Stappenteller
Eindrapport, Groep B

Fordeyn Tibo, Dobbelaer Xander, De Wispelaere Tobias, Decavele Julie

Begeleiders: Prof. Beunis Filip, de Mildt Maarten, Bracke Vincent

23 december 2022

Inhoudsopgave

Samenvatting

In iedere GSM zit een accelerometer. De output van dit stuk hardware is de valversnelling van het toestel - de GSM - tegenover de aarde. Dit is data die kan worden geïnterpreteerd met een algoritme om te achterhalen hoeveel stappen de gebruiker met het toestel zet. Het uiteindelijk geïncorporeerd algoritme voor dit project had een foutmarge van hoogstens zes procent, dit bij één van de testen waarbij de gebruiker het toestel in de hand hield. Testen met het toestel in de achterzak of jaszak hadden significant lagere foutmarges. Vaak werden perfecte resultaten verkregen. Toen dit algoritme vergeleken werd met dat van de meest populaire app uit de app store, bleek het meer accuraat in bijna elke test. Eenmaal het momentane aantal stappen achterhaald is, wordt het weergegeven in een mobiele app. Deze app is geschreven in Java, met Android Studio als IDE - Integrated Development Environment. Naast het weergeven van stappen biedt de app, met een overzichtelijke UI, extra info bij dat aantal stappen. Zo wordt bijvoorbeeld het aantal verbrande calorieën weergegeven, als ook hoe dicht de gebruiker is bij zijn of haar dagelijks doel.

1 Accelerometer en dataverwerking

Het project begint bij de accelerometer. Om stappen te kunnen detecteren wordt de ouput van de accelerometer in het toestel gebruikt.

1 Sample rates

Verschillende sample rates geven een verschillend aantal metingen per minuut als output. Er zijn vier sample rates; Normal, Game, UI en fastest. In de eerste labsessie werd nagegaan hoeveel metingen per minuut bij iedere sample rate verkregen wordt met ons toestel - Tabel 1.1.

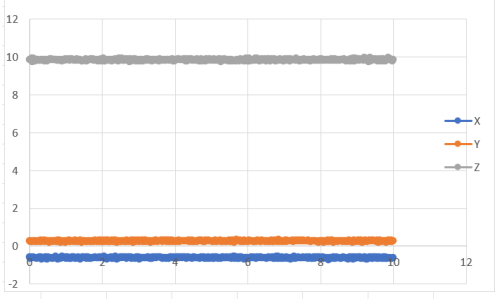
Sample rates	Metingen per minuut
Normal	300
UI	1000
Game	3000
Fastest	6000

Tabel 1.1: Sample rates.

Het is prominent dat de gepaste sample rate gekozen wordt. Een enorme hoeveelheid data is in theorie wenselijk, maar de hardware in het toestel mag niet overbelast worden. Om hier een evenwicht te vinden wordt UI geütiliseerd.

2 Output met het toestel in rust

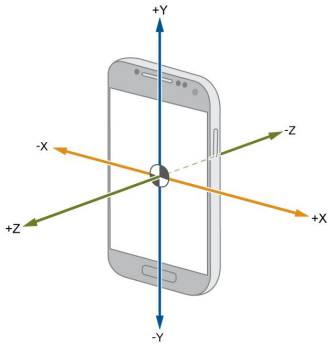
Wanneer het toestel plat op tafel wordt gelegd, wordt langs de x-richting noch langs de y-richting valversnelling ondervonden. Enkel langs de z-richting nadert de ondervonden valversnelling een getal 9.81, de valversnelling op aarde. Wat meteen opvalt bij het kijken naar deze output is de ruis; er is duidelijk een niet verwaarloosbare hoeveelheid ruis aanwezig. Dit is een logisch fenomeen; elk object op aarde trilt continu. Op ruwe data zal bijgevolg steeds een bepaalde hoeveelheid ruis zitten.



Figuur 1.1: Output waarden van accelerometer in rust

3 Oriëntatie van de assen

Wanneer het toestel in andere posities wordt geplaatst, zal de uitvoerwaarde verschillen, en zal aan de hand van de nieuwe output de oriëntatie van de assen kunnen worden bepaald. Figuur 1.2 geeft de bevindingen weer.



