

Bewegingsherkenning met een smartphone

Arne De Brabandere

arne.debrabandere@student.kuleuven.be

Menno Keustermans

menno.keustermans@student.kuleuven.be

Abstract

Het herkennen van menselijke bewegingen is een belangrijk onderdeel van ‘context aware computing’. Uit de metingen van de accelerometer en gyrocoop van een smartphone kan de activiteit van de gebruiker bepaald worden. In het eerste deel van ons onderzoek zoeken we naar modellen om afzonderlijke activiteiten te herkennen. Het nauwkeurigste model wordt gevonden met de classificatiemethode Random Forests met een accuraatheid van 93%. Vervolgens gebruiken we dit model om een sequentie van verschillende bewegingen te evalueren. We ontwikkelen hiervoor een algoritme dat de sequentie in opeenvolgende tijdsvensters van een aantal seconden knipt, die ook met een bepaald percentage kunnen overlappen. Uit experimenten blijkt dat hiervoor een tijdsvenstergrootte van vier tot zes seconden met een overlapping van 75% het nauwkeurigst is met 84% accuraatheid.

1 Introductie

Door de versnellingen en rotaties van menselijke bewegingen te meten, is het mogelijk om automatisch te herkennen welke activiteiten uitgevoerd worden. We maken hier een onderscheid tussen metingen van afzonderlijke activiteiten (één activiteit per meting) en sequenties van activiteiten (verschillende activiteiten na elkaar in een meting). We gaan voor beide soorten metingen op zoek naar een methode om de activiteiten te herkennen.

Het nauwkeurig detecteren van menselijke bewegingen is een belangrijke doelstelling van ‘context aware computing’. Hierbij wordt software gekoppeld aan de gebruiker, zodat die bewust is van de omgeving en toestand van de gebruiker. Toepassingen hiervan bevinden zich bijvoorbeeld in e-health applicaties. Voor dit onderzoek werd gebruik gemaakt van een smartphone als bewegingssensor. Dit levert verschillende voordelen op. Het zijn namelijk populaire apparaten de dag van vandaag, die bovendien bruikbare ingebouwde sensoren (accelerometer en gyrocoop) hebben.

In deze paper wordt een oplossing gezocht voor twee probleemstellingen: (1) Welke classificatiemethode levert het nauwkeurigste model op om afzonderlijke activiteiten te her-

kennen? (2) Kunnen we dit model gebruiken om sequenties van verschillende activiteiten nauwkeurig te voorspellen?

De eerste probleemstelling wordt besproken in sectie 2. De oplossing verloopt in drie stappen (zie figuur 1 voor een overzicht). In de eerste stap worden gegevens verzameld (sectie 2.1). Vervolgens worden uit de gegevens features berekend (sectie 2.2). Die gebruiken we ten slotte om modellen te leren met behulp van classificatiemethodes (sectie 2.3).

Om een oplossing te vinden voor de tweede probleemstelling wordt het beste model van het eerste deel van het onderzoek gebruikt. De stappen van het proces om sequenties van activiteiten te herkennen verlopen in gelijkaardige stappen als die van het proces voor afzonderlijke activiteiten (zie figuur 5 voor een overzicht). Ten eerste moeten opnieuw gegevens verzameld worden (sectie 3.1). Vervolgens worden deze gegevens verwerkt (sectie 3.2). Om tenslotte de activiteiten van de sequenties te voorspellen, wordt gebruik gemaakt van een algoritme met drie parameters (sectie 3.3).

Gerelateerd werk

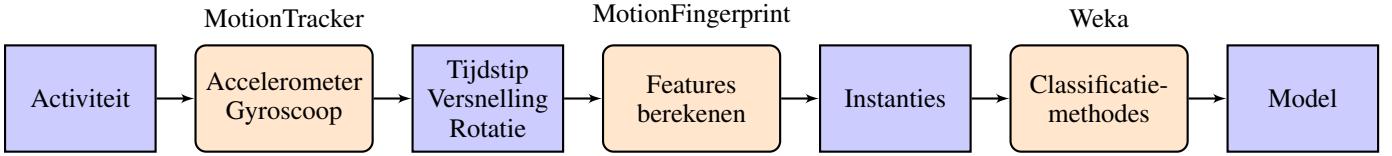
Zoals eerder vermeld, is bewegingsherkenning een belangrijk concept in ‘context aware computing’. Hierdoor is er al veel onderzoek naar gedaan.

Zo werd er in de paper van Ravi, Dandekar, Mysore en Littman onderzocht naar herkenning van afzonderlijke activiteiten met behulp van gegevens opgemeten door een externe accelerometer [Ravi2005]. Zij maakten gebruik van meta-level classifiers om modellen te genereren.

Ook in andere papers werd er gebruik gemaakt van externe accelerometers, maar zij verschillen dan in het gebruik van classifiers. Zo hebben Chen, Alregib en Juang HMM classifiers onderzocht [Chen2012].

In ons onderzoek worden gegevens opgemeten waarbij de meetapparatuur zich bevindt rondom het bekken. Casale, Pujoal en Radeva deden gegevensopmetingen met een accelerometer rondom de borstzone en vonden een nauwkeurig model met behulp van Random Forests [Casale].

De meeste van deze papers hebben zich gefocust op het herkennen van afzonderlijke activiteiten. Deze paper focust zich naast de afzonderlijke activiteiten ook op het herkennen van sequenties van activiteiten.



