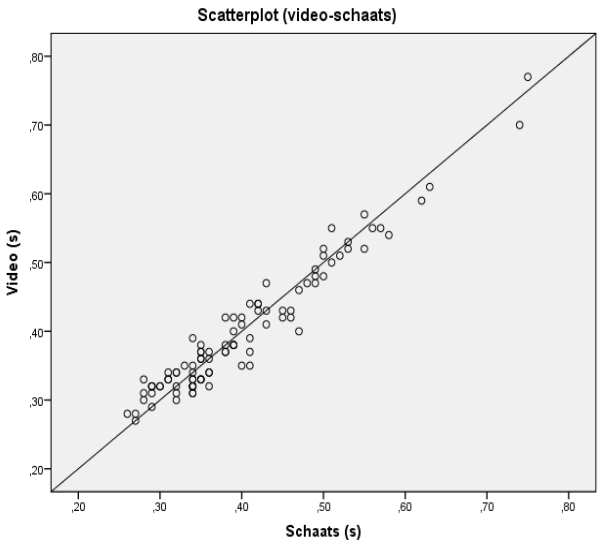
	r	Sig. (r)	ICC	Sig. (ICC)
Algoritme 3 (romp)	0,978	0,000	0,971	0,000
Algoritme 4 (schaats)	0,969	0,000	0,969	0,000
Video	-	-	0,970	0,000

In Figuur 22 en Figuur 23 zijn de twee scatterplots zichtbaar van de algoritmes tegen de high-speed video. De formules van de regressielijnen zijn gegeven in formule 7 en 8.

(5)
(6)



Figuur 22. Scatterplot van starts (video-romp)

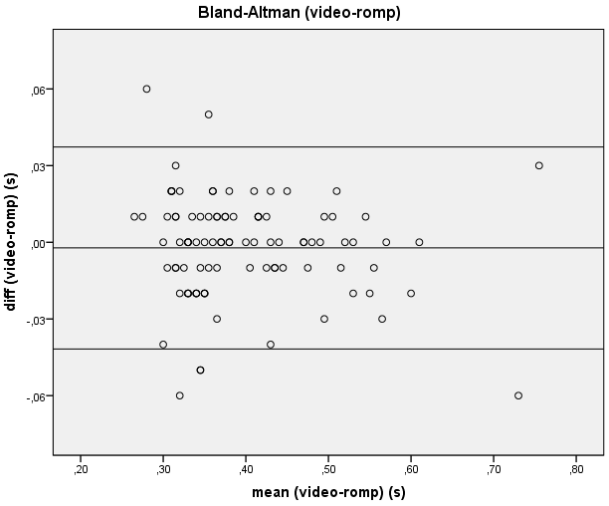


Figuur 23. **Scatterplots van starts (video-schaats)**

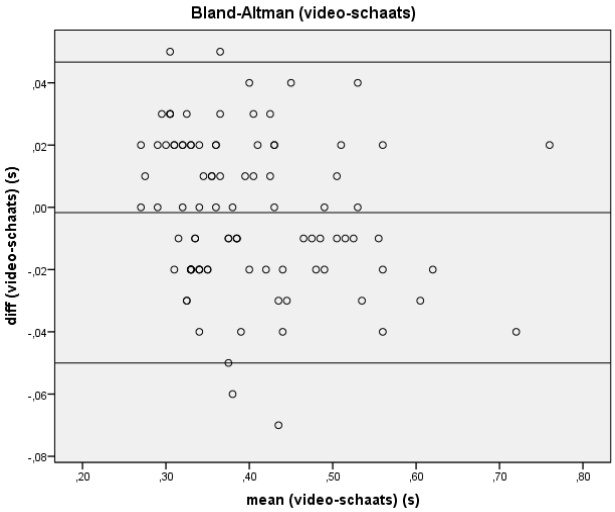
Tabel 11 bevat informatie over de deviaties van algoritme 3 en 4 ten opzichte van de high-speed video. De gemiddelde deviatie van beide algoritmes is heel laag. De gemiddelde absolute deviatie(Abs gem.) van de algoritmes is iets hoger.

Tabel 11. **Deviaties tussen slagtijden van algoritme 3, algoritme 4 en de high-speed video**

	Gemiddelde (s)	Std (s)	Minimum (s)	Maximum (s)	Abs gem. (s)	Std Abs (s)
Diff(video-romp)	-0,0022	0,02017	-0,06	0,06	0,0145	0,01413
Diff(video-schaats)	-0,0017	0,02469	-0,07	0,05	0,0206	0,01352



Figuur 24. Bland-Altman video-romp (algoritme 2)



Figuur 25. Bland-Altman video-schaats (algoritme 2)

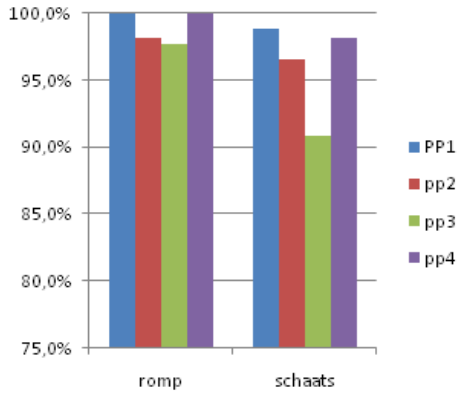
Figuur 24 en Figuur 25 tonen de Bland-Altman plots. De deviaties van de ropsensor zijn meer geclusterd. In de scatterplots is te zien dat de slagtijden bij het starten voornamelijk tussen de 0,30 en 0,60 seconde lagen. In de Bland-Altman plots is te zien dat er een aantal meetwaardes zijn met een deviatie rond de 0,06 seconde ten opzichte van de high-speed video. Een deviatie van 0,06 komt in dat geval overeen met een procentuele afwijking van 10-20 % ten opzichte van de high-speed video.

5.2 Ijsbaan

In Tabel 12 zijn de resultaten van de schaatstest weergegeven. Zowel algoritme 1 (romp) als algoritme 2 (schaats) detecteerden meer dan 98% van de slagen. Algoritme1 was in dit geval foutloos. De percentages liggen lager bij het correcte aantal slagen van de bochten en de rechte stukken. Algoritme 1 scoort ook in dit geval hoger dan 98%. In Figuur 26 zijn de percentages van de segmenten van elke proefpersoon te zien. De Algoritme 1 scoort beter bij alle proefpersonen. Beide algoritmes berekenden bij 13 van de 16 rondes een rondetijd over de juiste periode. De gemiddelde absolute deviatie van de rondetijden ten opzichte van de stopwatch en de standaard deviatie van de gemiddelde absolute deviatie zijn voor beide algoritmes gelijk.

Tabel 12 Resultaten algoritme 1 (romp) en algoritme 2 (schaats) schaatstest

	Romp	Schaats	Video
Totaal aantal slagen	676	660	670
Totaal goede slagen Percentage	670 *100%*	660 *98,5%*	670 -
Totaal goed bocht/recht Percentage	663 *99%*	645 *96,3%*	670 -
Gemiste pieken	0	10	-
Foutieve pieken	7	0	-
Rondetijden goed Percentage	13 81,3%	13 81,3%	16 -
Gemiddelde abs deviatie t.o.v. stopwatch	0,5 s	0,5 s	-
Standaard deviatie van gemiddelde abs deviatie	0,4 s	0,4 s	-



Figuur 26. **Percentage goede slagen bocht/recht per proefpersoon (skeelertest)**

6. Discussie

In dit onderzoek is onderzocht of het mogelijk is om op basis van data uit gyroscopen of accelerometers een algoritme te schrijven dat zelfstandig schaatsslagen kan detecteren op het rechte stuk, in de bocht en tijdens het starten?. Het algoritme moest 98% van de slagen detecteren.

6.1 Algoritme 1 en algoritme 2 skeelertest

6.1.1 Aantal slagen

Algoritme 1 en 2 kwamen bij de skeelertest niet tot de gewenste 98%. Algoritme 1 kwam tot 89,7%. Algoritme 2 kwam tot 96,9%. Door de afwezigheid van ijs, is er getest met skeeleren op de ondergrond van een ijsbaan. De vraag is of de tegenvallende resultaten het gevolg zijn van de verschillen tussen schaatsen en skeeleren. Skeeleren gaat trager dan schaatsen door een grotere weerstand. Daarnaast lagen het aantal slagen van de proefpersonen in de bochten en op de rechte stukken dichter bij elkaar. Dit heeft als gevolg dat de twee pieken uit het powerspectrum dichter tot elkaar komen en dat het scheiden van de segmenten mogelijk minder goed werkt. De signalen van zowel de romp als de schaatssensoren waren minder vloeiend dan in de data van het schaatsen. Dit was vooral te zien in de bochten. De percentages verschillen per proefpersoon. Waarschijnlijk ligt dit aan verschillen in techniek. Het is ook mogelijk dat een sensor niet helemaal goed is geplaatst.

6.1.2 Rondetijden

Algoritme 1 berekende bij de skeelertest in 37,5% van de gevallen een rondetijd over de juiste periode. Algoritme 2 behaalde 68,8%. De rondes zijn in dit project vastgesteld als het tijdsverschil tussen de laatste slagen van elke tweede bocht. Het algoritme maakt een fout als één van deze twee slagen niet als laatste bochtslag wordt gezien. De meeste fouten in de algoritmes ontstonden in de overgang van segmenten. De lage percentages voor de rondetijden is daar een logisch gevolg van. De gemiddelde absolute deviatie van de juiste rondetijden bij de skeelertest is voor beide algoritmes 0.3 seconde. De bijbehorende standaard deviaties zijn 0.3 seconde voor algoritme 1 en 0.2 seconde voor algoritme 2. De algoritmes kunnen een goede schatting geven, maar de berekende rondetijden zijn afhankelijk van hoe een persoon uit de bocht komt. Een stopwatch of elektronische tijdmeting zal daarom in bijna alle gevallen nauwkeuriger zijn.

6.1.3 Slagtijden

De correlatie tussen slagtijden van algoritme 1 en de high-speed video is 0,689. De correlatie tussende slagtijden van algoritme 2 en de high-speed video is 0,578. Deze waardes zijn zeer laag. De lage correlaties en de regressie analyses suggereren dat er een matig verband is tussen de slagtijden van de twee algoritmes en de video. De Bland-Altman plots tonen aan dat de procentuele deviaties in een aantal gevallen tussen de 12,5 en 22,2 % liggen. Deze waardes zijn veel te hoog. Het verschil kan deels te verklaren zijn door verschillen in samplefrequentie, maar dit zou hooguit een honderdste kunnen schelen in de meetfout. De FBW sensoren meten met 500 Hz en de high-speed camera's met 120 fps. De framerate van de camera's is niet hoger gekozen, omdat de beeldkwaliteit anders erg achteruit zou gaan. Er is gekozen om de slagtijden uit de video berekenen op het moment dat het achterste wiel van de skeeler loskwam. Het is de vraag hoe groot de deviaties zouden zijn als een ander moment was gekozen, bijvoorbeeld het eerste grondcontact. De tijd tussen het eerste grondcontact en het loskomen van het achterste wiel is namelijk variabel. Daarnaast zijn de signalen

bij het skeeleren minder vloeiend dan bij het schaatsen. De slagtijden zullen nog een keer op het ijs getest moeten worden om te kijken of de procentuele verschillen daar minder zijn.

6.2 Algoritme 3 en algoritme 4 skeelertest

De correlaties tussen slagtijden van algoritme 3 en de high-speed video is 0,978. De correlaties tussen slagtijden van algoritme 4 en de high-speed video is 0,969. Dit zou betekenen dat er een sterk verband is tussen de slagtijden van de algoritmes en de high-speed video. Uit de Bland-Altman plots blijkt dat een aantal deviaties rond de 0,06 seconde liggen. Dit komt overeen met 10-20 %. De waardes zijn net als bij algoritme 1 en 2 veel te hoog. Een klein aandeel van de verschillen is mogelijk te verklaren door het verschil in samplefrequentie. Het is wederom de vraag of de deviaties anders zouden zijn als een ander moment in de videos was gekozen om de slagtijden te berekenen.

6.3 Algoritme 1 en algoritme 2 schaatstest

6.3.1 Aantal slagen

Bij de schaatstest kwam algoritme 1 tot 99%. Algoritme 2 kwam tot 96,3%. Algoritme 1 scoorde beter bij alle proefpersonen. Bij algoritme 2 ontstonden de fouten vooral in de overgang van het rechte stuk naar de bocht. Er zal een correctie geschreven moeten worden voor de overgang van het rechte stuk naar de bocht. De schaatssensoren zullen dan hoogstwaarschijnlijk ook de 98 % halen.
Beide algoritmes berekende bij de schaatstest in 81,3% van de gevallen een rondetijd over de juiste periode. De percentages zijn veel hoger dan bij de skeelertest. Het verschil komt voornamelijk doordat er minder fouten zijn gemaakt in de overgangen van de segmenten.