Figuur 1.2: Oriëntataie van de assen

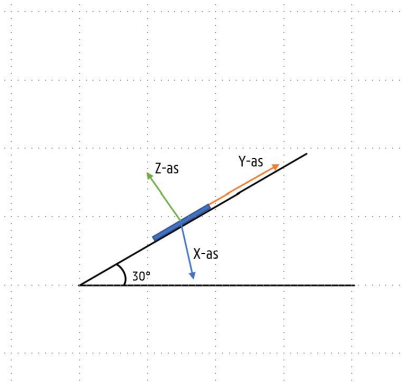
4 Output onder invloed van een hoek

De outputwaarden voor de verschillende assen zijn zeer afhankelijk van de hoek waarin het toestel zich bevindt. Met 1.1 en 1.2 wordt de versnelling langs de assen (in $\frac{m}{s^2}$) gevonden.

$$s_z = s_{tot} \cdot \cos(\alpha) \tag{1.1}$$

$$s_y = s_{tot} \cdot \sin(\alpha) \tag{1.2}$$

Met α als de hoek tussen het toestel en het aardoppervlak. s is de gebruikte variabele omdat over versnelling wordt gesproken. Er staan nu formules voor z en y, maar het toestel kan gedraait worden om de x-waarden te vinden. In Figuur



Figuur 1.3: Verduidelijking vergelijkingen

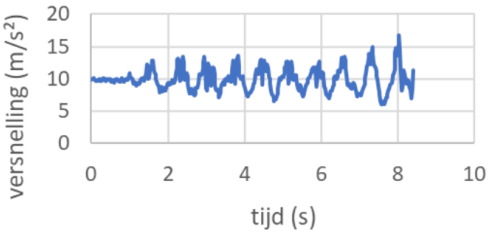
1.3 worden de vergelijkingen visueel ondersteund. Hoe s_{tot} berekend wordt, is verduidelijkt in 1.3.

5 Van 3D naar 1D

Met al deze gegevens werken is mogelijks lastig en onduidelijk. De drie versnellingen moeten tot één enkele versnelling worden omgevormd, door gebruik te maken van de afstandsformule zoals weergegeven in 1.3.

$$s_{tot} = \sqrt{s_x^2 + s_y^2 + s_z^2} \tag{1.3}$$

Hierbij moeten geen zorgen gebaard worden wat betreft dataverlies; het enige relevante om stappen te bepalen is de totale versnelling s_{tot}. Figuur 1.4 toont een output na het zetten van 10 stappen omgezet naar totale versnelling.



Figuur 1.4: Onbewerkte output totale versnelling

Merk op dat er ongewenst veel ruis, kleine pieken, in deze grafiek zitten. Dit zou problemen kunnen veroorzaken, aangezien stappen tellen eigenlijk neerkomt op het tellen van deze pieken.

6 Ruis aanpakken

Om de ruis kwestie zo goed mogelijk aan te pakken, wordt gebruik gemaakt van bepaalde technieken om gegevens te verwerken. Twee technieken werden bekeken; lopend gemiddelde en gewogen gemiddelde.

6.1 *Lopend gemiddelde*

Voor een lopend gemiddelde worden minstens twee opeenvolgende elementen genomen en wordt daar het gemiddelde van genomen tot vorming van een nieuw element. Dat nieuw element komt op de grafiek van het lopend gemiddelde. Hierbij kunnen deze opeenvolgende termen waarden voor of na het huidige element zijn. Het kan ook dat enkele elementen ervoor en enkele elementen na het huidige element liggen. Het lopend gemiddelde berekend voor twee elementen wordt beschreven door 1.4.

$$s_{i,lg} = \frac{s_{i+j} + s_{i+j+1}}{2} \tag{1.4}$$

Daarbij is $s_{i,lg}$ het nieuwe lopend gemiddelde element met index i. s_{i+j} en s_{i+j+1} zijn datapunten uit de grafiek van de totale versnelling. De index wordt op die manier geschreven, om erop te wijzen dat de elementen gekozen worden afhankelijk van i. Kies $j < 0$ om elementen voor i te gebruiken. Kies $j = 0$ om gebruik te maken van datapunt i en de opeenvolgende.

Dit moet niet perse met twee elementen gebeuren, er blijft zo namelijk nog te veel ruis over. Algemeen wordt het lopend gemiddelde als bij 1.5 neergeschreven.

$$s_{i,lg} = \frac{s_{i+j} + s_{i+j+1} + \ldots + s_{i+j+n}}{N} \tag{1.5}$$

Waarbij N naar het aantal elementen verwijst. Hier zou dit $n + 1$ zijn. Dit kan meer compact geschreven worden.

$$\equiv s_{i,lg} = \frac{1}{N} \sum_{x=0}^{N-1} s_{i+j+x} \tag{1.6}$$

Waarbij de keuze van j dus te bepalen valt.