Figuur 1: Proces voor zoeken van een model om afzonderlijke activiteiten te herkennen

2 Afzonderlijke activiteiten

Het eerste probleem is om van een gegeven reeks samples van de accelerometer en gyroscoop van een smartphone de activiteit van een persoon te bepalen. Elke sample bestaat uit een reeks opeenvolgende tijdstippen met bijhorende x-, y- en z-versnellingen en rotatiesnelheden die werden opgemeten. We veronderstellen hier dat telkens één afzonderlijke activiteit gemeten wordt.

We willen tien verschillende activiteiten kunnen herkennen:

- wandelen,
- lopen,
- fietsen,
- een trap opwandelen,
- een trap afwandelen,
- springen,
- niets doen (zitten, liggen, staan),
- een lift versnelt omhoog,
- een lift versnelt omlaag,
- tanden poetsen.

In bovenstaande lijst hebben we tanden poetsen als moeilijke activiteit toegevoegd. De beweging lijkt sterk op niets doen en zal waarschijnlijk minder goed te herkennen zijn. De x-, y- en z-versnellingen van de verschillende activiteiten worden weergeven in figuur 8 (zie bijlage).

Merk op dat we in plaats van de activiteiten ‘lift naar boven of beneden nemen’ enkel de versnelling van een lift naar boven of beneden in de lijst opnemen. Een lift naar boven nemen bevat dus de volgende activiteiten: lift versnelt omhoog – niets doen – lift versnelt omlaag. Het feit dat deze activiteiten ook ‘niets doen’ bevatten, is de verklaring waarom we alleen de versnelling van een lift herkennen.

Het proces om de afzonderlijke activiteiten te herkennen verloopt in drie stappen (zie figuur 1). De eerste stap bestaat uit het verzamelen van de gegevens. Hiervoor meten we de versnellingen en rotaties van de verschillende activiteiten. Als tweede stap worden features berekend uit de verzamelde gegevens. Deze zijn nodig om in de laatste stap modellen te leren met behulp van classificatiemethodes. Als criterium om de verschillende modellen te vergelijken, gebruiken we de accuraatheid als percentage van het aantal juist geclasseerde samples ten op zichte van het totaal aantal metingen met behulp van cross-validatie.

2.1 Gegevensverzameling

Alle gegevens werden opgemeten door de MotionTracker tool [MT]. Dit is een Android-applicatie die de versnelling en rotatie (respectievelijk gemeten door de accelerometer en gyroscoop van de smartphone) meet aan 50Hz. Als uitvoer geeft de applicatie een logbestand met de gemeten versnellingen (in de x-as, y-as en z-as; de z-as is evenwijdig met de gravitatie) en rotaties (in quaternion notatie) met bijhorende timestamps.

Voor elke meting werd de applicatie gestart alvorens de smartphone in de broekzak gestoken werd en gestopt na het uithalen. Daarom bevat het begin en einde van elke meting enkele seconden die niet tot de gemeten activiteit horen. Om het fout labelen te vermijden, werd na elke meting een stuk van de start en het einde van het logbestand weggesneden, zodat elke meting exact één activiteit bevat van vier tot twintig seconden lang.

Op deze manier hebben we voor elke activiteit 22 metingen verzameld, opgemeten door twee verschillende personen. Om voldoende variatie te hebben, gebeurden de metingen op verschillende dagen. Ook hebben we ervoor gezorgd dat we niet telkens dezelfde broek droegen, aangezien de gemeten versnelling kan variëren in verschillende broekzakken. Na elke meting werd het resulterende logbestand geknipt en gelabeld met de juiste activiteit.

2.2 Gegevensverwerking: features berekenen

Voor we classificatiemethodes kunnen gebruiken, moeten we eerst features berekenen. Dit zijn parameters die we uit de samples van de accelerometer en gyroscoop kunnen halen. Om de verschillende features te berekenen, maken we gebruik van de MotionFingerprint tool [MFP].

MotionFingerprint berekent in totaal 134 features¹ verdeeld onder vier soorten:

- *Statistische features*: gemiddelde en standaardafwijking van z- en xy-versnelling en vermogen, correlatie tussen z- en xy-versnelling, etc.
- *Fourier transformatie*: amplitudes horende bij bepaalde frequenties, pieken, etc.
- *Wavelet transformatie*: gemiddelde van de coëfficiënten per schaal, energie per schaal, etc.
- *Hidden Markov Models*: log-likelihood voor de verschillende HMM modellen. Voor elke activiteit wordt een model berekend uit zes metingen die apart werden gehouden. (Daarom worden ook in de rest van de paper slechts 16 van de 22 metingen per activiteit gebruikt.)

¹Het aantal features is afhankelijk van de instellingen van de tool.

Na het berekenen van de features hebben we voor elke meting een set van parameters. Elke set vormt een instantie van de training set om een model uit te leren.

Feature selectie

Tussen de verschillende soorten features is er verschil in berekeningstijd en hoeveelheid informatie. Zo kunnen statistische features in het algemeen tegen een lagere kost berekend worden ten op zichte van Fourier transformatie features. In sommige toepassingen (zoals smartphone applicaties) kan het daarom interessant zijn om slechts een deel van het totaal aantal features te berekenen en juist diegene die de meeste informatie bevatten.

Met behulp van feature selectie onderzoeken we hoeveel features van het totaal aantal features van elke soort nodig zijn om een redelijke accuraatheid te behalen. Om de features met de meeste informatie te vinden, maken we gebruik van Weka's *InfoGainAttributeEval* klasse². Deze klasse evalueert de waarde van elke feature met behulp van information gain.