6.3.2 Rondetijden

De gemiddelde absolute deviatie van de juiste rondetijden bij de schaatstest is voor beide algoritmes 0.5 seconde. De bijbehorende standaard deviaties zijn 0.4 seconde. De waardes liggen hoger dan bij de skeelertest. De deviaties zijn bij de schaatstest over meer rondes berekend dan bij de skeelertest. De verschillen kunnen het gevolg zijn van de verschillende snelheden waarmee de proefpersonen uit de bocht kwamen bij de twee testen.

6.4 Beperkingen algoritmes

De algoritmes zijn geschreven voor rondes en starts. Het programma is niet geschikt voor afstanden met veel frequentiewisselingen, zoals sprints. Een eindsprint op een recht stuk wordt door de hogere frequentie gezien als bocht. De algoritmes voor de rondes zijn waarschijnlijk beter geschikt voor marathonschaatsers dan voor het langebaanschaanschaatsen.
Bij het rijden van rondes ligt de frequentie op het eerste rechte stuk een stuk hoger dan voor de overige rechte stukken. De persoon moet namelijk op snelheid komen. Deze verhoogde frequentie heeft als gevolg dat er een piek in het powerspectrum ontstaat tussen de pieken van de rechte stukken en de bochten. Hierdoor komt de scheidingslijn niet in alle gevallen op de ‘ideale’ frequentie te liggen. De eerste ronde moet momenteel handmatig uit het signaal geknipt worden. Dit probleem is goed op te lossen indien de sensoren op afstand gestart kunnen worden.
De data moet achteraf verwerkt worden en dat is vrij bewerkelijk. Sensoren die data draadloos kunnen versturen zijn in dit geval gewenst.
Met draadloos bestuurbare sensoren is een mogelijke vervolgstap om het programma om te schrijven tot een real-time feedbacksysteem. Dit zou vooral interessant kunnen zijn voor het starten.

Een andere mogelijkheid is het om het programma verder uit te werken tot een tracking programma. Deze programma's zijn de laatste jaren erg populair onder hardlopers en wielrenners. Een tracking programma kan dienen als een persoonlijke database voor recreatieve schaatsers.

7. Conclusie

In dit onderzoek is onderzocht of het mogelijk is om op basis van data uit gyroscopen of accelerometers een algoritme te schrijven dat zelfstandig schaatsslagen kan detecteren op het rechte stuk, in de bocht en tijdens het starten. Het doel was om minimaal 98 % van de slagen te detecteren. Het is niet gelukt om één algoritme te schrijven dat zowel starts als rondes kan analyseren. Voor de rondes en starts zijn elk twee algoritmes geschreven.
Uit de resultaten van de skeelertesten bleek dat beide algoritmes voor de starts niet voldoen. De twee algoritmes scoorden beide lager dan de gewenste 98%. Bij de schaatstest behaalde algoritme 1 99% en algoritme 2 96,3 %. Het percentage van algoritme 2 zal zeer waarschijnlijk hoger worden als er een correctie wordt geschreven voor de overgang van het rechte stuk naar de bocht.
De slagtijden van algoritme 1 en algoritme 2 vertonen lage correlaties met high-speed video. De lage correlaties kunnen verklaard worden door de grote procentuele deviaties.
De slagtijden van algoritme 3 en algoritme 4 vertonen hoge correlaties met high-speed video, maar de deviaties zijn in een aantal gevallen veel te hoog. De slagtijden moeten nog een keer gevalideerd worden met een schaatstest om hier een betere uitspraak over te kunnen doen.

7.1 Aanbevelingen/suggesties vervolgstappen

- Correctie schrijven voor de overgang van het rechte stuk naar de bocht voor algoritme 2. Bij de correctie kan gebruik worden gemaakt van het feit dat de piek van de rechterschaats in de bocht altijd lager is dan de linkerschaats.
- Slagtijden valideren met schaatstesten.
- De algoritmes werkend krijgen voor sprintafstanden.
  De bochten en rechte stukken zullen hiervoor gescheiden moeten worden op een andere manier dan de frequentie. Een optie is om te kijken naar de kanteling van de schaats.
- Testen of draadloos bestuurbare sensoren gebruikt kunnen worden.
- Het programma omschrijven tot real-time feedback systeem.
- Het programma uitwerken tot een tracking programma.

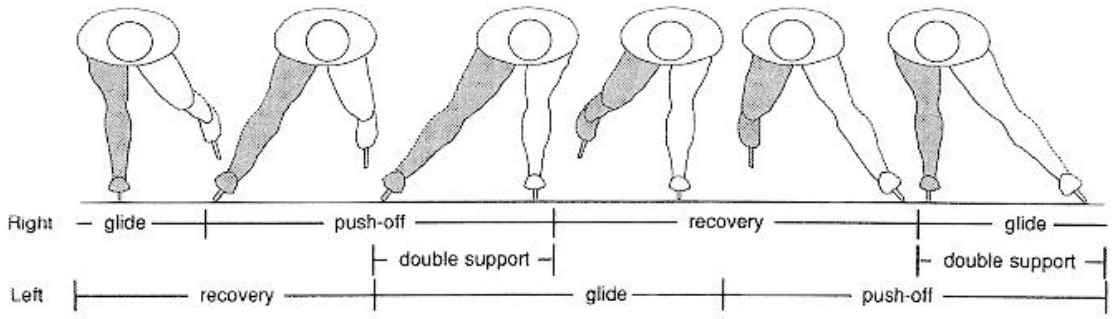
8. Literatuurlijst

1. Allinger, T. L., & Bogert, A. J. van den. (1997). Skating technique for the straights, based on the optimization of a simulation model. *Medicine & Science in Sports & Exercise*, *29*(2), 279-286.
2. Beers, R. van, & Verheij, R. (2015). *Meten aan beweging*. Op 26 maart 2015 opgehaald van Vrije Universiteit Amsterdam, Faculteit der Bewegingswetenschappen : https://bb.vu.nl/bbcswebdav/pid-2136457-dt-content-rid-5152198_2/courses/FBW_B_METENVANFG_2014_140/SyllabusMvFG2015.pdf
3. Boer, R.W. de, Ettema, G.J.C., Gorkum, H. van, Groot, G. de, & Ingen Schenau, G.J. van. (1987). Biomechanical aspects of push-off techniques in speed skating the curves. *International Journal of Sport Biomechanics*, *3*(1), 69-79.
4. Boer, R.W. de, Ettema, G.J.C., Gorkum, H. van, Groot, G. de, & Ingen Schenau, G.J. van. (1988). A geometrical model of speed skating the curves. *Journal of Biomechanics*, *21*(6), 445-450.
5. Boer, R.W. de, & Nilsen, K.L. (1989). The gliding and push-off technique of male and female olympic speed skaters. *International Journal of Sport Biomechanics*, *5*(2), 119-134.
6. Ingen Schenau, G. J. van, & Bakker , K. (1980). A biomechanical model of speed skating. *Journal of Human Movement Studies*, 6, 1-18.
7. Ingen Schenau, G. J. van, Groot, G. de, & Boer, R. W. de. (1985). The control of speed in elite female speed skaters. *J. Biomech.*, *18*(2), 91-96.
8. Ingen Schenau, G.J. van, Boer, R.W. de, & Groot, G. de. (1987). On the technique of speed skating. *International Journal of Sport Biomechanics*, *3*(4), 419-431.
9. InvenSense Inc. (2013). MPU-9150 Product Specification Revision 4.3. Op 15 april opgehaald van http://store.invensense.com/datasheets/invensense/MPU-9150_DataSheet_V4%203.pdf
10. Koning, J.J. de, Boer, R.W. de, Groot, G. de, & Ingen Schenau, G.J. van (1987). Push-off force in speed skating. *International Journal of Sport Biomechanics*, *3*(2), 103-109.
11. Koning, J.J. de, Groot, G. de, & Ingen Schenau, G.J. van (1989). Mechanical aspects of the sprint start in olympic speed skating. *International Journal of Sport Biomechanics. 5*(2), 151-168.
12. Koning, J.J. de, Groot, G. de, & Ingen Schenau, G.J. van. (1991). Speed skating the curves: A study of muscle coordination and power production. *International Journal of Sport Biomechanics*, *7* (4), 344-358.
13. Kuznietsov, A. (2012). Inertial measurement system for performance evaluation of track and field sprinters*. Instrumentation and Measurement Technology Conference (I2MTC)*, *2012 IEEE International*. pp. 1681 -1686.
14. Neville, J., Wixted, A., Rowlands, D., & James, D. (2010). Accelerometers: An underutilized resource in sports monitoring. *Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, *2010 Sixth International Conference on.* pp. 287-290.
15. Veltink, P.H., Bussmann, H.B.J., Vries, W. de, Martens, J., & Lummel, R.C. van. (1996). Detection of static and dynamic activities using uniaxial accelerometers. *IEEE Transactions on Rehabilitation Engineering*, *4*(4), 375-385.
16. Ying, H., Silex, C., Schnitzer, A., Leonhardt, S., & Schiek, M. (2007). Automatic step detection in the accelerometer signal. In S. Leonhardt, T. Falck & P. Mähönen (Eds.), 4th International Workshop on Wearable and Implantable Body Sensor Networks (pp. 80-85). Aachen: Springer Berlin Heidelberg.

9. Bijlage

9.1 Bijlage 1

In Figuur 1 (Allinger& Van den Bogert, 1997, p. 279) is de schaatscyclus op het rechte eind weergegeven. Elk been doorloopt in de cyclus drie fases: glide, push-off en recovery. In het vervolg worden deze fases respectievelijk glijfase, afzet en zwaaifase genoemd. Tijdens de glijfase steunt het lichaam op één been. De glijfase gaat over in de afzet. Deze overgang wordt gekenmerkt door het extenderen van het standbeen. De afzet eindigt zodra de schaats loskomt van het ijs. Het been bevindt zich nu in de zwaaifase. Gedurende de zwaaifase wordt het been naar achter bewogen en geflecteerd. Vervolgens wordt het been naar voren versneld. Als de schaats het ijs raakt, gaat de zwaaifase over in de glijfase. Bovenstaande volgorde beschrijft de cyclus van één been. De cyclus van twee benen bevat een periode van 'double support'. Bij double support maken beide schaatsen contact met het ijs. Double support vindt plaats wanneer het ene been in de glijfase zit en het andere been in de afzet. Tijdens de periode van double support wordt het gewicht verplaatst van het afzetbeen naar het standbeen.

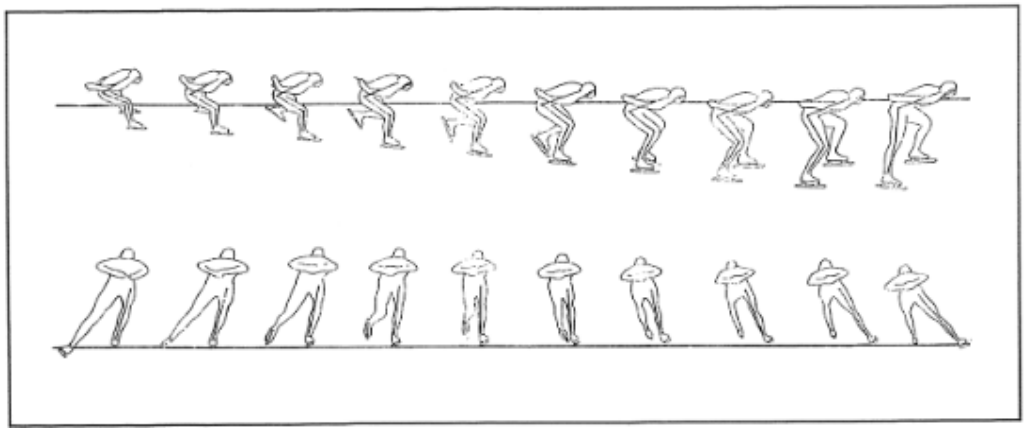


Figuur 1. **Frontal view of a speed skater showing the three phases of a skating stroke.Herdrukt van "Skating technique for the straights, based on the optimization of a simulation model," door T.L. Allinger& A.J. Bogert, 1997, *Medicine & Science in Sports & Exercise, 29*(2), p. 279.**

Schaatsers bewegen zich voort via de 'glidingtechnique'. Het kenmerk van de glidingtechnique is dat de schaatser zich afzet terwijl naar voren wordt gegleden. Schaatsers kunnen zich alleen afzetten in een richting loodrecht op de glijrichting (Van Ingen Schenau, De Boer & De Groot, 1987).

