Bewegingsherkenning met een smartphone

Arne De Brabandere
arne.debrabandere@student.kuleuven.be

Menno Keustermans
menno.keustermans@student.kuleuven.be

Abstract

Bewegingsherkenning is een belangrijke doelstelling van 'context aware computing'. Uit de metingen van de accelerometer en gyroscoop van een smartphone kan de activiteit van de gebruiker bepaald worden. In het eerste deel van ons onderzoek hebben we gezocht naar een model om afzonderlijke activiteiten te herkennen. Het nauwkeurigste model werd gevonden met de classificatiemethode Random Forest met een accuraatheid van 93%. Vervolgens hebben we dit model gebruikt om een sequentie van verschillende bewegingen te evalueren. Het algoritme dat we hiervoor ontwikkeld hebben, knipt de sequentie in opeenvolgende tijdsvensters van x aantal seconden die ook met een bepaald percentage kunnen overlappen. Uit experimenten bleek dat hiervoor een tijdsvenstergrootte van vier tot zes seconden met een overlapping van 75% het nauwkeurigst is.

1 Introductie

TODO situering van het werk + bijdragen: - waarom bewegingen herkennen? (extra bronnen gebruiken - waarom met smartphones? - korte uitleg over onderzoek dat al gedaan is hierover voor afzonderlijke activiteiten (zie papers van literatuurstudie) - uitleg over eigen bijdrage: (1) van model om afzonderlijke activiteiten te herkennen naar algoritme om sequenties van activiteiten te herkennen (+ duidelijk zeggen wat we met afzondelijke activiteiten en sequenties bedoelen)

Gerelateerd werk

2 Afzondelijke 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. 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.

Merk op dat we in plaats van de activeiten '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. De eerste stap bestaat uit het verzamelen van de gegevens. Hiervoor meten we de versnellingen 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 juiste geclassificeerde samples ten op zichte van het totaal aantal samples met behulp van cross-validatie.

2.1 Gegevensverzameling

Alle gegevens werden opgemeten door de MotionTracker tool. Dit is een Android-applicatie die de versnelling en rotatie (respectievelijk gemeten door de accelerometer en gyroscoop van de smartphone) samplet aan 50Hz. Als uitvoer geeft de applicatie een .log-bestand 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 gestopt 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 .log-bestand weggeknipt, 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 .log-bestand 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.

MotionFingerprint berekent in totaal 134 features[1] verdeeld onder vier soorten:

- *Statistische features:* dit zijn gemiddelde, standaardafwijking van zowel z- als xy-versnellingen en vermogen. En correlatie tussen z- en xy-versnelling,
- *Fourier transformatie:* deze worden berekend in het frequentie domein van de metingen, zoals amplitudes volgens de verschillende assen,
- *Wavelet transformatie:* **TODO**,
- *Hidden Markov models:* log-likelihood voor het model van elke activiteit.

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 te leren.

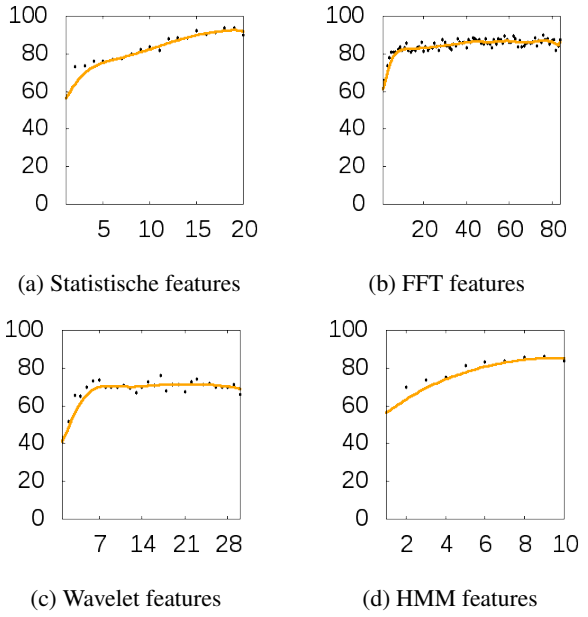
Feature selectie

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* klasse. 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 het proces dat 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 Forest classificatiemethode in Weka. Het gevonden model wordt uiteindelijk geëvalueerd op de test set van de eerste cross-validatie.

In figuur 1 worden de resulaten 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% activiteit voorspellen. We stellen vast dat van het grote aantal Fourier tranformatie features slechts een tiental nodig zijn om een nauwkeurigheid van 80% te bekomen. 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.



(a) Statistische features (b) FFT features

(c) Wavelet features (d) HMM features

Figuur 1: 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 Forest geleerd werd.

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 Forest, 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:

- *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

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

voor op de feature van de knoop waaruit de tak vertrekt. Om een nieuwe instantie te classificeren worden de juiste takken gevolgd tot in een bepaalde bladknoop. De instantie wordt dan geclassificeerd als het label in die bladknoop.