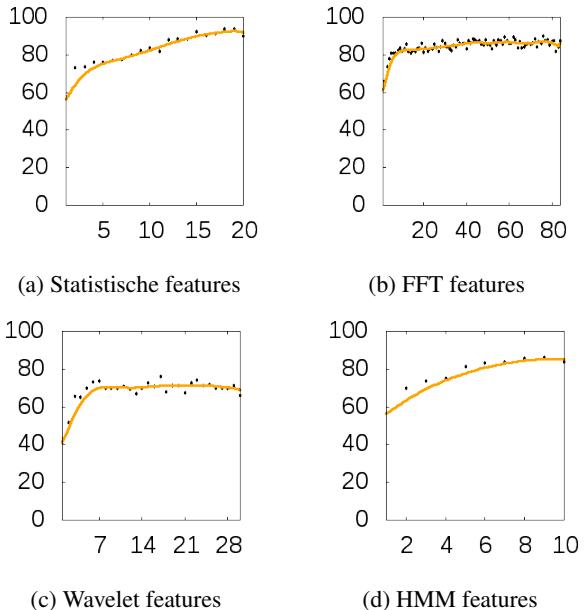
Om vervolgens de geselecteerde features te evalueren maken we gebruik van dubbele cross-validatie. Bij de eerste cross-validatie (10-fold) wordt de gegevensset telkens verdeeld in een training en test set. Voor elke training set selecteren we de beste features met de methode die hierboven werd beschreven, gebruik makend van een tweede cross-validatie (2-fold). Met de geselecteerde features wordt een model opgesteld met behulp van de Random Forests classificatiemethode in Weka. Het gevonden model wordt uiteindelijk geëvalueerd op de test set van de eerste cross-validatie.

In figuur 2 worden de resultaten voor de verschillende soorten features getoond. We zien dat we met enkel de statistische features al een accuraatheid van 90% behalen. Zelfs met de helft van de features kan het model al met een accuraatheid van 80% activiteiten voorspellen. We stellen vast dat van het grote aantal Fourier transformatie features slechts een tiental nodig zijn om een nauwkeurigheid van 80% te bekomen. De features die hierbij geselecteerd worden zijn voornamelijk die voor lagere frequenties (< 2.5 Hz) volgens het xy-vlak. Wavelet transformatie features lijken minder goed te werken met slechts 70% accuraatheid. Tenslotte merken we in de laatste grafiek op dat de Hidden Markov Models enkel degelijk werken wanneer alle features gebruikt worden.

Voor de rest van het onderzoek wordt gebruik gemaakt van de volledige set features.

2.3 Classificatiemethodes

We gebruiken classificatiemethodes om een model te zoeken om afzonderlijke activiteiten te herkennen. We vergelijken enkele veel voorkomende methodes: beslissingsbomen, Random Forests, k-Nearest Neighbours, Naive Bayes en Support Vector Machines. De methodes leren telkens een model uit de instanties met de hierboven beschreven features voor de verschillende activiteiten. Hieronder wordt kort de werking van de vernoemde methodes beschreven:



Figuur 2: Resultaten van het feature selectie experiment: op de x-as wordt steeds het aantal features van elke soort features geplot en op de y-as de accuraatheid (in percentages) van het model dat door Random Forests geleerd werd.

- *Beslissingsbomen* zijn bomen waarvan de interne knopen features voorstellen en de bladknopen één (of meerdere) labels bevatten. Een tak in de boom stelt een test voor op de feature van de knoop waaruit de tak vertrekt. Om een nieuwe instantie te classificeren worden de juiste takken gevuld tot in een bepaalde bladknop. De instantie wordt dan geklassificeerd als het label in die bladknop.
- *Random Forests* zijn combinaties van beslissingsbomen met een willekeurig deel van de trainingset en een willekeurig deel van de features. Bij de classificatie van een nieuwe instantie ‘stemt’ elke boom op een label. De instantie krijgt vervolgens het label met de meeste stemmen.
- *k-Nearest Neighbours* classificeert nieuwe instanties door te zoeken naar de k dichtstbijzijnde instanties in de trainingset. Dit zijn de instanties waarvan de waarden voor de features het dichtst bij die van de te classificeren instantie liggen. Hieruit wordt het meest voorkomende label gekozen.
- *Naive Bayes* is een probabilistische classifier die gebruik maakt van voorwaardelijke kansen. De features van een te classificeren instantie worden hier als een vector $\mathbf{F} = (F_1, \dots, F_n)$ voorgesteld. Een label C_1 is waarschijnlijker dan een label C_2 wanneer $P(\mathbf{F}|C_1)P(C_1) > P(\mathbf{F}|C_2)P(C_2)$. Hierin kunnen de kansen $P(\mathbf{F}|C_i)$ en $P(C_i)$ voor alle labels C_i uit de trainingset berekend worden. Bij het classificeren wordt het meest waarschijnlijke label gekozen.
- *Support Vector Machines* beschouwt de features als een

²Meer informatie over deze klasse is te vinden op <http://weka.sourceforge.net/doc.dev/weka/attributeSelection/InfoGainAttributeEval.html>

multidimensionele ruimte. De instanties zijn dan punten in de featureruimte. Bij het leren worden de instanties lineair van elkaar gescheiden door een hypervlak, zodat de instanties aan de ene kant van het vlak een ander label hebben dan die aan de andere kant. Wanneer de instanties niet lineair te scheiden zijn, worden ze getransformeerd met een *kernel functie*. We gebruiken hier LibSVM, waarbij binaire (één-tegen-één) classificatie gedaan wordt: elk hypervlak scheidt de instanties van 2 verschillende labels. Een nieuwe instantie wordt ge-classificeerd door elke binaire classificatie als een stem voor een label te beschouwen. Het label met het grootste aantal stemmen wordt gekozen.