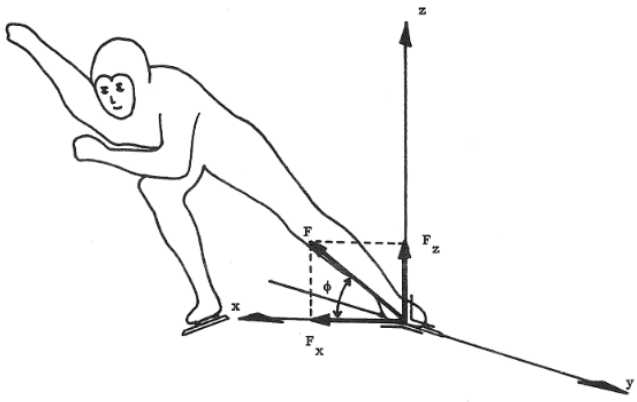
In Figuur 2 (De Boer &Nilsen, 1989, p. 125) is een halve schaatscyclus te zien van een zij- en achteraanzicht. De FBW sensoren zijn bevestigd op de schaats. Het Zephyr systeem wordt op de romp gedragen. Het is van belang om te achterhalen welke globale bewegingen de schaats en de romp maken gedurende de schaatscyclus, omdat deze bewegingen waarschijnlijk in de signalen van de accelerometers en de gyroscopen zijn terug te zien.
Het figuur begint met de glijfase van het rechterbeen. Het linkerbeen komt in de zwaaifase. Zodra de schaats van het ijs komt, vindt er een plantairflexie beweging plaats t.o.v. de verticaal. Bij klapschaatsen vindt er al een plantairflexie plaats terwijl het ijzer nog enige tijd op het ijs blijft. Tevens zien we een adductie(bijtrekken) van het linkerbeen. De romp wijst in dezelfde richting als het standbeen. Tijdens het bijtrekken gaat de plantairflexie over in een dorsaalflexie beweging t.o.v. de verticaal. De romp verplaatst zich tijdens de dorsaalflexie langzaam in de richting van het nieuwe standbeen (linkerbeen). Zodra het linkerbeen het ijs raakt, begin van double support, volgt de romp

in dezelfde richting. Tijdens de dorsaalflexie gaat het rechterbeen over in de afzet. Gedurende de afzet beweegt de schaats zich in een loodrechte richting op zijn eigen glijrichting. De rechterschaats kantelt naar binnen gedurende de afzet. Aan het einde van de afzet begint de glijfase van het linkerbeen.



Figuur 2. **Stick diagrams of one complete speed skating stride in 100-msec intervals. Herdrukt van "The Gliding and Push-off Technique of Male and Female Olympic Speed Skaters," door R.W. Boer & K.L. Nilsen, 1989,*International Journal of Sport Biomechanics, 5*(2), p. 125.**

In Figuur 3 (Van Ingen Schenau et al., 1987, p. 424) is te zien dat de afzetkracht F in het x-z vlak ligt, loodrecht op de glijrichting (y).
Het vermogen dat een schaatser levert is het product van de arbeid per slag en de slagfrequentie. De arbeid per slag wordt vastgesteld als de Fx component van de afzetkracht F. Schaatsers reguleren snelheid bijna geheel met de slagfrequentie terwijl de gemiddelde afzetkracht weinig veranderd bij verschillende snelheden (De Koning, De Boer, De Groot & Van Ingen Schenau, 1987). De arbeid per slag op de rechte stukken is nagenoeg constant bij het schaatsen op verschillende snelheden (Van Ingen Schenau et al., 1985). Op de rechte stukken kan de schaatser daarom zelf een slagfrequentie bepalen om tot een bepaalde snelheid en een gewenst vermogen te komen.

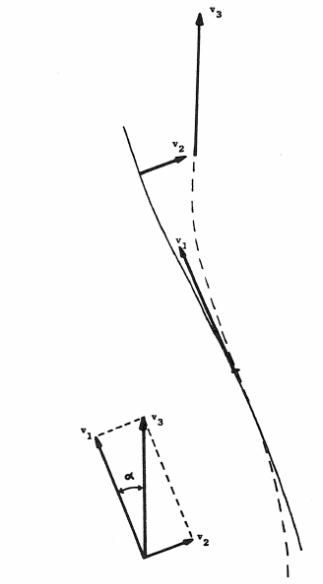


Figuur 3. **The push-off force F lies in a plane (x-z) which is at right angles to the gliding direction (Y) of the skate. The more horizontally F is directed, the larger the effective component FX of F. Herdrukt van "On the Technique of Speed Skating," door G.J. IngenSchenau, R.W. Boer & G. Groot, 1987, *International Journal of Sport Biomechanics, 3*(4), p.424.**

In Figuur 4 (Van Ingen Schenau et al., 1987, p. 425) is de verandering van de snelheidsrichting te zien als gevolg van de afzet. De ononderbroken lijn is de glijrichting. De gebroken lijn is het traject van het lichaamszwaartepunt. Deze legt een traject af in de vorm van een sinusoïde. De afzet loodrecht op de glijrichting zorgt voor een snelheidstoename v2. Deze snelheidstoename zorgt voor een verandering van de totale snelheid van v1 naar v3. Daarnaast verandert v2 de richting van het lichaamszwaartepunt met een hoek α.

Voorgaande beschrijvingen van de schaatscyclus hadden allemaal betrekking op de schaatsbeweging op het rechte eind. De techniek in de bocht verschilt op een aantal punten.

Schaatsers leunen gedurende de hele bocht naar links (De Koning, De Groot & Van Ingen Schenau, 1991). In Figuur 5 (De koning et al., 1991, p. 349) is een schaatscyclus in de bocht te zien. De positie van de romp volgt nagenoeg de lijn van de bocht. Het is de vraag of er duidelijke cycli zijn te zien in het versnellingssignaal van de romp.



Figuur 4. **Since skaters can only push of in a direction perpendicular to the gliding direction of the skate (solid line), the center of gravity follows a sinuous trajectory. The result of the push-off is a velocity increment v2 which increases the total velocity of the skater from v1 to v3. Due to the sideward push-off, the result of the push-off is not only an increase of kinetic energy but also a change of direction *α* of the center of gravity. Herdrukt van "On the Technique of Speed Skating," door G.J. IngenSchenau, R.W. Boer & G. Groot, 1987, *International Journal of Sport Biomechanics, 3*(4), p.425.**

Een horizontale positie van de romp is één van de technische factoren die invloed heeft op de rondetijden (Van Ingen Schenau et al., 1987). Elite schaatsers zijn in staat om deze houding langere tijd aan te nemen. Het is daarom wederom de vraag of er duidelijke cycli waarneembaar zijn in het versnellingssignaal. Indien vermoeidheid optreedt, is het aannemelijk dat de schaatster de romp minder stabiel kan houden. Dit kan invloed hebben op het versnellingssignaal van de romp.

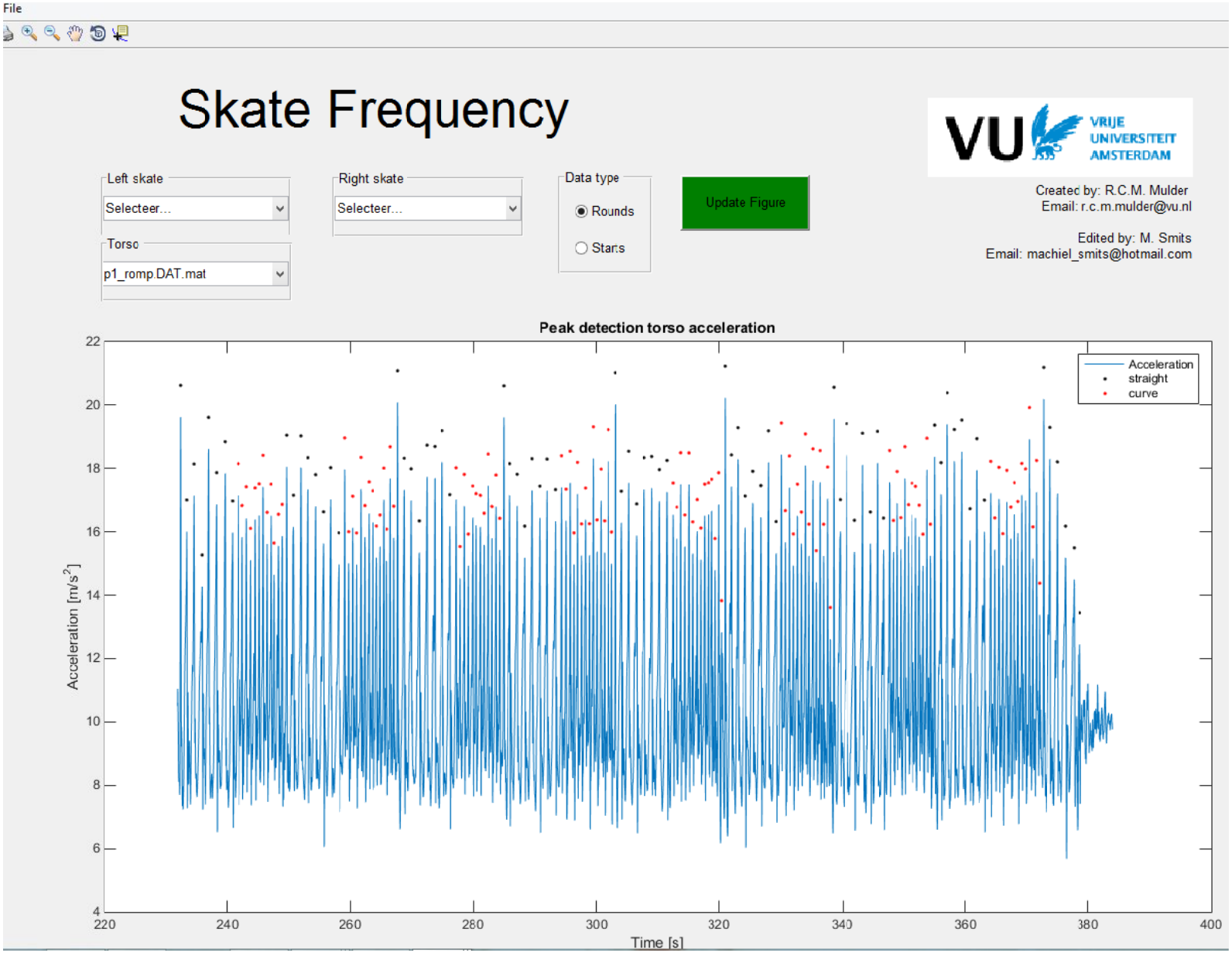
Een zijwaartse afzet naar links zorgt voor een verandering in richting naar rechts. Het grote verschil tussen het schaatsen van de bochten en de rechte stukken is dat in de bochten altijd naar rechts wordt afgezet (De Boer, Ettema, Van Gorkum, De Groot & Van Ingen Schenau, 1988).In de bochten duurt de linkerslag korter dan de rechterslag door een kortere glijfase (De Boer, Ettema, Van Gorkum, De Groot & Van Ingen Schenau, 1987). Dit verschil komt vooral door het 'pootje over' principe. Het rechter been stapt over het linkerbeen tijdens de afzet van het linkerbeen.

Op het rechte stuk kan een schaatser zelf een slagfrequentie bepalen. In de bocht ligt deze slagfrequentie (f) vast volgens vergelijking 1 als de schaatser de radius (R) wil volgen met een bepaalde snelheid (Van Ingen Schenau et al., 1987, p. 427) .

$$f = \frac{v_2}{v_1} \quad (1)$$

In deze vergelijking is v2 de snelheidsverandering door de afzet en v1 is de gemiddelde snelheid. De arbeid per slag bepaalt de grootte van v2. De snelheid die bereikt kan worden in een bocht is daarom sterk afhankelijk van de arbeid per slag en de snelheid voor het ingaan van de bocht (Van Ingen Schenau et al., 1987).