6.2 *Gewogen gemiddelde*

Bij een gewogen gemiddelde gebeurt exact hetzelfde als bij het lopend gemiddelde, maar een factor q wordt met ieder element vermenigvuldigd. De som van alle factoren q is gelijkk aan één. Op deze manier worden bepaalde waarden

zwaarder doorgewogen dan anderen, vandaar de naam ”gewogen gemiddelde”.

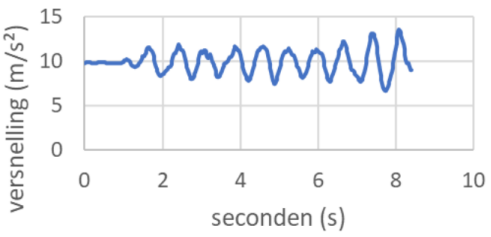
$$s_{i,gg} = \frac{1}{N} \sum_{j=0}^{N-1} q_j \cdot s_{i+j+x} \quad | \quad \sum_{j=0}^{N-1} q_j = 1 \qquad (1.7)$$

6.3 *Uiteindelijke keuze voor gegevensverwerking*

Naarmate een hoger aantal elementen gekozen wordt om mee te werken, kost dit het toestel een stuk meer energie. Er moet een evenwicht gevonden worden. Er is nood aan duidelijke data zonder ruis, maar de app moet efficiënt blijven. Het viel op dat de verkregen grafiek bij een gecentreerd lopend gemiddelde met elf elementen goed werkt. Dit zorgt niet voor een te grote hoeveelheid calculaties die moeten worden verricht. Desondanks levert het een grafiek op met minder ruis.

$$s_{i,lg} = \frac{1}{11} \sum_{j=0}^{10} s_{i+j-5} \qquad (1.8)$$

Er werd dus gewerkt met 1.8. Figuur 1.5 toont de nieuw verkregen output.



Figuur 1.5: Gecentreerd lopend gemiddelde van 11 elementen

Het gewogen gemiddelde had een zeer gelijkaardige output. Het lijkt eerder onnodig zaken extra moeilijk te maken door met een extra factor q te vermenigvuldigen. We maakten geen gebruik van deze techniek.

2 Detectiemechanisme

Het doel van de stappenteller is uiteraard om het correct aantal stappen weer te geven. Dit is het kerndoel van de app. Om hierin te slagen zijn er meerdere detectoren gemaakt en getest door legio metingen. Zo is de beste detector voor de diverse bekeken bewegingen - contexten - gevonden. De bekeken contexten zijn: toestel in de hand, broekzak, jaszak, achterzak.

1 Implementatie lopend gemiddelde

Er wordt natuurlijk begonnen met een lege lijst totale versnelling waarden. De eerste zes datapunten die gemeten worden, worden omgerekend naar hun totale versnelling en in een lijst gezet. Pas na deze zes waarden er zijn zal er dus een lopend gemiddelde kunnen berekend worden - de eerste waarden van de grafiek zijn dus een gemiddelde van zes, niet van elf waarden, aangezien de vijf eerdere waarden nog niet bestaan. Zoals besproken in 1.1 wordt de UI sample rate gebruikt met een output van duizend waarden per minuut. Het duurt dus slechts iets meer dan een derde van een seconde om deze zes waarden te verkrijgen, verwaarloosbaar weinig. Praktisch onmiddellijk kunnen er stappen geteld worden door op die manier te werken, zorgen over een trage respons zouden overbodig zijn. Vanaf elf waarden verkregen zijn, wordt er telkens een element verwijderd in het begin, en toegevoegd aan het einde.
Visueel wordt na zes datapunten een lijst zoals hieronder verkregen.

$$s_1 \quad | \; s_{...} \quad | \; s_6.$$

Hiermee wordt de output voor $s_{1,lg}$ berekend, waarbij met de s_i datapunten van totale versnelling bedoeld worden. Na elf datapunten wordt dan de onderstaande lijst verkregen.

$$s_1 \quad | \; s_{...} \quad | \; s_{11}.$$

Daarmee wordt het gemiddelde voor $s_{6,lg}$ genomen. Vanaf datapunt s_{12} er is, wordt een lijst als hieronder verkregen.

$$s_2 \quad | \; s_{...} \quad | \; s_{12}.$$

Dit voor het element $s_{7,lg}$. Dit proces blijft zich herhalen terwijl stappen geteld worden door te kijken naar de nieuwe output.