2.4 Experimenten en resultaten

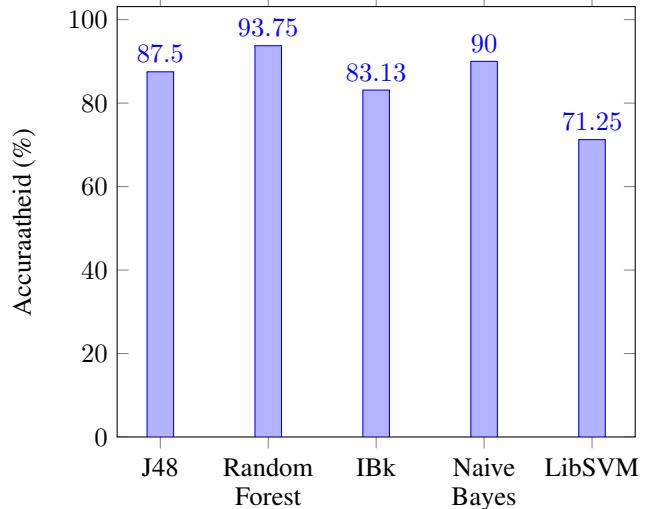
Om de methodes te evalueren, maken we gebruik van de Weka Machine Learning Toolkit [Weka]. Die bevat classifiers die de hierboven vermelde methodes implementeren. Voor beslissingsbomen maken we gebruik van de J48 classifier, een implementatie van het C4.5 algoritme. Verder gebruiken we IBk voor k-Nearest Neighbours en LibSVM voor Support Vector Machines.

De evaluatie gebeurt met 10-fold cross-validatie. Hierbij worden de instanties op tien verschillende manieren opgesplitst zodat telkens 90% van de instanties als trainingset wordt gebruikt om een model uit te leren. Op de overige 10% wordt het model geëvalueerd. De accuraatheid van elke methode wordt dan berekend als het gemiddelde percentage van correct geclassificeerde instanties in de verschillende test sets.

In figuur 3 wordt de accuraatheid van de verschillende classificatiemethodes vergeleken. We zien dat RandomForest de hoogste accuraatheid (93,75%) geeft voor de metingen. Ook J48 geeft een goede accuraatheid (87,5%). De hogere accuraatheid van Random Forests ten opzichte van beslissingsbomen is te verklaren door het feit dat Random Forests minder kans op overfitting heeft. [Breiman, 2001] Naive Bayes lijkt ook goed te werken, terwijl met IBk en vooral LibSVM een lagere accuraatheid bereikt wordt. De accuraatheid van de methodes kan echter wel verbeterd worden door de parameters te optimaliseren.

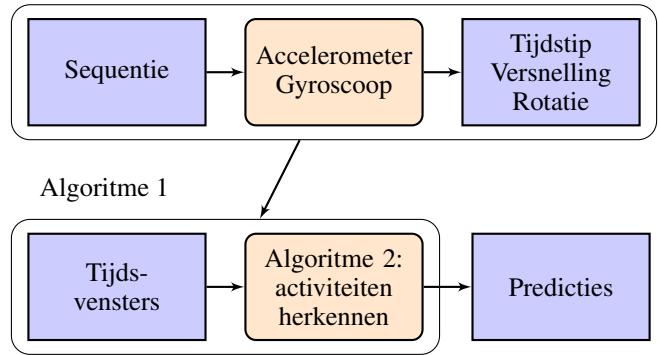
Voor het model dat door Random Forests geleerd werd, hebben we in figuur 5 een confusion-matrix geplot. Het valt op dat activiteiten met grote versnellingen – wandelen, lopen, fietsen en springen – goed te herkennen zijn. Alle metingen voor deze activiteiten worden immers correct geclassificeerd door het model. Niets doen, lift omhoog/omlaag en tanden poetsen hebben kleinere versnellingen dan de andere activiteiten. We zien dat twee metingen van niets doen als tanden poetsen worden geclassificeerd. Ook lift omhoog en omlaag worden makkelijk met elkaar verward. Omwille van de kleine versnellingen zijn de verschillen tussen de activiteiten subtieler, wat een verklaring kan zijn waarom deze activiteiten moeilijker te herkennen zijn.

We kunnen voor de afzonderlijke activiteiten besluiten dat door Random Forests het nauwkeurigste (93,75%) model geleerd wordt.



Figuur 3: Accuraatheid van classificatiemethodes

3 Sequenties van activiteiten



Figuur 4: Proces om predicties te maken voor een sequentie van activiteiten

Bij het herkennen van afzonderlijke activiteiten bevat elke meting exact één activiteit. De classificatiemethodes die we hiervoor gebruiken zijn dus niet rechtstreeks toepasbaar op metingen waarin verschillende activiteiten na elkaar gebeuren. Daarom gaan we op zoek naar een methode om voor een sequentie van activiteiten te bepalen welke activiteiten daarin gedaan worden en wanneer.

In figuur 4 wordt het proces beschreven om sequenties van activiteiten te herkennen. We beginnen met het opmeten van de versnelling en rotatie van verschillende sequenties. In het algoritme dat we zullen bespreken, worden de sequenties vervolgens in overlappende tijdsvensters geknipt. De activiteiten van de tijdsvensters worden bepaald met behulp van een model om afzonderlijke activiteiten te herkennen. Ten slotte worden hiermee predicties gemaakt voor de sequenties.