Figuur 5. **Position of the body when skating the curve.Herdrukt van "Speed Skating the Curves: A study of Muscle Coordination and Power Production," door J.J. Koning, G. Groot & G.J. IngenSchenau, 1991, *International Journal of Sport Biomechanics, 7*(4), p.349.**


File
Skate Frequency
Left skate
Selecteer...
Right skate
Selecteer...
Data type
Rounds
Stairs
Update Figure
Torso
p1_romp.DAT.mat
VU VRIJE UNIVERSITEIT AMSTERDAM
Created by: R.C.M. Mulder
Email: r.c.m.mulder@vu.nl
Edited by: M. Smits
Email: machiel_smits@hotmail.com
Peak detection torso acceleration
Acceleration
straight
curve
Acceleration [m/s²]
Time [s]
22
20
18
16
14
12
10
8
6
4
220
240
260
280
300
320
340
360
380
400

9.3 Bijlage 3

Testprotocol skeelerbaan (ijsbaan)

Benodigdheden:

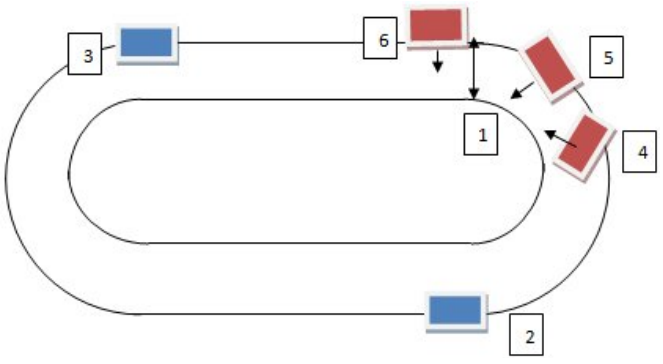
- 4 proefpersonen +skeelermateriaal
- 2 video camera's
- 3 statieven
- 3 FBW sensoren
- Laptop
- 400m skeelerbaan
- 3 HS camera's
- Stopwatch
- Micro-USB kabels
- tape

Test 1 (rondes)
Te meten parameters:

- Aantal slagen
- Aantal slagen per segment
- Slagtijd
- Rondetijd

1	Startlijn + rondetijdmeting
2	Videocamera 1
3	Videocamera 2
4	Hs camera 1
5	Hs camera 2
6	Hs camera 3

Op de proefpersoon worden 3 FBW sensoren bevestigd. De eerste sensor wordt op de onderrug geplakt met tape. De tape wordt eenmalig om het middel van de proefpersoon gewikkeld om te voorkomen dat de tape loslaat. De andere twee sensoren worden geplaatst onder de veters van de skeelers, ter hoogte van de wreven. De proefpersoon zal 5 rondes rijden op een zo constant mogelijk tempo. Er wordt gestart aan het begin van een recht stuk. De eerste ronde is nodig om op gang te komen. De 4 rondes worden continu gefilmd met videocamera's. Rondetijden worden opgenomen met een stopwatch. Het aantal slagen en slagen per segment wordt uit de video bepaald. De slagtijd wordt gemeten met high-speed video op 120 fps. Het is niet mogelijk om dit voor alle slagen te doen. De video bestanden worden dan veel te groot en bovendien rijdt de proefpersoon van de camera weg. High-speed camera's worden op een vaste positie neergezet aan het einde van de bocht en op het begin van het rechte stuk. Gedurende de rondes worden de laatste slagen van de bocht en de eerste slagen van het rechte stuk gefilmd. Dit is afhankelijk van hoelang de proefpersoon goed in beeld is. De High-speed camera's worden gesynchroniseerd door een proefpersoon te laten springen in het zicht van alle camera's. Een slag wordt in de video gedefinieerd als het eerste moment van loskomen van de skeeler. De slagfrequentie wordt berekend uit de gevonden slagtijden. De sensoren en camera's worden uitgelezen na elke proefpersoon.



Volgorde test 1:

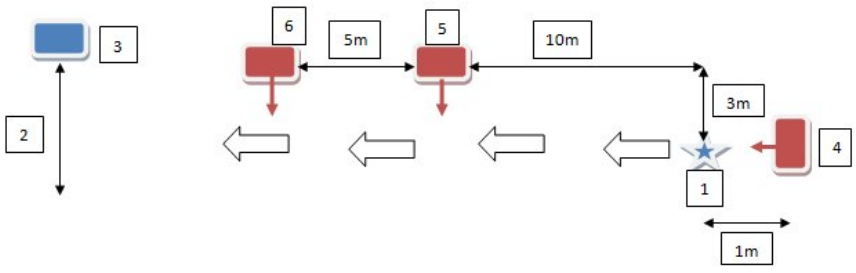
- FBW sensoren gesynchroniseerd aanzetten
- FBW sensoren plaatsen op proefpersoon
- Videocamera's aan
- Starten
- 200meter voor bocht high-speed camera's aan
- Synchroniseren high-speed camera's via sprong
- High-speed camera's uit als proefpersoon voorbij is
- Bovenstaande 3 stappen 4x herhalen
- Sensoren en video's uitlezen
- Volgende proefpersoon

Test 2 (starts)

Te meten parameters:

- Eerste 8 slagen
- Slagtijd

Elke proefpersoon voert 3 starts uit van 30 meter. De proefpersoon krijgt de 3 FBW sensoren op dezelfde manier bevestigd als bij test 1. De slagtijd en slagfrequentie worden opnieuw bepaald met high-speed video op 120 fps. De camera's worden gesynchroniseerd door een persoon te laten springen in het zicht van alle camera's. Er wordt 1 camera achter de proefpersoon geplaatst en 2 aan de zijkant. Er wordt geprobeerd om de slagtijden van de eerste 8 slagen te berekenen. Uit de pilot bleek dat de eerste 8 slagen vallen in de eerste 15 meter. Dit kan per proefpersoon verschillen.
Een video camera filmt de proefpersoon tot aan de finishlijn als controle voor het aantal slagen. Een proefpersoon voert 3 starts achter elkaar uit. Na elke proefpersoon worden de sensoren en de camera's uitgelezen.



1	Proefpersoon
2	HS camera 1
3	HS camera 2
4	HS camera 3
5	Video camera
6	finishlijn

Volgorde test 2:

- FBW sensoren gesynchroniseerd aanzetten
- FBW sensoren plaatsen op proefpersoon
- Videocamera aan
- High-speed camera's aan

- Synchroniseren high-speed camera's via sprong
- Starten
- High-speed camera's uit
- Bovenstaande stappen 3x herhalen
- Sensoren en video's uitlezen
- Volgende proefpersoon

Tijdsplanning meting

Vrijdag 29 mei 2015, Ijsbaan Haarlem.

Instellen camera's en sensoren	Voorafgaand aan de meting
Aanvang	13.45
Warm up proefpersonen + klaarzetten meetopstelling	14.00 – 14.15
Test 1	14.15 – 14.30 PP 1 14.30 – 14.45 PP 2 14.45 – 15.00 PP 3 15.00 – 15.15 PP 4
Test 2	15.15 – 15.25 PP 1 15.25 – 15.35 PP 2 15.35 – 15.45 PP 3 15.45 – 15.55 PP 4
Opruimen	15-55 – 16.00