2 Dit algoritme in pseudocode

Algorithm 1: Hoe de lijst voor lopend gemiddelde wordt gemaakt in pseudocode

Input: Output van de accelerometer
Output: Lijst om het lopend gemiddelde mee te maken

```
1  s_tot = list(totale versnelling 1, totale
     versnelling 2, ...)
2  elementsInList = len(s_tot)
3  listForMeanAverage = list()
4  if elementsInList < 11 then
5  |  listForMeanAverage.append(nieuwste
   |    element in s_tot)
6  else
7  |  listForMeanAverage.remove(oudste
   |    element in de lijst)
   |    listForMeanAverage.append(nieuwste
   |    element in s_tot)
8  end
9  return listForMeanAverage
```

De elementen die s_{tot} opmaken worden be-

rekend als in 1.3; het eerste datapunt - een x, y en z versnelling - wordt hiermee omgezet en is het eerste element in die lijst.

3 Dummydetector

Dit was onze eerste, eerder primitieve, stappendetector. Het controleerde of het lopend gemiddelde boven een bepaalde grenswaarde kwam en indien dit het geval was werd er een stap geteld. Vervolgens wacht de detector tot het lopend gemiddelde er opnieuw onder zakt, vanaf dan herhaalt het proces zich en kan het een nieuwe stap tellen indien de grenswaarde wordt bereikt. Op die manier worden in essentie de pieken van de grafiek geteld. Het tellen van die pieken is eigenlijk het overkoepelende idee van een stappendetector. Om dit te doen wordt onze grenswaarde op 5 - proefondervindelijk bepaald - gelegd. In algoritme 2 wordt deze waarde CF genoemd, voor "constant function".

Algorithm 2: Dummydetector algoritme

Input: Lopend gemiddelde
Output: Aantal stappen

```
1  lg = (een gemiddelde van totale
     versnellingen)
2  Control = 0
3  Steps = 0
4  CF = 5
   /* Onderstaande code wordt herhaald
      telkens er een nieuw element lg is.  */
5  if lg > CF and Controle = 0 then
6  |  Control = 1
7  |  Steps ++
8  if i < CF then
9  |  Control = 0
10 return Aantal stappen;
```

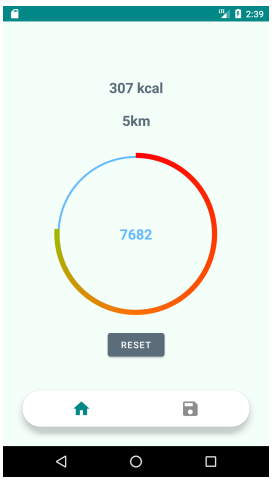
Dit principe is heel simplistisch, en gaf desondanks voor de meeste bewegingen al een relatief goed resultaat. Echter gaf het voor bepaalde bewegingen een totaal verkeerd beeld. Dit wordt later met diepgang besproken (Hoofdstuk 4).

4 Complexe detector

Uit de eerste labsessies bleek dat het verschil in versnelling tussen stappen met het toestel in de hand tegenover lopend met het toestel in de broekzak zeer groot is. Te groot om met een gemiddelde grenswaarde voor beide een goed resultaat te krijgen. De stappenteller moest dus op een of andere manier weten welke beweging de gebruiker aan het maken is. Dit probleem wordt aangepakt door de maximumwaarde variabel te maken. We schrijven een dynamische detector die de maximumwaarde opnieuw zal berekenen voor elke 400 waarden. Met deze nieuwe methode zou de grens zeer snel correct aangepast worden. Dit is het hoofdprincipe dat de nieuwe dynamische detector onderscheidt van de dummydetector, maar daarnaast is er nog één extra verandering tegenover de dummydetector. Stel het toestel wordt op tafel geplaatst voor een minuut en vervolgens opgepakt voor een wandeling; de drempelwaarde is nu zeer laag. Er zullen meerdere ongezette stappen geteld worden tot de grens opnieuw is veranerd. Om dit tegen te gaan moet de waarde gebruikt om de drempelwaarde te berekenen groter zijn dan twee plus de versnelling in rust. Er werd eerst drie in plaats van twee gekozen, maar dan ondstonden moeilijkheden bij testen met het toestel vlak in de hand.