3.1 Gegevensverzameling

Met behulp van de MotionTracker [MT] applicatie werd opnieuw de versnelling en rotatie gemeten, maar nu van meer-

Wandelen	Lopen	Fietsen	Trap op	Trap af	Springen	Niets doen	Lift omhoog	Lift omlaag	Tanden poetsen	geklassificeerd ← als
16										Wandelen
	16									Lopen
		16								Fietsen
			15	1						Trap op
				16						Trap af
					16					Springen
						14			2	Niets doen
							10	6		Lift omhoog
							1	15		Lift omlaag
									16	Tanden poetsen

Figuur 5: Confusion matrix voor Random Forests

dere activiteiten na elkaar. In totaal zijn vier sequenties van elk ongeveer drie minuten lang opgemeten: twee verschillende combinaties van activiteiten, telkens door twee personen opgemeten. De ene combinatie bevat de activiteiten wandelen, trap op en af, niets doen, lopen en springen. De andere bevat wandelen, trap op en af, niets doen en lift omhoog en omlaag. Omdat lift versnellingen moeilijk van elkaar te onderscheiden zijn door het model voor afzonderlijke activiteiten (zie sectie 2.4), verwachten we dat de tweede combinatie moeilijker te herkennen is dan de eerste.

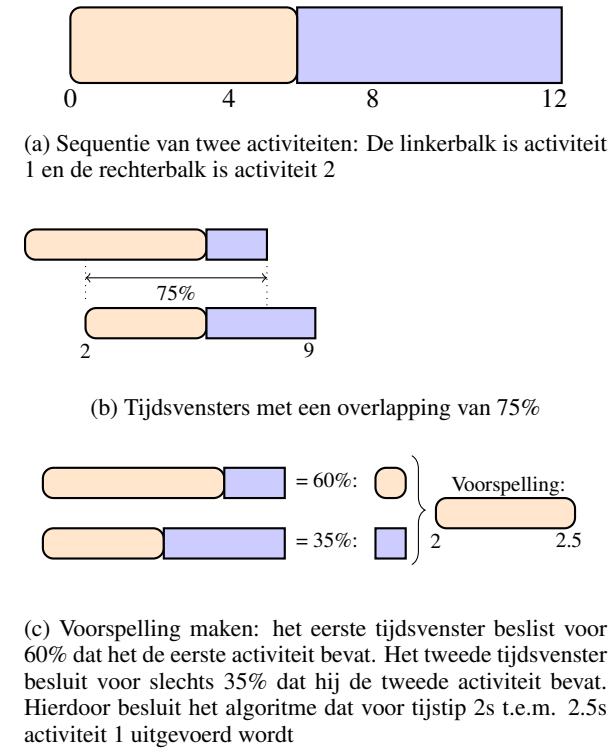
Om de metingen te evalueren, werd elke sequentie gelabeld. Elke label bevat de start- en eindtijden van de verschillende activiteiten in de bijhorende meting. De delen die niet gelabeld werden, beschouwen we als ruis. Ruis definiëren we als niet-herkenbare activiteit. Het gaat om periodes met activiteiten die niet tot de tien activiteiten behoren die we kunnen herkennen. De smartphone in de broekzak stoppen is hier een voorbeeld van.

3.2 Gegevensverwerking

Aangezien de modellen voor afzonderlijke activiteiten niet rechtstreeks kunnen toegepast worden op sequenties, moeten de sequenties geknipt worden in delen die wel één activiteit bevatten. We doen dit door de sequenties in tijdsvensters op te splitsen. De vensters hebben een lengte van enkele seconden en kunnen overlappen opdat er meer kans is dat een venster één activiteit bevat. We definiëren een overlapping als het percentage van de lengte van een tijdsvenster dat ook tot het eerst volgende tijdsvenster hoort. In figuur 6b wordt geïllustreerd wat een overlapping van bijvoorbeeld 75% betekent.

3.3 Algoritme

Bij het voorspellen van de activiteiten van een sequentie maken we gebruik van de tijdsvensters. We beschrijven een algoritme waarmee voor elke halve seconde van een sequentie een voorspelling kan gemaakt worden voor de activiteit die daarin gebeurde. Een algemeen overzicht van het algoritme is te vinden in figuur 6.



Figuur 6: Voorstelling van het algoritme om sequenties van activiteiten te classificeren. In figuur (a) wordt een sequentie van twee activiteiten getoond over 12 seconden. Vervolgens worden er tijdsvensters met overlapping van 75% geknipt in figuur (b). Tenslotte wordt een voorspelling gemaakt in figuur (c).

Het algoritme heeft drie parameters nodig: de lengte van de tijdsvensters in seconden, de overlapping ervan als een percentage en een ruis cutoff kans. De laatste parameter wordt verder verduidelijkt. In sectie 3.4 worden experimenten uitgevoerd om de waarden voor deze parameters te kiezen zodat de voorspellingen van het algoritme zo nauwkeurig mogelijk zijn.