9.4 Bijlage 4

9.4.1 Algoritme 1

```
clear all
close all
clc
%% info
% Author: Machiel Smits
% machiel_smits@hotmail.com
% FBW sensor met romp versnelling
% Script voor slagdetectie specifiek voor het rijden van rondjes.

%% Inladen Data
%romp = load('filename.mat'); %

%romp signalen toekennen
accrompx = romp.data.Acceleration(:,1);
accrompy = romp.data.Acceleration(:,2);
accrompz = romp.data.Acceleration(:,3);
%% tijdas romp
fs = romp.data.sample_frequency;
dt = 1/fs;
N = length(accrompx);
k = [0:N-1];
t = k*dt;

%% Totale versnelling berekenen
atot = sqrt((accrompx.^2)+(accrompy.^2)+(accrompz.^2));
atot = atot(int);
t = t(int);
plot(t,atot)

%% powerspectrum en filteren
[pxx,f] = pwelch(atot-mean(atot), [],[],[],fs);
df = f(2)-f(1); % afstand tussen twee datapunten van f en pxx
minpeakdis = 0.3/df; %minimale afstand pieken bocht en recht stuk = 0.3 hz.
fmin = find(f>=0.2); % hoge piek bij omstreeks 0hz vermijden.
[powerpks,powerlocs,powerw,powerp] =
findpeaks(pxx(fmin),'MinPeakDistance',minpeakdis,'MinPeakHeight',2,'SortStr
','descend');
powerlocs = [powerlocs(1);powerlocs(2)]; % 2 hoogste pieken selecteren
powerlocs = sort(powerlocs);
afstand_tussenpieken = powerlocs(2)-powerlocs(1);
middelpunt = powerlocs(1)+(0.5*afstand_tussenpieken);
findex = fmin((middelpunt-(0.1/df)):1:(middelpunt+(0.1/df)));
lowest = min(pxx(findex)); % laagste power bepalen tussen de twee pieken
minpxx = find(pxx==lowest); % samplenummer van laagste power bepalen

Nfilt = 2;
fc = 10;
Wn = fc/(fs/2);
[B,A] = butter(Nfilt,Wn);
atot_filt = filtfilt(B,A,atot);
plot(t,atot_filt)
%% piekdetectie
 [pks,locs,w,p] =
findpeaks(atot_filt,'MinPeakDistance',0.4*fs,'MinPeakHeight',10);
%Echte pieken hebben een width (w>=20) en een prominence (p>3)
realpiek = find(w>=1 & p>=1);
```

```
realpieklocs = locs(realpiek)';

%% bocht/rechtstuk
piekafstand = diff(realpieklocs);
aantal_pieken = length(realpieklocs);
realpieklocs(2,1) = 0;
for i = 2:aantal_pieken-1
       if (piekafstand(i)<=(fs/f(minpxx))) % als frequentie rechts van de
scheidingslijn van het powerspectrum ligt.
        realpieklocs(2,i) = 1; % bocht
        else
        realpieklocs(2,i) = 0; % recht stuk
       end
end
% correctie voor enkele misslag
for i = 2:aantal_pieken-1
   if realpieklocs(2,i) == 1 % bocht
       if (realpieklocs(2,i-1) == 0 && realpieklocs(2,i+1)==0) % als vorige
en volgende piek recht stuk zijn
        realpieklocs(2,i) = 0; % recht stuk
        else
        realpieklocs(2,i) = 1; % bocht
        end

    else %recht stuk
        if (realpieklocs(2,i-1) == 1 && realpieklocs(2,i+1)==1) % als
vorige en volgende piek bocht zijn
        realpieklocs(2,i) = 1; % bocht
        else
        realpieklocs(2,i) = 0; % recht stuk
       end
    end
end

%correctie voor misslagen
for i = 2:aantal_pieken-5
    if (realpieklocs(2,i) == 0 && realpieklocs(2,i-1)==1) % als huidige
piek = recht en vorige piek = bocht
        realpieklocs(2,i+2:i+5) = 0; % recht stuk minimaal 6 slagen
    end

    for j = 2:aantal_pieken-1
        if realpieklocs(2,j) == 1 %bocht
             if (realpieklocs(2,j-1) == 0 && realpieklocs(2,j+1)==0) % als
vorige en volgende piek recht stuk zijn
                 realpieklocs(2,j) = 0; % recht stuk
             else
                realpieklocs(2,j) = 1; % bocht
             end

            else %recht stuk
            if (realpieklocs(2,j-1) == 1 && realpieklocs(2,j+1)==1) % als
vorige en volgende piek bocht zijn
                 realpieklocs(2,j) = 1; % bocht
            else
                 realpieklocs(2,j) = 0; % recht stuk
            end
        end
    end
end
```

```
bocht = find(realpieklocs(2,:) == 1);
recht = find(realpieklocs(2,:) == 0);

%% slagtijd en slagfrequentie
piektijden = (realpieklocs(1,:)./fs);
slagtijd = diff(piektijden(1,:));
for i = 1:length(piektijden)-1
    if (realpieklocs(2,i) == 0) % recht stuk
        slagtijd(2,i) = 0; % recht stuk
    else
        slagtijd(2,i) = 1; % bocht
    end
end
slagfrequentie = [60./slagtijd(1,:);slagtijd(2,:)];
figure(1)
subplot(2,1,1),plot(piektijden(1:end-1),slagtijd)% laatste piek heeft geen
slagtijd
xlabel('Tijd [s]')
ylabel('Slagtijd [s]')
title('Slagtijd')
legend('slagtijd','recht(0)/bocht(1)')
subplot(2,1,2),plot(piektijden(1:end-1),slagfrequentie(1,:)) %laatste piek
heeft geen slagfrequentie
xlabel('Tijd [s]')
ylabel('Slagfrequentie [N/min]')
title('Slagfrequentie')
hold on
bocht = find(slagfrequentie(2,:) == 1);
recht = find(slagfrequentie(2,:) == 0);
plot(piektijden(1,recht),slagfrequentie(1,recht),'.k',piektijden(1,bocht),s
lagfrequentie(1,bocht),'.r')
xlabel('Tijd [s]')
ylabel('Slagfrequentie [N/min]')
%% rondetijd
nummer = 1; % segmentnummer
bochtrecht = diff(slagfrequentie(2,:)); % bocht = 1, recht = 0
slagfrequentie(3,1) = nummer; %1e waarde krijgt nummer 1
for i = 1:length(slagfrequentie)-1
if (bochtrecht(i)~=0)%als verschil geen 0 is : overgang van segment
    slagfrequentie(3,i+1) = nummer+1; % overgang segment
    nummer = nummer+1; % naar volgende segment
else
    slagfrequentie(3,i+1) = nummer; % slag krijgt zelfde nummer
end
end

rondetijd_index = mod(slagfrequentie(3,:),4); % segmentnummers die deelbaar
zijn door vier krijgen waarde 0
j = find(rondetijd_index == 0,1,'last'); % zoek laatste waarde die 0 heeft
rondetijden_locs(1) = 1; %startpunt van eerste ronde is eerste piektijd
index = 2; % 1e waarde is startpunt, eindpunt eerste ronde krijgt waarde 2.
for i = 4:4:slagfrequentie(3,j) % aantal rondes vaststellen (veelvoud van 4
segmenten)
rondetijden_locs(index) = find(slagfrequentie(3,:) == i,1,'last'); %zoek
laaststе punt van laatste segment van de ronde
index = index+1;
end

doorkomst = piektijden(rondetijden_locs);
rondetijden = diff(doorkomst); % verschil in doorkomstpunten
```

```
figure(6)
hold on
for i = 2:length(doorkomst)
plot([doorkomst(i) doorkomst(i)], [20 120],'--k');
end
legend('Slagfrequentie','recht','bocht','ronde')

ronde(1,:) = 1:length(rondetijden); % aantal rondes
ronde(2,:) = rondetijden; % bijbehorende rondetijden
ronde(3,:) = diff(rondetijden_locs); % aantal slagen per ronde
ronde(3,1) = ronde(3,1)+1; % correctie omdat 1e slag anders nergens wordt
meegenomen
%% Gemiddelde slagtijd en gemiddelde frequentie
for i = 1:max(slagfrequentie(3,:))
segment = find(slagfrequentie(3,:) == i); %zoek segmentwaardes
gem_slagfrequentie(i) = mean(slagfrequentie(1,segment)); %gem
slagfrequentie segmentwaarde
gem_slagtijd(i) = mean(slagtijd(1,segment)); %gem slagtijd segmentwaarde
aantal_slagen(i) = length(segment); % aantal slagen segment
end
%laatste slag heeft geen slagfrequentie omdat er geen volgende piek is
aantal_slagen(end) = aantal_slagen(end)+1; % laatste slag optellen bij
laatste segment

for i = 1:4 %gem segmentgegevens per ronde scheiden
    if i ==1;
    recht1_slagtijd = gem_slagtijd(i:4:end); %gem slagtijd
    recht1_slagfrequentie = gem_slagfrequentie(i:4:end); % gem frequentie
    recht1_slagen = aantal_slagen(i:4:end); % aantal slagen
    end
    if i ==2;
    bocht1_slagtijd = gem_slagtijd(i:4:end);
    bocht1_slagfrequentie = gem_slagfrequentie(i:4:end);
    bocht1_slagen = aantal_slagen(i:4:end);
    end
    if i ==3;
    recht2_slagtijd = gem_slagtijd(i:4:end);
    recht2_slagfrequentie = gem_slagfrequentie(i:4:end);
    recht2_slagen = aantal_slagen(i:4:end);
    end
    if i ==4;
    bocht2_slagtijd = gem_slagtijd(i:4:end);
    bocht2_slagfrequentie = gem_slagfrequentie(i:4:end);
    bocht2_slagen = aantal_slagen(i:4:end);
    end
end
%% Tabel

% afronden op 2 decimalen
rondetijden = round(rondetijden,2);
recht1_slagtijd = round(recht1_slagtijd,2);
bocht1_slagtijd = round(bocht1_slagtijd,2);
recht2_slagtijd = round(recht2_slagtijd,2);
bocht2_slagtijd = round(bocht2_slagtijd,2);
recht1_slagfrequentie = round(recht1_slagfrequentie,2);
bocht1_slagfrequentie = round(bocht1_slagfrequentie,2);
recht2_slagfrequentie = round(recht2_slagfrequentie,2);
bocht2_slagfrequentie = round(bocht2_slagfrequentie,2);
```

```
aantal_rondes = length(rondetijden); % aantal rondes
% tabel slagtijd
T1 =
table(rondetijden(1:aantal_rondes)',recht1_slagtijd(1:aantal_rondes)',bocht
1_slagtijd(1:aantal_rondes)',recht2_slagtijd(1:aantal_rondes)',bocht2_slagt
ijd(1:aantal_rondes)');
T1.Properties.VariableNames{1} = 'rondetijden';
T1.Properties.VariableNames{2} = 'recht1';
T1.Properties.VariableNames{3} = 'bocht1';
T1.Properties.VariableNames{4} = 'recht2';
T1.Properties.VariableNames{end} = 'bocht2';
T1
%tabel slagfrequentie
T2 =
table(rondetijden(1:aantal_rondes)',recht1_slagfrequentie(1:aantal_rondes)'
,bocht1_slagfrequentie(1:aantal_rondes)',recht2_slagfrequentie(1:aantal_ron
des)',bocht2_slagfrequentie(1:aantal_rondes)');
T2.Properties.VariableNames{1} = 'rondetijden';
T2.Properties.VariableNames{2} = 'recht1';
T2.Properties.VariableNames{3} = 'bocht1';
T2.Properties.VariableNames{4} = 'recht2';
T2.Properties.VariableNames{end} = 'bocht2';
T2
%tabel aantal slagen
T3 =
table(rondetijden(1:aantal_rondes)',recht1_slagen(1:aantal_rondes)',bocht1_
slagen(1:aantal_rondes)',recht2_slagen(1:aantal_rondes)',bocht2_slagen(1:aa
ntal_rondes)',ronde(3,:)');
T3.Properties.VariableNames{1} = 'rondetijden';
T3.Properties.VariableNames{2} = 'recht1';
T3.Properties.VariableNames{3} = 'bocht1';
T3.Properties.VariableNames{4} = 'recht2';
T3.Properties.VariableNames{5} = 'bocht2';
T3.Properties.VariableNames{end} = 'Totaal_slagen';
T3
```

9.4.2 Algoritme 2

```
clear all
close all
clc
%%info
% Author: Machiel Smits
% machiel_smits@hotmail.com
% Script voor slagdetectie specifiek voor starten.
%% Inladen Data
%links = load('filename.mat')
%rechts = load('filename.mat')

%gyro signalen toekennen
gyrolx = links.data.Gyroscope(:,1);
gyrorx = rechts.data.Gyroscope(:,1);
%% tijdas romp
fs = links.data.sample_frequency;
dt = 1/fs;
N = length(gyrolx);
k = [0:N-1];
t = k*dt;

N2 = length(gyrorx);
k2 = [0:N2-1];
t2 = k2*dt;
if (N2>=N)
    t_tot = t(1:N);
    gyrorx = gyrorx(1:N);
else
    t_tot = t(1:N2);
    gyrolx = gyrolx(1:N2);
end
w = gyrolx;
w2 = gyrorx;
wtot = w+w2;
%% plots
[pxx,f] = pwelch(wtot-mean(wtot), [],[],[],fs);
df = f(2)-f(1); % afstand tussen twee datapunten van f en pxx
minpeakdis = 0.3/df; %minimale afstand pieken bocht en recht stuk =  0.3
hz.
fmin = find(f>=0.2); % hoge piek bij omstreeks 0hz vermijden.
[powerpks,powerlocs,powerw,powerp] =
findpeaks(pxx(fmin),'MinPeakDistance',minpeakdis,'MinPeakHeight',1000,'Sort
Str','descend');
powerlocs = [powerlocs(1);powerlocs(2)]; % 2 hoogste pieken selecteren
powerlocs = sort(powerlocs);
afstand_tussenpieken = powerlocs(2)-powerlocs(1);
middelpunt = powerlocs(1)+(0.5*afstand_tussenpieken);
findex = fmin((middelpunt-(0.1/df)):1:(middelpunt+(0.1/df)));
lowest = min(pxx(findex)); % laagste power bepalen tussen de twee pieken
minpxx = find(pxx==lowest); % samplenummer van laagste power bepalen

Nfilt = 2;
fc = 10;
Wn = fc/(fs/2);
[B,A] = butter(Nfilt,Wn);
w_filt = filtfilt(B,A,w);
w2_filt = filtfilt(B,A,w2);
```

```
w_tot = -(w_filt+w2_filt);

%% piekdetectie
 [pks,locs,w,p] =
findpeaks(w_tot,'MinPeakDistance',0.4*fs,'MinPeakHeight',75,'MinPeakWidth',
50);
realpiek = find(w>=1 & p>=50);
pks = pks(realpiek);
locs = locs(realpiek);
%% bocht/rechtstuk
piekafstand = diff(locs);
aantal_pieken = length(locs);
locs(1,2) = 0;
for i = 2:aantal_pieken-1
       if (piekafstand(i)<=(fs/f(minpxx))) % als frequentie rechts van de
scheidingslijn van het powerspectrum ligt.
        locs(i,2) = 1; % bocht
        else
        locs(i,2) = 0; % recht stuk
       end
end

% correctie voor enkele misslag
for i = 2:aantal_pieken-1
   if locs(i,2) == 1 % bocht
       if (locs(i-1,2) == 0 && locs(i+1,2)==0) % als vorige en volgende
piek recht stuk zijn
        locs(i,2) = 0; % recht stuk
        else
        locs(i,2) = 1; % bocht
        end

    else %recht stuk
        if (locs(i-1,2) == 1 && locs(i+1,2)==1) % als vorige en volgende
piek bocht zijn
        locs(i,2) = 1; % bocht
        else
        locs(i,2) = 0; % recht stuk
       end
    end
end

%correctie voor misslagen
for i = 2:aantal_pieken-5
    if (locs(i,2) == 0 && locs(i-1,2)==1) % als huidige piek = recht en
vorige piek = bocht
       locs(i+2:i+5,2) = 0; % recht stuk minimaal 6 slagen
    end

    for j = 2:aantal_pieken-1
        if locs(j,2) == 1 %bocht
             if (locs(j-1,2) == 0 && locs(j+1,2)==0) % als vorige en
volgende piek recht stuk zijn
                 locs(j,2) = 0; % recht stuk
             else
                locs(j,2) = 1; % bocht
             end

            else %recht stuk
```

```
            if (locs(j-1,2) == 1 && locs(j+1,2)==1) % als vorige en
volgende piek bocht zijn
                locs(j,2) = 1; % bocht
            else
                locs(j,2) = 0; % recht stuk
            end
        end
    end
end
 for i = 2:aantal_pieken-1
   if (locs(i-1,2) == 1 && locs(i,2)== 0 && locs(i+1,2) == 0) % correctie
overgang bocht naar recht
        if (pks(i)<=(pks(i-1)))
            locs(i,2) = 1;
        end
    end
end
for i = 2:aantal_pieken-1
   if (locs(i-1,2) == 1 && locs(i,2)== 1 && locs(i+1,2) == 0) % correctie
overgang bocht naar recht
        if (pks(i)>=(pks(i-1)))
            locs(i,2) = 0;
        end
   end
end

bocht = find(locs(:,2) == 1);
recht = find(locs(:,2) == 0);
%% slagtijd en slagfrequentie
piektijden = (locs(:,1)./fs);
slagtijd = diff(piektijden);
for i = 2:length(piektijden)-1
    if (locs(i,2) == 0) % recht stuk
        slagtijd(i,2) = 0; % recht stuk
    else
        slagtijd(i,2) = 1; % bocht
    end
end
slagfrequentie = [60./slagtijd(:,1) slagtijd(:,2)];
figure(6)
subplot(2,1,1),plot(piektijden(1:end-1),slagtijd)% laatste piek heeft geen
slagtijd
xlabel('Tijd [s]')
ylabel('Slagtijd [s]')
title('Slagtijd')
legend('slagtijd','recht(0)/bocht(1)')
subplot(2,1,2),plot(piektijden(1:end-1),slagfrequentie(:,1)) %laatste piek
heeft geen slagfrequentie
xlabel('Tijd [s]')
ylabel('Slagfrequentie [N/min]')
title('Slagfrequentie')
hold on
bocht_slag = find(slagfrequentie(:,2) == 1);
recht_slag = find(slagfrequentie(:,2) == 0);
plot(piektijden(recht_slag,1),slagfrequentie(recht_slag,1),'.k',piektijden(
bocht_slag,1),slagfrequentie(bocht_slag,1),'.r')
xlabel('Tijd [s]')
ylabel('Slagfrequentie [N/min]')
%% rondetijd
nummer = 1; % segmentnummer
bochtrecht = diff(slagfrequentie(:,2)); % bocht = 1, recht = 0
```

```
slagfrequentie(1,3) = nummer; %1e waarde krijgt nummer 1
for i = 1:length(slagfrequentie)-1
if (bochtrecht(i)~=0)%als verschil geen 0 is : overgang van segment
    slagfrequentie(i+1,3) = nummer+1; % overgang segment
    nummer = nummer+1; % naar volgende segment
else
    slagfrequentie(i+1,3) = nummer; % slag krijgt zelfde nummer
end
end

rondetijd_index = mod(slagfrequentie(:,3),4); % segmentnummers die deelbaar
zijn door vier krijgen waarde 0
j = find(rondetijd_index == 0,1,'last'); % zoek laatste waarde die 0 heeft
rondetijden_locs(1) = 1; %startpunt van eerste ronde is eerste piektijd
index = 2; % 1e waarde is startpunt, eindpunt eerste ronde krijgt waarde 2.
for i = 4:4:slagfrequentie(j,3) % aantal rondes vaststellen (veelvoud van 4
segmenten)
rondetijden_locs(index) = find(slagfrequentie(:,3) == i,1,'last'); %zoek
laaststе punt van laatste segment van de ronde
index = index+1;
end

doorkomst = piektijden(rondetijden_locs);
rondetijden = diff(doorkomst)'; % verschil in doorkomstpunten

figure(6)
hold on
for i = 2:length(doorkomst)
plot([doorkomst(i) doorkomst(i)], [20 120],'--k');
end
legend('Slagfrequentie','recht','bocht','ronde')

ronde(1,:) = 1:length(rondetijden); % aantal rondes
ronde(2,:) = rondetijden; % bijbehorende rondetijden
ronde(3,:) = diff(rondetijden_locs); % aantal slagen per ronde
ronde(3,1) = ronde(3,1)+1; % correctie omdat 1e slag anders nergens wordt
meegenomen
%% Gemiddelde slagtijd en gemiddelde frequentie
for i = 1:max(slagfrequentie(:,3))
segment = find(slagfrequentie(:,3) == i); %zoek segmentwaardes
gem_slagfrequentie(i) = mean(slagfrequentie(segment,1)); %gem
slagfrequentie segmentwaarde
gem_slagtijd(i) = mean(slagtijd(segment,1)); %gem slagtijd segmentwaarde
aantal_slagen(i) = length(segment); % aantal slagen segment
end
%laatste slag heeft geen slagfrequentie omdat er geen volgende piek is
aantal_slagen(end) = aantal_slagen(end)+1; % laatste slag optellen bij
laatste segment

for i = 1:4 %gem segmentgegevens per ronde scheiden
    if i ==1;
    recht1_slagtijd = gem_slagtijd(i:4:end); %gem slagtijd
    recht1_slagfrequentie = gem_slagfrequentie(i:4:end); % gem frequentie
    recht1_slagen = aantal_slagen(i:4:end); % aantal slagen
    end
    if i ==2;
    bocht1_slagtijd = gem_slagtijd(i:4:end);
    bocht1_slagfrequentie = gem_slagfrequentie(i:4:end);
    bocht1_slagen = aantal_slagen(i:4:end);
    end
```

```
    if i ==3;
    recht2_slagtijd = gem_slagtijd(i:4:end);
    recht2_slagfrequentie = gem_slagfrequentie(i:4:end);
    recht2_slagen = aantal_slagen(i:4:end);
    end
    if i ==4;
    bocht2_slagtijd = gem_slagtijd(i:4:end);
    bocht2_slagfrequentie = gem_slagfrequentie(i:4:end);
    bocht2_slagen = aantal_slagen(i:4:end);
    end
end
%% Tabel

% afronden op 2 decimalen
rondetijden = round(rondetijden,2);
recht1_slagtijd = round(recht1_slagtijd,2);
bocht1_slagtijd = round(bocht1_slagtijd,2);
recht2_slagtijd = round(recht2_slagtijd,2);
bocht2_slagtijd = round(bocht2_slagtijd,2);
recht1_slagfrequentie = round(recht1_slagfrequentie,2);
bocht1_slagfrequentie = round(bocht1_slagfrequentie,2);
recht2_slagfrequentie = round(recht2_slagfrequentie,2);
bocht2_slagfrequentie = round(bocht2_slagfrequentie,2);

aantal_rondes = length(rondetijden); % aantal rondes
% tabel slagtijd
T1 =
table(rondetijden(1:aantal_rondes)',recht1_slagtijd(1:aantal_rondes)',bocht
1_slagtijd(1:aantal_rondes)',recht2_slagtijd(1:aantal_rondes)',bocht2_slagt
ijd(1:aantal_rondes)');
T1.Properties.VariableNames{1} = 'rondetijden';
T1.Properties.VariableNames{2} = 'recht1';
T1.Properties.VariableNames{3} = 'bocht1';
T1.Properties.VariableNames{4} = 'recht2';
T1.Properties.VariableNames{end} = 'bocht2';
T1
%tabel slagfrequentie
T2 =
table(rondetijden(1:aantal_rondes)',recht1_slagfrequentie(1:aantal_rondes)'
,bocht1_slagfrequentie(1:aantal_rondes)',recht2_slagfrequentie(1:aantal_ron
des)',bocht2_slagfrequentie(1:aantal_rondes)');
T2.Properties.VariableNames{1} = 'rondetijden';
T2.Properties.VariableNames{2} = 'recht1';
T2.Properties.VariableNames{3} = 'bocht1';
T2.Properties.VariableNames{4} = 'recht2';
T2.Properties.VariableNames{end} = 'bocht2';
T2
%tabel aantal slagen
T3 =
table(rondetijden(1:aantal_rondes)',recht1_slagen(1:aantal_rondes)',bocht1_
slagen(1:aantal_rondes)',recht2_slagen(1:aantal_rondes)',bocht2_slagen(1:aa
ntal_rondes)',ronde(3,:)');
T3.Properties.VariableNames{1} = 'rondetijden';
T3.Properties.VariableNames{2} = 'recht1';
T3.Properties.VariableNames{3} = 'bocht1';
T3.Properties.VariableNames{4} = 'recht2';
T3.Properties.VariableNames{5} = 'bocht2';
T3.Properties.VariableNames{end} = 'Totaal_slagen';
T3
```