3 Mobiele applicatie

Voor het schrijven van de Android app werd een applicatie van de begeleiders waarin een eerste basis reeds geschreven stond, ter beschikking gesteld. Deze is voorzien van twee tabbladen; een tabblad waar de stappen moesten worden weergegeven en een tabblad waar de accelerometer output opgeslagen wordt. Vervolgens werd de app uitgebreid met het eerder besproken algoritme, het aantal stappen, het aantal calorieën, afgelegde afstand en een progressiecirkel.[1] Oorspronkelijk was er ook de ambitie om meer toepassingen te implementeren, maar omwille van de beperkte tijd is niet alles verwezelijkt.



Figuur 3.1: Screenshot van de applicatie

1 Android Studio

Voor het schrijven van de applicatie werd de programmeeromgeving Android Studio gebruikt. Android Studio is de officiële omgeving voor het ontwikkelen van Android-applicaties. Deze programmeeromgeving simuleert Android-apparaten op een computer, zodat toepassing op verschillende apparaten en Android API-niveaus kunnen getest worden zonder dat een fysiek apparaat nodig is. De code wordt geschreven in Java.

2 Aantal stappen weergeven

Het aantal stappen is een object dat zichtbaar is op de UI. Dit houdt in dat het aantal stappen wordt weergegeven als een "TextView".[2] Een TextView is een Android gebonden object dat bedoeld is voor het weergeven van tekst en optioneel het bewerken van tekst door de gebruiker. Een TextView wordt aangemaakt in de grafische editor van Android Studio. Vervolgens is deze gelinkt met de code, zodanig dat bij iedere stap die geregistreerd wordt dit aantal met één verhoogd wordt.

3 Aantal calorieën weergeven

Het aantal calorieën wordt op analoge manier weergegeven in de UI. Dit aantal wordt berekend afhankelijk van het aantal stappen, dit via het lineair verband dat stelt dat iedere stap 0.04 calorieën is.[3]

4 Afgelegde afstand weergeven

Net zoals het aantal calorieën werd de afgelegde afstand op analoge wijze geïmplementeerd. De afgelegde afstand is ook een lineair verband, waarbij 1333 stappen het equivalent zou zijn van één kilometer.[4]

5 Progressiecirkel weergeven

Android Studio beschikt zelf over een ingebouwde progressiecirkel. Standaard is deze progressiecirkel ingesteld als een cirkelvormige laadbalk die blijft ronddraaien zolang er een bewerking bezig is, net zoals vaak op populaire apps wordt gedaan. Na het programmeren van de progressiecirkel en de functionaliteit te veranderen naar het weergeven van progressie, bleek de progressiecirkel niet langer cirkelvormig te zijn,

maar balk vormig. De cirkelvorm moeten worden gecreëerd via een XML-bestand als drawable bron. Een drawable bron is een algemeen concept voor een afbeelding of object die op het scherm kan worden weergegeven. Zo'n bron kan op verschillende objecten worden toegepast, zodanig dat deze slechts eenmalig dient gedefiniëerd te worden. Voor het creëren van de progressiecirkel werden twee van zulke bronnen gedefinieerd, voor de cirkelvorm en voor de kleuren. De progressiecirkel verkleurt ook naarmate er meer stappen worden gezet, beginnend bij donkerrood waarna hij telkens groener kleurt. Dit natuurlijk omdat mensen de kleur rood associëren met "slecht, het doel is niet voltooid", en groen daaraantegen met "goed, het doel is voltooid".

6 Bespreking van de code

De volledige app bestaat uit verschillende classes. Er zijn drie classes gelinkt aan het hoofdscherm. Zo is er bijvoorbeeld "HomeFragment" dit is waar de calorieën, de totale afstand en progressiecirkel worden weergegeven. Er is ook een class genaamd "BetereDetector" en "DummyDetector", BetereDetector is de class waar het algoritme is geschreven dat stappen telt. DummyDetector was één van onze eerste algoritmes voor het tellen van stappen, deze is minder nauwkeurig en wordt dan ook niet meer gebruikt in de app, zoals eerder al besproken. Deze class staat nog in de code omdat voor het testen de beide detectoren moesten worden vergeleken. Op die manier kan snel gewisseld worden tussen detectoren.

7 Reset knop weergegeven

De reset knop is een object genaamd "Button". Dit is een object waarop gedrukt kan worden die in respons een voorop geprogrammeerde functie uitvoert. In ons programma zet deze knop het aantal stappen terug op nul, net zoals de afstand, aantal calorieën en de progressiecirkel. Het idee is dat gebruikers dan bij het aanvangen van een nieuwe dag, of eender welk moment naar keuze de teller opnieuw kunnen laten beginnen.