Voor de pseudo-code van het algoritme verwijzen we naar Algoritme 1. De eerste stap bestaat uit het splitsen van de sequentie in tijdsvensters met de gegeven lengte en overlapping. Vervolgens wordt voor elk deel van een halve seconde lang van de sequentie gecontroleerd welke tijdsvensters dat deel bevatten. Deze vensters zijn samen met de ruis cutoff kans de parameters van Algoritme 2 om de activiteit van het deel van de sequentie te bepalen. Hierbij wordt voor elk venster een voorspelling gemaakt met behulp van een model om afzonderlijke activiteiten te classificeren. In sectie 2 bleek Random Forests het best te werken. Daarom gebruiken we hier het model dat door deze methode geleerd werd. Voor elk venster laten we ook de kans berekenen dat het voorspelde label correct is. Bij Random Forests wordt deze kans bepaald als het percentage van de bomen die dat label voorspellen. Als minstens twee vensters met het deel van de sequentie overlappen, houdt het algoritme geen rekening met de kansen wanneer de voorspelde labels van de vensters gelijk zijn. In dat geval wordt als voorspelling het label van de tijdsvensters voorspeld. Anders controleert het algoritme of er een label is met een kans groter dan de ruis cutoff kans. Indien dat zo is, wordt voor het deel van de sequentie het label met de grootste kans als activiteit voorspeld. Wanneer alle kansen onder de ruis cutoff kans liggen, zal de voorspelling ‘Ruis’ zijn.

Algoritme 1 Activiteiten van een sequentie (met start en eind timestamp in seconden) voorspellen door als uitvoer een lijst terug te geven met voor elke halve seconde de voorspelde activiteit

Parameters:

lengte van de tijdsvensters (in seconden)
overlapping van de tijdsvensters (percentage)
cutoff voor ruis (kans)

Splits de sequentie op in tijdsvensters met gegeven *lengte* en *overlapping*

```
for t = start; t ≤ einde; t += 0.5s do
    vensters = tijdsvensters die t en t + 0.5s bevatten
    voorspellingen[t] = ACTIVITEIT(vensters, cutoff)
        ▷ zie Algoritme 2
end for

return voorspellingen
```

3.4 Experimenten

We evalueren het algoritme door het toe te passen op de vier verschillende sequenties die we opgemeten hebben (zie sectie 3.1). Op die manier gaan we op zoek naar optimale waar-

Algoritme 2 Activiteit van een deel van een sequentie bepalen, met een gegeven lijst van tijdsvensters en een ruis cutoff kans

```
procedure ACTIVITEIT(vensters, cutoff)
    labels = voorspelde activiteiten voor vensters
    kansen = kansen voor de labels
        ▷ berekend door model van Random Forests

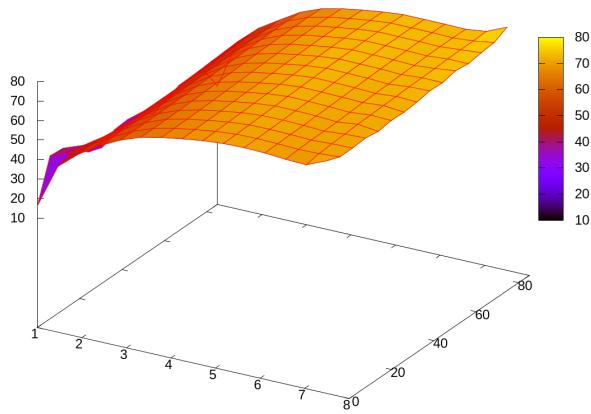
    if aantal vensters ≥ 2 and alle labels gelijk then
        return labels[0]
    else
        if kansen[i] > cutoff voor een i then
            return labels[i] met kansen[i] maximaal
        else
            return ‘Ruis’
        end if
    end if
end procedure
```

den voor de parameters. De parameters moeten zodanig gekozen worden dat de voorspellingen van het algoritme zo nauwkeurig mogelijk zijn. We verwachten dat de tijdsvensters vier seconden lang moeten zijn om lift omhoog en omlaag te kunnen herkennen, aangezien deze activiteiten ongeveer vier seconden duren. Wanneer we daarbij een overlapingspercentage van 75% kiezen, zullen de tijdsvensters steeds 1s verschuiven. Op die manier zullen de tijdsvensters waarschijnlijk vaker één activiteit bevatten. Verder verwachten we dat voor de ruis cutoff een kans van 50% een redelijke waarde zal zijn. Zo worden voorspellingen steeds met minstens 50 % zekerheid gemaakt.

Om de accuraatheid van de voorspellingen van het algoritme te kunnen evalueren, definiëren we eerst hoe we die berekenen. Zoals eerder vermeld, werden de opgemeten sequenties gelabeld zodat bekend is wanneer de verschillende activiteiten in elke meting starten en eindigen. Voor delen die niet gelabeld werden, veronderstellen we dat die ‘Ruis’ als label hebben. We zullen de voorspellingen van het algoritme vergelijken met deze labels. Dit doen we door per halve seconde van de sequentie te controleren of de voorspelling overeenkomt met het label. Deze controle wordt echter niet gedaan voor overgangen in de sequentie: dat zijn delen van de sequentie die op maximaal 1 seconde van een start- of eindtijd van een gelabelde activiteit liggen. De accuraatheid wordt dan berekend door van alle gecontroleerde voorspellingen het percentage correcte voorspellingen te bepalen.

3.5 Resultaten

In figuur 7 worden de verschillende combinaties van de tijdsvenster- en overlapingsgrootte getoond met de behaalde accuraatheid. Indien we de x-as volgen (door de overlapping constant te houden), zien we dat naarmate de tijdsvenster-grootte stijgt, ook de accuraatheid stijgt. Er wordt een maximum gehaald rond de vier tot zes seconden, daarna daalt de accuraatheid terug. Bij het volgen van de y-as (met een constante tijdsvenster-grootte), constateren we dat hoe groter het overlapingspercentage wordt, hoe nauwkeuriger de voorspellingen gemaakt kunnen worden.