9.4.3 Algoritme 3

```
clear all
close all
clc
%% info
% Author: Machiel Smits
% machiel_smits@hotmail.com
% FBW sensor met rompversnellingen
% Script voor slagdetectie specifiek voor starten.
%% Inladen Data
%romp = load('filename.mat'); % startjes

%romp signalen toekennen
accrompx = romp.data.Acceleration(:,1);
accrompy = romp.data.Acceleration(:,2);
accrompz = romp.data.Acceleration(:,3);
%% tijdas romp
fs = romp.data.sample_frequency;
dt = 1/fs;
N = length(accrompz);
k = [0:N-1];
t = k*dt;
%% z-signaal wordt gebruikt om slagen te detecteren
r = accrompz;
%% powerspectrum en filteren
Nfilt = 2;
fc = 10;
Wn = fc/(fs/2);
[B,A] = butter(Nfilt,Wn);
r_filt = filtfilt(B,A,r);
%% piekdetectie
figure(4)
[pks,locs,w,p] =
findpeaks(r_filt,'MinPeakDistance',0.2*fs,'MinPeakHeight',10);
%% Startjes eruit filteren
verschil = diff(locs);
for i = 1:length(verschil)-1
    if (verschil(i)<=200 && verschil(i+1) <=200) % startslagen zoeken
        locs(i,2) = 1; % startslag
    else
        locs(i,2) = 0; % geen startslag
    end
end
startslagen = find(locs(:,2) ==1);
onderscheiden = diff(locs(startslagen,1)); %tijd tussen startslagen
for i = 1:length(onderscheiden)
    if (onderscheiden(i)>=10000) % verschil groter dan 10000 data punten --
> nieuwe start
        startslagen(i+1,2) = 1; % eerste slag van elke start krijgt waarde
1
    else
        startslagen(i+1,2) = 0; % andere waarde 0
    end
end
startslagen(1,2) = 1; % eerste startslag krijgt ook waarde 1
Aanvang_start = find(startslagen(:,2) ==1);
locs(:,2) = 0;
locs(startslagen(Aanvang_start),2)=2; % eerste startslagen krijgen waarde 2
in kolom
```

```
for i = 1:length(locs)
    if(locs(i,2) == 2) % als waarde 2 is
        locs(i:i+15,2) = 1; % maak datapunt met waarde 2 en 16 volgende
waarde 1 = start(16 slagen)
    end
end
start = find(locs(:,2) == 1);
%% slagtijd & slagfrequentie
piektijden = (locs(:,1)./fs); %samplenummer omzetten naar secondes
slagtijd = diff(piektijden);
slagfrequentie = (60./slagtijd);

%% Slagtijd & slagfrequentie voor alleen de startjes
piektijden_start = (locs(start,1)./fs); %samplenummer omzetten naar
secondes voor alleen de startslagen

nummer = 1; % startnummer
nieuwe_start = diff(piektijden_start);
piektijden_start(1,2) = 1;
for i = 1:length(piektijden_start)-1
if (nieuwe_start(i)>=5)%als meer dan 5 sec tussen slagen zit : overgang
naar nieuwe start
    piektijden_start(i+1,2) = nummer+1; % overgang start
    nummer = nummer+1; % naar volgende start
else
    piektijden_start(i+1,2) = nummer; % slag krijgt zelfde nummer
end
end

figure(7)
for i = 1:max(piektijden_start(:,2))
    startnummer = find(piektijden_start(:,2) == i); %zoek startwaarde
    slagtijd_start= diff(piektijden_start(startnummer)); %slagtijd per
start scheiden
    slagfrequentie_start= (60./slagtijd_start); %slagfrequentie per start
berekenen

    max_i= max(piektijden_start(:,2)); % maximale waarde i =  aantal
kolommen subplots

    subplot(2,max_i,i),plot(piektijden_start(startnummer(1:end-
1),1),slagtijd_start)
    hold on
    plot(piektijden_start(startnummer(1:end-1),1),slagtijd_start,'.r') %
rode punten plotten om per slag indruk te krijgen
    hold on
    xlabel('Tijd [s]')
    ylabel('Slagtijd [s]')
    title('Slagtijd')

    subplot(2,max_i,i+max_i),plot(piektijden_start(startnummer(1:end-
1),1),slagfrequentie_start) %laatste piek heeft geen slagfrequentie
    hold on
    plot(piektijden_start(startnummer(1:end-
1),1),slagfrequentie_start,'.r') % rode punten plotten om per slag indruk
te krijgen
    hold on
    xlabel('Tijd [s]')
    ylabel('Slagfrequentie [N/min]')
    title('Slagfrequentie')
```

```
    tabel_slagtijd(:,i) = slagtijd_start; % in kolommen zetten voor tabel
    tabel_slagfrequentie(:,i) = slagfrequentie_start; % in kolommen zetten
voor tabel
end
%% Tabel met outputwaarden
tabel_slagtijd = round(tabel_slagtijd,2); % afronden op 2 decimalen
tabel_slagfrequentie = round(tabel_slagfrequentie,2); % afronden op 2
decimalen
T = table(tabel_slagtijd,tabel_slagfrequentie);
T.Properties.VariableNames{1} = 'slagtijd';
T.Properties.VariableNames{end} = 'slagfrequentie';
T
```