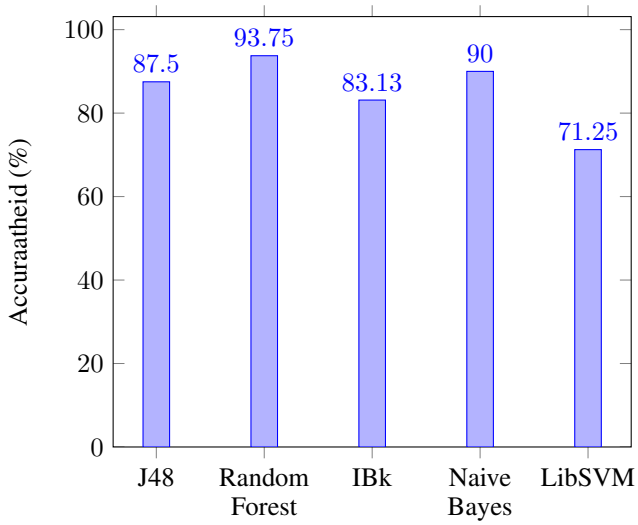
- *Random Forests* zijn combinaties van beslissingsbomen met een willekeurig deel van de trainingset en een willekeurig deel van de features. Bij de classificatie van een nieuwe instantie 'stemt' elke boom op een label. De instantie krijgt vervolgens het label met de meeste stemmen.
- *k-Nearest Neighbours* 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 ligt. Hieruit wordt het meest voorkomende label gekozen.
- *Naive Bayes* is een probabilistische classifier die gebruik maakt van voorwaardelijke kansen. De kans op een label C met gegeven waarden voor de features $\mathbf{F} = (F_1, ..., F_n)$ wordt berekend als $P(C|\mathbf{F}) = \frac{P(\mathbf{F}|C)P(C)}{P(\mathbf{F})}$ waarin $P(\mathbf{F}|C)$, $P(C)$ en $P(\mathbf{F})$ uit de trainingset kunnen berekend worden. Bij het classificeren wordt het label met de grootste kans gekozen.
- *Support Vector Machines* beschouwt de features als een multidimensionele ruimte. De instanties zijn dan punten in de featureruimte. Bij het leren worden de instanties lineair van elkaar gescheiden door hypervlakken, 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 geclassificeerd 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. 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 testsets.

In figuur 2 wordt de accuraatheid van de verschillende classificatiemethodes vergeleken. We zien dat Random Forest de hoogste accuraatheid (93,75%) geeft voor de metingen. Ook



Figuur 2: Accuraatheid van classificatiemethodes

J48 geeft een goede accuraatheid (87.5%). De hogere accuraatheid van Random Forest ten opzichte van beslissingsbomen is te verklaren door het feit dat Random Forest 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 de optimaliseren.

Voor het model dat door Random Forest geleerd werd, hebben we in figuur 3 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 versnelt 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 gemakkelijk 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.

3 Sequenties 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.

3.1 Gegevensverzameling

Met behulp van de MotionTracker applicatie werd opnieuw de versnelling en rotatie gemeten, maar nu van meerdere activiteiten na elkaar. In totaal zijn vier sequenties van elk ongeveer drie minuten lang opgemeten: twee verschillende combinaties van activiteiten, telkens door twee personen opgeme-

Wandelen	Lopen	Fietsen	Trap op	Trap af	Springen	Niets doen	Lift omhoog	Lift omlaag	Tanden poetsen	geclassificeerd ← 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

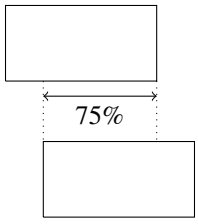
Figuur 3: Confusion matrix voor Random Forest

ten. 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 met behulp van een .csv-bestand. Dit bestand 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 4 wordt geïllustreerd wat een overlapping van bijvoorbeeld 75% betekent.



Figuur 4: Tijdsvensters met een overlapping van 75%

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.

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

3.4 Experimenten en resultaten

TODO: - hoe het experiment wordt gedaan - hoe gebeurt de evaluatie? - accuraatheid plotten in functie van grootte van

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 = $start$; $t \leq einde$; t += $0.5s$ **do**
    $vensters$ = tijdsvensters die t en $t + 0.5s$ bevatten
    $voorspellingen[t]$ = ACTIVITEIT($vensters, cutoff$)
        ▷ zie algoritme 2
end for

return $voorspellingen$

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 Forest

    if aantal $vensters \geq 2$ **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

tijdsvensters en overlap + tijdsvensters vergelijken + iets over ruis cut-off + hypothese (waarschijnlijk 4sec en 3/4 overlap, omwille van activiteit lift versnelt omhoog/omlaag)

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

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. Grotere tijdsvensters leveren een lagere accuuraatheid 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 [TODO]% voor de opgemeten sequenties.

4 Conclusie

Het doel van dit onderzoek was om een model te vinden om afzonderlijke bewegingen te herkennen. Vervolgens gebruikten we dit model om een sequentie van verschillende activiteiten 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 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 tiental nodig zijn. In het tweede experiment werden verschillende classificatiemethodes met elkaar vergeleken. We maakten gebruik van vijf veel voorkomende machine learning technieken: beslissingsbomen, Random Forest, k-Nearest Neighbours, Naive Bayes en Support Vector Machines. Van deze methodes kwam Random Forest 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 [TODO]. Het algoritme ver-

eist drie parameters: een tijdsvenstergrootte in seconden, een overlappingspercentage en een een ruis cutoff kans. Uit het experiment in sectie [TODO] bleek dat de optimale waarde voor de tijdsvenstergrootte vier tot zes seconden is en 75% overlappingen een goede accuraatheid geven. De waarden werden getest op sequenties van ongeveer 3 minuten lang met steeds 4 á 5 verschillende activiteiten. Met behulp van het model van Random Forest (dit is het model dat gevonden werd in sectie [TODO]) werd er een gemiddelde nauwkeurigheid van 80% 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 meerdere 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 Hidden Markov Models. 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 gefilterd kunnen worden met behulp van HMMs.

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.