Figuur 7: 3D-plot dat de resultaten toont van het experiment (besproken in sectie 3.4). Op de x-as wordt de venstergrootte geplot in seconden. Op y-as zijn de overlappendingen te zien in procenten. Uitvoering van het algoritme met de combinatie van deze parameters resulteert in een accuraatheid (geplot op de z-as)

Deze resultaten komen overeen met de hypothese die eerder gemaakt werd. De meeste activiteiten duren enkele seconden lang. Liftversnellingen bijvoorbeeld duren ongeveer vier seconden. Hierdoor is te verklaren waarom de tijdsvensters langer dan vier seconden moeten zijn. Tijdvensters van één of twee seconden zijn niet altijd lang genoeg om een hele periode van een activiteit te bevatten. Grottere tijdsvensters leveren een lagere accuraatheid op omdat de kans dat éénzelfde venster meerdere activiteiten bevat groter is. Daardoor is het moeilijker om de juiste activiteit te herkennen. De stijgende nauwkeurigheid voor toenemende overlappingsgrootte was ook te verwachten. Hoe groter de overlapping, hoe meer voorspellingen gemaakt kunnen worden per halve seconde van de sequentie. Hierbij moet wel een trade-off gemaakt worden: meer voorspellingen maken, vraagt immers meer tijd.

Uit het vorige experiment kunnen we concluderen dat de optimale venstergrootte vier tot zes seconden is in combinatie met een overlappingspercentage van 75%. Deze waarden samen met 50% ruis cut-off leveren een gemiddelde nauwkeurigheid van 84% voor de opgemeten sequenties.

Voor de sequentie van activiteiten kunnen we concluderen dat voor ons algoritme de optimale venstergrootte vier tot zes seconden is in combinatie met 75% overlapping. Er wordt een gemiddelde accuraatheid behaald van 84%

4 Conclusie

Het doel van dit onderzoek was om een model te vinden om afzonderlijke bewegingen te herkennen. Vervolgens gebruiken we dit model om een sequentie van verschillende activi-

teiten te evalueren.

In sectie 2 werden twee experimenten uitgevoerd in verband met afzonderlijke activiteiten. Het eerste experiment betreft de feature selectie. Hierbij werd onderzocht hoeveel features nodig zijn om een voldoende nauwkeurig model te vinden. Het resultaat was dat enkel statistische features een model opleveren met 90% nauwkeurigheid. We kunnen bij hetzelfde experiment ook concluderen dat van de 80 Fourier transformatie features slechts een aantal nodig zijn. In het tweede experiment werden verschillende classificatiemethodes met elkaar vergeleken. We maakten gebruik van vijf veel voorkomende machine learning technieken: beslissingsbomen, Random Forests, k-Nearest Neighbours, Naive Bayes en Support Vector Machines. Van deze methodes kwam Random Forests als beste naar boven met een nauwkeurigheid van 93,75%.

In sectie 3 werden sequenties van activiteiten geëvalueerd. Hiervoor hebben we een algoritme ontwikkeld waarvan de pseudocode besproken werd in sectie 3.3.

Het algoritme vereist drie parameters: een tijdsvenstergrootte in seconden, een overlappingspercentage en een ruis cutoff kans. Uit het experiment in sectie 3.4 bleek dat de optimale waarde voor de tijdsvenstergrootte vier tot zes seconden is en 75% overlappenden een goede accuraatheid geven. De waarden werden getest op sequenties van ongeveer 3 minuten lang met steeds 4 à 5 verschillende activiteiten. Met behulp van het model van Random Forests (dit is het model dat gevonden werd in sectie 2.4) werd er een gemiddelde nauwkeurigheid van 84% gevonden.

5 Verder werk

Ondanks het feit dat tijdens dit onderzoek een redelijk model werd gevonden, kan dit altijd verder verbeterd worden.

5.1 Afzonderlijke activiteiten

Het model van de afzonderlijke activiteiten kan verbeterd worden door de gegevensverzameling te vergroten. Dit betekent dat meer gegevens verzameld moeten worden door meerder proefpersonen. Hierdoor kan een algemener model berekend worden voor de verschillende afzonderlijke activiteiten, waardoor het model nauwkeurigere voorspellingen kan maken bij de sequenties van activiteiten.

Bovendien kunnen er modellen gegenereerd worden voor nieuwe activiteiten. Het toevoegen van extra activiteiten zal zoals eerder vermeld ook meer gegevens vragen.

5.2 Sequenties van activiteiten

Het algoritme voor sequenties van activiteiten te evalueren kan ook nog verder geoptimaliseerd worden. Zo zijn bepaalde overgangen van activiteiten totaal onmogelijk. Het herkennen van fietsen na een liftversnelling is bijvoorbeeld onwaarschijnlijk. Zulke overgangen kunnen gefilterd worden door het gebruik van Markov ketens. Verder hebben we tijdens het onderzoek gemerkt dat soms enkele seconden als tanden poetsen herkend worden tijdens niets doen. Ook dit is onwaarschijnlijk en zou ook gefilterd kunnen worden met behulp van Markov ketens.

Tijdens het onderzoek werd geen rekening gehouden met de performantie (noch snelheid, noch geheugengebruik). Deze kan verder onderzocht en eventueel verbeterd worden.

Tenslotte is het nog mogelijk om het algoritme te implementeren in een smartphone-applicatie.