9.4.4 Algoritme 4

```
clear all
close all
clc
%% info
% Author: Machiel Smits
% machiel_smits@hotmail.com
% FBW sensor met schaatsversnellingen
% Script voor slagdetectie specifiek voor starten.
%% Inladen Data
%links = load('filename.mat'); % startjes
%rechts = load('filename.mat'); % startjes

%romp signalen toekennen
acclx = links.data.Acceleration(:,1);
accly = links.data.Acceleration(:,2);
acclz = links.data.Acceleration(:,3);
accrx = rechts.data.Acceleration(:,1);
accry = rechts.data.Acceleration(:,2);
accrz = rechts.data.Acceleration(:,3);
%% tijdas romp
fs = links.data.sample_frequency;
dt = 1/fs;
N = length(acclx);
k = [0:N-1];
t = k*dt;

N2 = length(accrx);
k2 = [0:N2-1];
t2 = k2*dt;
%% Totale versnelling berekenen
r = sqrt((acclx.^2)+(accly.^2)+(acclz.^2));
r2 = sqrt((accrx.^2)+(accry.^2)+(accrz.^2));
%% powerspectrum en filteren
Nfilt = 2;
fc = 10;
Wn = fc/(fs/2);
[B,A] = butter(Nfilt,Wn);
r_filt = filtfilt(B,A,r);
r2_filt = filtfilt(B,A,r2);
%% piekdetectie
[pks,locs,w,p] =
findpeaks(r_filt,'MinPeakDistance',0.2*fs,'MinPeakHeight',40);
[pks2,locs2,w2,p2] =
findpeaks(r2_filt,'MinPeakDistance',0.2*fs,'MinPeakHeight',60);

%% startslagen zoeken links
verschil_links = diff(locs);
```

```
for i = 1:length(verschil_links)-1
    if (verschil_links(i)<=300 && verschil_links(i+1) <=200) % startslagen
zoeken
        locs(i,2) = 1; % startslag
    else
        locs(i,2) = 0; % geen startslag
    end
end
startslagen_links = find(locs(:,2) ==1);
onderscheiden_links = diff(locs(startslagen_links,1)); %tijd tussen
startslagen
for i = 1:length(onderscheiden_links)
    if (onderscheiden_links(i)>=10000) % verschil groter dan 1000 data
punten --> nieuwe start
        startslagen_links(i+1,2) = 1; % eerste slag van elke start krijgt
waarde 1
    else
        startslagen_links(i+1,2) = 0; % andere waarde 0
    end
end
%% startslagen zoeken rechts
verschil_rechts = diff(locs2);
for i = 1:length(verschil_rechts)-1
    if (verschil_rechts(i)<=300 && verschil_rechts(i+1) <=200) %
startslagen zoeken
        locs2(i,2) = 1; % startslag
    else
        locs2(i,2) = 0; % geen startslag
    end
end
startslagen_rechts = find(locs2(:,2) ==1);
onderscheiden_rechts = diff(locs2(startslagen_rechts,1)); %tijd tussen
startslagen
for i = 1:length(onderscheiden_rechts)
    if (onderscheiden_rechts(i)>=10000) % verschil groter dan 1000 data
punten --> nieuwe start
        startslagen_rechts(i+1,2) = 1; % eerste slag van elke start krijgt
waarde 1
    else
        startslagen_rechts(i+1,2) = 0; % andere waarde 0
    end
end
%% Eerste startslagen per been zoeken
startslagen_links(1,2) = 1; % eerste startslag krijgt ook waarde 1
Aanvang_start_links = find(startslagen_links(:,2) ==1);
locs(:,2) = 0;
locs(startslagen_links(Aanvang_start_links),2)=2; % eerste startslagen
krijgen waarde 2 in kolom

startslagen_rechts(1,2) = 1; % eerste startslag krijgt ook waarde 1
Aanvang_start_rechts = find(startslagen_rechts(:,2) ==1);
locs2(:,2) = 0;
locs2(startslagen_rechts(Aanvang_start_rechts),2)=2; % eerste startslagen
krijgen waarde 2 in kolom

%% Bepalen welk been begint
beginslagen_links = find(locs(:,2)==2); %beginslagen links vinden
beginslagen_rechts = find(locs2(:,2)==2); % beginslagen rechts vinden
piekhoogte_links = pks(beginslagen_links); % piekhoogtes beginslagen links
piekhoogte_rechts = pks2(beginslagen_rechts); %piekhoogtes beginslagen
rechts
```

```
for i = 1:length(piekhoogte_links)
    if piekhoogte_links(i)<=piekhoogte_rechts(i) % als linkerpiek hoger is
dan rechter
        locs(startslagen_links(Aanvang_start_links(i)),2)=3; % 1e startslag
is links
    else
        locs2(startslagen_rechts(Aanvang_start_rechts(i)),2)=3; %1e
startslag is rechts
    end
end
for i = 1:length(locs)
    if(locs(i,2) == 3) % als linkerbeen 1e slag is
        locs(i:2:i+8,2) = 1; % Geef 1e piek en 8 volgende oneven pieken de
waarde 1;
    else if (locs(i,2) == 2) % als rechterbeen 1e slag is
            locs(i+1:2:i+7,2) = 1; % Geef 2e piek en 7 volgende even pieken
de waarde 1;
        end
    end
end
for i = 1:length(locs2)
    if(locs2(i,2) == 3) % als rechterbeen 1e slag is
        locs2(i:2:i+8,2) = 1; % Geef 1e piek en 8 volgende oneven pieken de
waarde 1;
    else if (locs2(i,2) == 2) % als linkerbeen 1e slag is
            locs2(i+1:2:i+7,2) = 1; % Geef 2e piek en 7 volgende even
pieken de waarde 1;
        end
    end
end
%% slagtijd & slagfrequentie
locs_totaal = [locs(start_links);locs2(start_rechts)];
locs_totaal = sort(locs_totaal);
piektijden_start = (locs_totaal./fs); %samplenummer omzetten naar secondes
voor alleen de startslagen

nummer = 1; % startnummer
nieuwe_start = diff(piektijden_start);
piektijden_start(1,2) = 1;
for i = 1:length(piektijden_start)-1
if (nieuwe_start(i)>=5)%als meer dan 5 sec tussen slagen zit : overgang
naar nieuwe start
    piektijden_start(i+1,2) = nummer+1; % overgang start
    nummer = nummer+1; % naar volgende start
else
    piektijden_start(i+1,2) = nummer; % slag krijgt zelfde nummer
end
end

figure(6)
hold on
for i = 1:max(piektijden_start(:,2))
startnummer = find(piektijden_start(:,2) == i); %zoek startwaarde
slagtijd_start= diff(piektijden_start(startnummer)); %slagtijd per start
scheiden
slagfrequentie_start= (60./slagtijd_start); %slagfrequentie per start
berekenen

max_i= max(piektijden_start(:,2)); % maximale waarde i =  aantal kolommen
subplots
```

```
subplot(2,max_i,i),plot(piektijden_start(startnummer(1:end-
1),1),slagtijd_start)
hold on
plot(piektijden_start(startnummer(1:end-1),1),slagtijd_start,'.r') % rode
punten plotten om per slag indruk te krijgen
xlabel('Tijd [s]')
ylabel('Slagtijd [s]')
title('Slagtijd')

subplot(2,max_i,i+max_i),plot(piektijden_start(startnummer(1:end-
1),1),slagfrequentie_start) %laatste piek heeft geen slagfrequentie
hold on
plot(piektijden_start(startnummer(1:end-1),1),slagfrequentie_start,'.r') %
rode punten plotten om per slag indruk te krijgen
xlabel('Tijd [s]')
ylabel('Slagfrequentie [N/min]')
title('Slagfrequentie')
tabel_slagtijd(:,i) = slagtijd_start; % in kolommen zetten voor tabel
tabel_slagfrequentie(:,i) = slagfrequentie_start; % in kolommen zetten voor
tabel
end
%% Tabel met outputwaarden
tabel_slagtijd = round(tabel_slagtijd,2); % afronden op 2 decimalen
tabel_slagfrequentie = round(tabel_slagfrequentie,2); % afronden op 2
decimalen
T = table(tabel_slagtijd,tabel_slagfrequentie);
T.Properties.VariableNames{1} = 'slagtijd';
T.Properties.VariableNames{end} = 'slagfrequentie';
T
```

9.5 Bijlage 5

Bijlage 2B

Document projectplan voor een ontwerpproject.

→uiterste inleverdatum maandag week 10 blok 2 aan alle leden van de commissie van de projectpresentaties (geldt alleen voor reguliere fase); bij andere planning ;3 werkdagen voor voorstel presentatie voor 1200u)

Naam: Machiel Smits

Studentnummer:11067659

e-mail: machiel_smits@hotmail.com

Behaalde studiepunten in de modules 9 t/m 12:33

Datum: 6-3-2015

1. Onderwerp (voorlopige titel): Slagdetectie bij het schaatsen

Werkveld: sport

Beroepsrol: programmeur

Extern project (J/N): J

Indien Extern:

.. naam Opdrachtgever/bedrijf/ECBT: VU Amsterdam

.. contactpersoon (naam en mailadres): Roy Mulder, r.c.m.mulder@vu.nl

2. Probleemstelling

Aanleiding :

Skate & Science is opgericht door wetenschappers van de Vrije Universiteit Amsterdam en de Technische Universiteit Delft. Samen voeren zij een vierjarig schaatswetenschappelijk onderzoeksproject uit. 'Real-time feedback voor een betere schaatsprestatie' is een project dat loopt tot 1 september 2016. Het doel van dit project is het verbeteren van de individuele schaatstechniek door middel van real-time feedback over de afzet tijdens het schaatsen.
Roy Mulder, PhD student, is begonnen met het detecteren van schaatsslagen aan de hand van data uit een FBW sensor (accelerometer+ gyroscoop). De FBW sensor wordt bevestigd op de schaatsschoen. Hij heeft een eenvoudige MATLAB GUI geschreven dat slagen kan detecteren van gyroscoop data. Een databestand wordt ingeladen. Daarna worden de signalen gefilterd en vervolgens gaat het programma opzoek naar pieken (slagen). Uit de gedetecteerde slagen kan tot slot de slagtijd en slagfrequentie bepaald worden.
Schaatscoaches zien slagdetectie wel zitten voor het analyseren van de start, maar dan moet de applicatie/software meer verfijnd en gebruiksvriendelijker worden. Daarnaast maken ze in Thialf

gebruik van het Zephyr systeem (o.a. accelerometer). Het Zephyr systeem bevat geen gyroscoop en wordt op de romp gedragen. Het programma zal aangepast moeten worden om ook pieken van accelerometer data te kunnen detecteren.

Doelgroep: Coaches en topschaatsers.

Doelstelling: Het schrijven van een MATLAB GUI die slagen kan detecteren aan de hand van data uit accelerometers op de schaats en romp.

Randvoorwaarden: Er dient beschikking te zijn over de huidige MATLAB GUI en de MATLAB versie waarmee de GUI is geschreven. Tevens is er data van zowel de gyroscoop als de accelerometer van de FBW sensor nodig. Hiermee kan gecontroleerd worden of de GUI zal werken voor beide type data. Daarnaast is er data nodig van een accelerometer die op de romp is gedragen. Verder moet er contact gelegd worden met coaches om het programma zo goed mogelijk aan hun wensen te laten voldoen. Aart van der Wulp, Innosportlab manager Thialf en trainer, is benaderd of hij input wil geven. Tevens staat hij in contact met meerdere trainers/coaches. Deze kunnen eventueel om input gevraagd worden via een vragenlijst. Daarnaast is er meer kennis nodig over de schaatsbeweging en de werking van de gebruikte accelerometers om de data te kunnen analyseren. Er moet achterhaald worden op welke punten in het signaal de verschillende bewegingen van het schaatsen zijn terug te zien (contactmoment, afzet, zwaaifase, etc.).
In de ideale situatie dient er kennis te zijn over methodes om MATLAB te gebruiken als een real-time feedback systeem.
Binnen de beschikbare tijd van 16 weken is het niet realistisch om een geheel werkend real-time feedback systeem te produceren. Het zal een proces worden van trial-and-error om het programma steeds beter te maken. De eerste doelstelling is het schrijven van een GUI die slagen kan detecteren aan de hand van data uit de accelerometers van de FBW sensoren. Deze doelstelling moet haalbaar zijn, aangezien de slagdetectie op een soortgelijke wijze zal werken als de slagdetectie met gyroscoop data. Pas na het bereiken van de eerste doelstelling kan aandacht worden besteed aan de accelerometers op de romp, wensen en de gebruiksvriendelijkheid van het programma.

3. Vooronderzoek

Allereerst is het van belang om de werking van het huidige programma te begrijpen. Bovendien is het programma geschreven op een computer met een Apple-besturingssyteem. Er zal in dit project gewerkt worden met een Windows-besturingssysteem. Gezien het verschil in mappenindeling kan het zijn dat het huidige programma al enigszins aangepast moet worden om het werkend te krijgen op een windows pc.
Verder zal er uit literatuurstudie moeten blijken of het mogelijk is om een real-time feedback systeem te creëren vanuit een MATLAB programma met data uit een accelerometer. Deze studie zal belangrijke gevolgen hebben voor de doelstelling van het project. MATLAB is in het bezit van een functie om grafieken real-time te plotten. Tevens is het mogelijk om met MATLAB data via een USB poort continu op te halen en vervolgens real-time te plotten (Manalo & Ashrafi, 2012).
Accelerometer data is in een eerdere studie bewerkt met MATLAB en vervolgens real-time afgebeeld met LabVIEW (Crowell et al., 2010).

Op 9 februari vond een meeting plaats in Thialf met Aart van der Wulp en embedded scientist Wouter van der Ploeg. Uit deze meeting zijn een aantal belangrijke punten naar voren gekomen. Deze punten zijn samengevat in de bijlage.