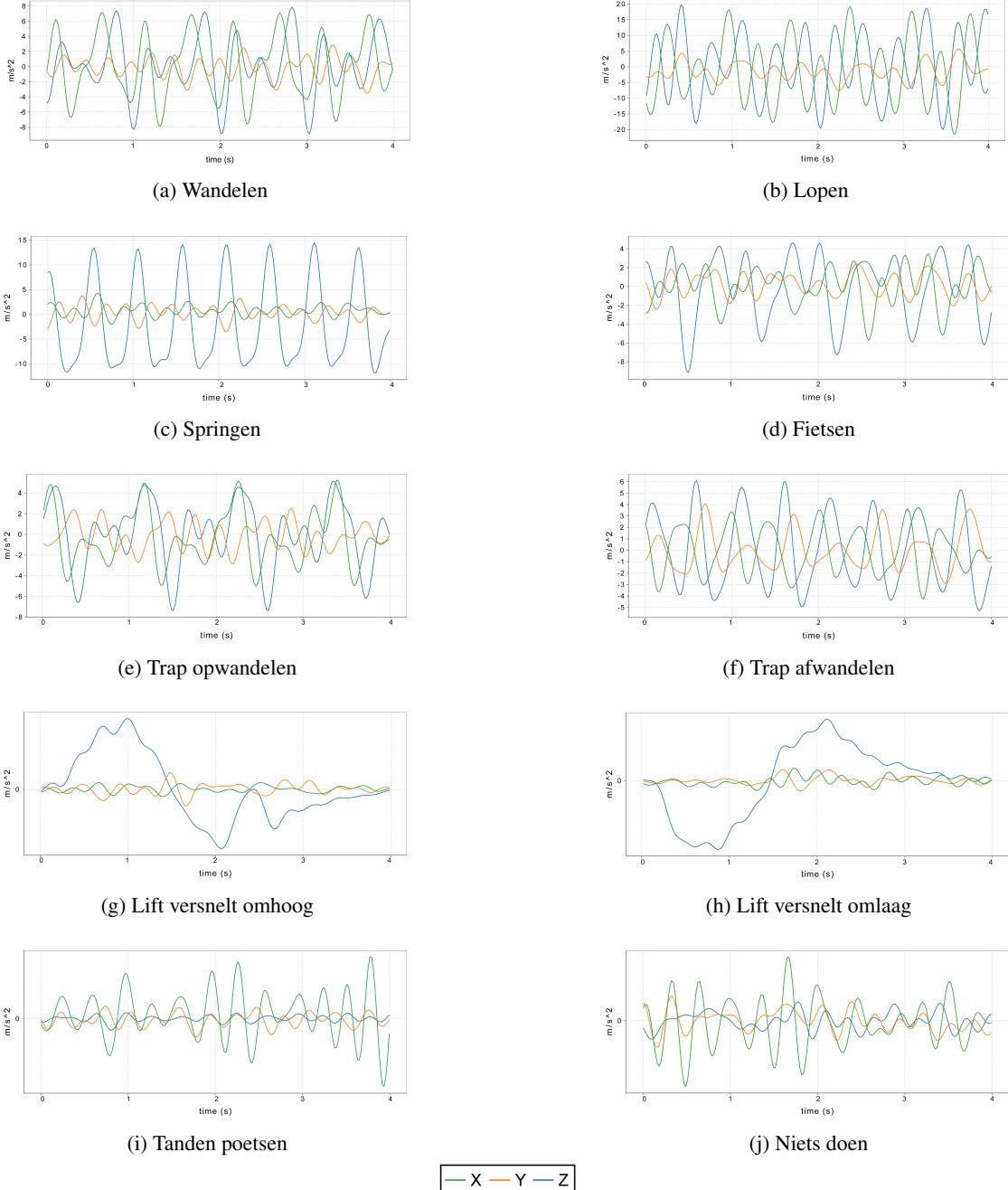
6 Dankwoord

Hierbij willen we graag Wannes Meert en Leander Schietgat bedanken. Eerst en vooral voor hun raad en ondersteuning. Ten tweede voor het ter beschikking stellen van de tools: MotionTracker en MotionFingerPrint.

Referenties

- [Breiman, 2001] Leo Breiman. Random Forests. *Machine Learning*, 45(1):5–32, Oktober 2001.
- [Ravi2005] Nishkam Ravi, Nikhil Dandekarm, Preetham Mysore and Michael L. Litmtman *Activity recognition from accelerometer data*, aaai.org, 2005.
- [Chen2012] Mingyu Chen, Ghassan AlRegib and Biing-Hwang Juang *Feature Processing and Modeling for 6D Motion Gesture Recognition*, Dept. of Applied Mathematics and Analysis, University of Barcelona, 2012.
- [Casale] Mingyu Chen, Ghassan AlRegib and Biing-Hwang Juang *Human Activity Recognition from Accelerometer Data Using a Wearable Device*, IEEE, 2012.
- [MFP] Wannes Meert and Leander Schietgat *MotionFingerPrint tool*.
- [MT] Wannes Meert and Leander Schietgat *MotionTracker tool*.
- [Weka] University of Waikato *Weka machine learning tool kit*, <http://www.cs.waikato.ac.nz/ml/weka/>.

7 Bijlage



Figuur 8: De tien verschillende activiteiten die afzonderlijk opgemeten werden voor dit onderzoek. Hierbij moet opgemerkt worden dat versnelling (y-as) niet op elke figuur dezelfde stap heeft. Zo hebben de activiteiten die getoond worden op figuur (g) t.e.m. (j) een versnelling kleiner dan één.