4. Analyse

1. Hoe ziet de schaatsbeweging eruit?

Er zal literatuuronderzoek moeten plaastvinden om deze vraag te beantwoorden. Tevens kunnen Roy Mulder en andere onderzoekers van het project belangrijke contacten zijn. Zij doen immers al een tijd onderzoek naar het schaatsen.

2.Hoe werken gyroscopen?

Het huidige programma kan slagen detecteren van gyroscoop data. De gebruikte gyroscopen zijn geproduceerd door de technische dienst van de VU. Deze vraag kan beantwoord worden door middel van zelfstudie. Daarnaast kan Roy Mulder een nuttige bron zijn. Hij heeft namelijk testen uitgevoerd met de gyroscopen.

3. Hoe werkt het huidige programma?

Dit zal vooral duidelijk worden door eigen kennis. De programmeur, Roy Mulder, is hier opnieuw een belangrijke bron. Daarnaast kan de cursus 'Verwerken van digital signalen' van Theo de Haan ondersteuning bieden. Verder kan de help functie van MATLAB nuttig zijn. Tot slot kan er ook contact worden opgenomen met verscheidene docenten.

4. Is MATLAB te gebruiken als real-time feedback systeem?

Het antwoord zal voortkomen uit literatuurstudie. Roy Mulder en collega's van de TU Delft hebben betere toegang tot PubMed en andere databases. Via hen is het mogelijk om meer artikelen full-text te verkrijgen. Als het antwoord 'nee' is, zal er een programma gezocht moeten worden die een MATLAB programma kan aansturen.

5. Aan welke eisen moet het programma voldoen?

De eisen zullen voortkomen uit de opdracht, de tekortkomingen van het huidige programma en vanuit de embedded scientists/coaches. Dit zal gebeuren aan de hand van zelfstudie en interviews/vragenlijsten. Daarnaast heeft Roy Mulder al een aantal punten aangegeven.

6. Hoe werken 3-assige accelerometers?

Voor het beantwoorden van deze vraag zal dezelfde methode gebruikt worden als bij vraag 2.

7. Welke factoren van de schaatsbeweging zien we terug in het signaal van de accelerometers op de schaats en romp?

Aan de hand van het huidige programma zal hier veel duidelijk over worden. Daarnaast kunnen de antwoorden van vraag 6 gebuikt worden. Tot slot kan literatuurstudie van de vorige vraag eventueel uitgebreid worden om een uitkomst te bieden.

8. Hoe kan MATLAB pieken van accelerometers detecteren?

De MATLAB help functie zal geraadpleegd worden om de nodige functies te achterhalen. Verder zullen er veel overeenkomsten zijn met de slagdetectie van gyroscopen in het huidige programma. Daarnaast kan de cursus 'Verwerken van digitale signalen' van Theo de Haan een bron zijn voor functies en filtering.

9. Wat zijn de wensen van de gebruiker?

Embedded scientists/coaches zullen hier doorslaggevend zijn. De wensen zullen duidelijk worden aan de hand van interviews of vragenlijsten.

10. Welke functies kunnen het programma gebruiksvriendelijker maken?

Roy Mulder heeft al een aantal punten genoemd. Er is momenteel geen help functie. Daarnaast werkt de knipfunctie nog niet optimaal. Verder kan er op dit gebied telkens vooruitgang worden geboekt door middel van feedback van derden. Ten slotte zullen een hoop knelpunten al gedurende het proces duidelijk worden door eigen ervaring met het programma.

De twee onderstaande deelvragen zijn alleen van toepassing als het zo ver mocht komen gedurende het project.

11. Hoe kan het programma gerund worden vanaf een opslagmedium?

ICT-specialisten met MATLAB ervaring zijn nodig om dit mogelijk te maken. Literatuurstudie en MATLAB tutorials kunnen hier belangrijke eerste stappen zijn.

12. Hoe kan het programma omgebouwd worden tot real-time feedback systeem?
De literatuurstudie van vraag 4 zal hiervoor uitgebreid moeten worden. Verder zal de hulp van onderzoekers/technici nodig zijn.

5. Ontwerpfase

Er hoeft geen compleet nieuwe GUI ontworpen te worden, aangezien Roy Mulder al een werkende GUI heeft geschreven. Het betreft hier een aanpassing. De GUI blijft eigendom van Roy Mulder en van Skate & Science (gesponsord door STW). De layout van de GUI zal daarom zoveel mogelijk gehandhaafd worden. Echter zullen uit de analyse een aantal eisen en wensen volgen. Deze eisen en wensen zullen resulteren in nieuwe functies in het programma. Door het toevoegen van nieuwe functies zal de layout van de GUI veranderen. Roy Mulder zal wederom een belangrijk contact zijn in deze fase. Verder zal de input/feedback van eventuele gebruikers (coaches) bepalend zijn.

6. Vervaardigingfase

Voor het vervaardigen van een real-time feedbacksysteem zijn meer onderdelen nodig dan alleen een GUI. De GUI zal gemaakt worden met MATLAB. Naast de GUI zal een programma nodig zijn die het real-time feedbacksysteem aanstuurt. Dit is mogelijk met een programma als LabVIEW. Daarnaast zal de feedback via een nog te ontwerpen bril aan de schaatser worden doorgegeven. Voor het vervaardigen van de GUI zal vooral contact gelegd worden met Roy Mulder en eventueel Aart van der Wulp. Voor het vervaardigen van het real-time feedbacksysteem zal contact nodig zijn met andere onderzoekers die werken aan andere deelprojecten van het systeem.

7. Testfase / Evaluatiefase

De MATLAB GUI kan pas data analyseren als deze uit de accelerometer is gelezen. Het is daarom niet gelijk te zien of het programma alle schaatsslagen detecteert. Natuurlijk kan wel het aantal slagen van een proefpersoon geteld worden en dit aantal is vervolgens te vergelijken met het aantal slagen dat het programma detecteerde. Daarnaast kan er een video worden gemaakt en deze moet dan gesynchroniseerd worden met de accelerometer data. De gebruiksvriendelijkheid en de werking van de GUI kan getest worden via eigen gebruik en door derden(coaches) het programma te laten gebruiken. Het programma kan echter pas echt goed geëvalueerd worden als de data real-time zichtbaar is. Er kan dan meteen geconcludeerd worden of het programma een slag detecteert.

8. Voorlopige literatuurlijst

1. Manalo, M.S., & Ashrafi, A. (2012). USB Interfacing and Real Time Data Plotting with MATLAB.

2. Crowell, H.P., Milner, C.E., Hamill, J., & Davis, I.S. (2010). Reducing impact loading during running with the use of real-time visual feedback. J Orthop Sports Phys Ther, 40(4):206-13.

9. Planning

Activiteit/deadlines	Data
Inleveren concept projectpan	Week 51
Aanpassen concept projectplan	Week 52 – week 2
Aanmelden projectplan presentatie	Uiterlijk vrijdag week 3
Inleveren definitief projectplan	Uiterlijk maandag week 6
Presentatie projectplan	Donderdag week 6
Start vooronderzoek	Week 7
Vooronderzoek (literatuurstudie/interviews)	Week 7 – week 10
Start uitvoeringsfase	Maandag week 11
Literatuurstudie Analysefase	Week 11– week13
Werkende Matlab GUI schrijven	Week 14– week 18
Aanpassen aan wensen + gebruiksvriendelijker	Week 19– week 21
Scriptie schrijven + evaluatie	Week 22 – woensdag week 25
Inleveren definitief afstudeerwerk	Woensdag week 25 voor 13.00 uur
Periode Eindgesprekken	Vrijdag week 26 – vrijdag week 27

Bovenstaande tabel geeft de globale planning weer. De benodigde tijd voor het programmeren hangt heel erg af van de vorderingen. Het is een proces van trial-and-error. De planning voor het maken van een real-time feedback systeem is niet in de planning opgenomen aangezien pas gedurende de uitvoerende fase duidelijk wordt of hier tijd voor is.

Voorlopige hoofdstukindeling scriptie:

1. Samenvatting
2. Inleiding
3. Analyse
   - Literatuuronderzoek
   - Werking en functies huidige MATLAB GUI
   - Gebreken en mogelijke toevoegingen
   - Lijst van eisen en wensen

4. Ontwerp nieuwe MATLAB GUI
5. Werking en functies nieuwe MATLAB GUI
6. Evaluatie MATLAB GUI
7. Discussie
8. Conclusie
9. Bijlage

10. Persoonlijke leerdoelen afstudeerfase (minimaal 3)

Leerdoel 1: Ik kan na het afstuderen signalen filteren en verwerken met Matlab tot een daadwerkelijk resultaat.

Tot nu toe werd er in de Matlabopdrachten altijd verteld wat voor soort filter je moest toepassen. Daarnaast werd er een signaal gegeven dat meestal kant en klaar was voor verwerking. Ik hoop na deze periode en na het volgen van het vak 'Meten van fysische grootheden' meer te weten over de totstandkoming van een signaal en het verwerken van dit signaal tot een resultaat.

Leerdoel2: Na het afstuderen kan ik een goede literatuurstudie uitvoeren.

Ik heb slechts twee keer eerder literatuurstudie gedaan. Het ging hier om kleinere projecten waar vaak maar een paar artikelen nodig waren. De literatuurstudie is in dit geval zeer belangrijk voor het vervolg van het project. Dit leerdoel is gelijk na de analysefase te evalueren.

Leerdoel3: Ik kan na de afstudeerperiode mijn werk beter evalueren.

Gedurende mijn studie heb ik aan verscheidene ontwerpprojecten gewerkt. Tijdens deze projecten waren de middelen naar mijn mening altijd zeer gering. Er ontstond bijna nooit een product dat echt gebruikt gaat worden door mensen. Met mijn vierdejaarsstage en afstudeerproject heb ik echter wel het budget/de middelen om iets te produceren waar mensen later echt wat mee kunnen. Ik hoop na het afstuderen te kunnen evalueren of mijn werk nuttig is geweest.

11 Bijlage

Samenvatting meeting 9 februari Thialf

Aart van der Wulp, Innosportlab manager
Wouter van der ploeg, Embedded scientist

Zephyr:

- Wordt vooral veel bij het shorttrack gebruikt. Langebaanschaatsen nauwelijks.
- Grote uitdaging om detectie in bocht eruit te halen (schaatser bijna in 1 lijn).
- Wel mooi als het werkt met zephyr systeem, maar focussen op FBW sensor is beter.

FBW Sensor:

- Kijken waar je bevestigd. Kastjes achter schaats worden er soms afgereden bij het bijtrekken.
- Eventueel overleggen met materiaalfabrikanten om ruimte hiervoor in te bouwen. Schaatsers vinden add-ons meestal in de weg zitten.

- Sensoren op beide voeten gesynchroniseerd bij huidige data? (soms slagfrequentie van 300+ bij combinatie van links en rechts).

Coaches:

- Koppelen aan rondetijden. Slagen per minuut zegt hen niks.
- Beginnen met slagfrequentie, daarna zal een coach zelf benaderen als ze meer willen weten.
- Na elke recht eind, bocht, rondje aantal slagen + frequentie.
- Slagfrequentie koppelen aan rondetijden
- Denken in twee benen. Getallen weergeven voor links en rechts gecombineerd.

Data weergeven:

- Live(real-time) is een pre. Achteraf analyseren doen coaches nauwelijks. Echter zal wel begonnen moeten worden met analyse achteraf.
- Slagfrequentie koppelen aan snelheid en rondetijd.
- Per ronde 4x terugkoppeling van gemiddelde frequentie/snelheid/slagtijd: rech-boch-recht-bocht.
- Grafieken van data (versnelling/hoeksnelheden) is vooral interessant voor onderzoeker.
- Grafieken van slagfrequentie/snelheid weergeven voor coaches.

Toekomstig contact:

- Vorderingen (plaatjes) mailen naar Wouter en Aart.
- Wouter woont vlakbij de vu en kan langskomen om te overleggen.
- Coaches kunnen later benaderd worden via het innosportlab als het programma werkt.

Komende acties:

- Nieuwe testen met gesynchroniseerde sensoren en rondetijden klokken.
- Kijken of uit het signaal een rondetijd komt die in de buurt komt van de geklokte tijd (hoeft niet nauwekeurig op tiende, maar wel het juiste aantal secondes.
- Slagdetectie werkend krijgen voor accelerometers
- Slagdetectie evalueren met video voor zowel gyroscoop als accelerometer.
- Programma bochten en rechte stukken laten herkennen.
- Koppeling maken tussen frequentie en rondetijd en eventueel frequentie en snelheid.