4 Resultaten

Na afloop van de labsessies is er nu een UI met functionerende stappenteller. De grootst gemeten foutmarge is 6 procent.

1 Resultaten van de detectoren

Hoe accuraat de detector is, hangt af van de manier waarop het toestel wordt vastgehouden. De onderstaande tabellen tonen de exacte bevindingen met onze twee detectoren voor verschillende manieren waarop het toestel kan worden vastgehouden.

Dit deel is beperkt tot het toestel in de broekzak en jaszak, aangezien hiermee interessante gegevens werden verkregen om te vergelijken.

1.1 *Toestel in broekzak*

Tabel 4.1: Toestel in broekzak

Dynamische detector	Statische detector
52	46
51	48
49	45
48	45

Het rekenkundig gemiddelde wordt gegeven door 4.1.

$$AM = \frac{1}{N}\sum_{i=1}^{N} a_i = \frac{a_1 + a_2 + \ldots + a_N}{N} \qquad (4.1)$$

De AM komt van de Engelse term; arithmetic mean. Dit wordt toegepast op de bevindingen.

- Toegepast op de dynamische detector: $AM_{dy} = 50$
- Toegepast op de statische detector: $AM_{st} = 46$

De root mean squared error wordt gegeven door 4.2.

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{\theta}_i - \theta_i)^2} \qquad (4.2)$$

Hierbij is $\hat{\theta}$ het aantal gemeten stappen en θ het aantal gezette stappen.

- Toegepast op de dynamische detector: $RMSE_{dy} = \sqrt{2,5}$
- Toegepast op de statische detector: $RMSE_{st} = \sqrt{17,5}$

Deze formules - 4.1 en 4.2 - toepassen op data is interessant om resultaten beter te kunnen interpreteren.

1.2 *Toestel in de jaszak*

Tabel 4.2: Toestel in jaszak

Dybamische detector	Statische detector
52	26
50	25
49	23
48	27

Hier worden eveneens de bewerkingen van 4.1 en 4.2 toegepast.

- Toegepast op de dynamische detector: $AM_{dy} = 49,75$ en $RMSE_{dy} = \sqrt{2,25}$
- Toegepast op de statische detector: $AM_{st} = 45,25$ en $RMSE_{st} = \sqrt{614,75}$

2 Bespreken van deze bevindingen

Zoals te zien in tabel 1.1 is de statische detector - dummydetector - nog behoorlijk correct

met het toestel in de broekzak, met een rekenkundig gemiddelde van 46. Dit omdat elke stap zeer duidelijke pieken oplevert. Toen de detector voor het eerst gestest werd - met het toestel in de achterzak - werden altijd bijna perfecte resultaten met een zeer laag foutmarge gevonden. De resultaten wanneer het toestel in de hand gehouden wordt of in de jaszak gestopt wordt daaraantegen, zijn een stuk minder accuraat. De dynamische detector zorgt voor lagere foutmarges in elke context.

3 Vergelijking met een andere applicatie

De stappenteller applicatie zoals hier geschreven, is vergeleken met een betalende uit de Play Store; StepsApp.[5] De keuze van deze specifieke applicatie om tegen te testen is gemaakt aangezien StepsApp bovenaan stond in de app store, en meer dan 20 miljoen downloads heeft. De app is in principe betalend, dus moest een free trial gestart worden. Deze werd vergeleken met onze dynamische detector, dit is namelijk de detector gebruikt in onze applicatie. Het toestel wordt in de hand gehouden, aangezien dat de moeilijkste test is met de meeste fouten. Het rekenkundig gemiddelde en de root mean squared error worden op de data toegepast om goed te kunnen vergelijken.

Tabel 4.3: Toestel in hand.

StepsApp resultaten	Onze dynamische detector resultaten.
57	47
50	50
48	51
54	49

$AM_{StepsApp} = 53$ en $RMSE_{StepsApp} = \sqrt{69}$.

$AM_{dy} = 49,25$ en $RMSE_{dy} = \sqrt{2,75}$.

Natuurlijk had de StepsApp wel een meer uitgebreide interface en allerlei andere opties, maar na vijf testen lijkt het hier ontwikkelde dynamische algoritme voor superieure resultaten te zorgen. Het is niet uitgesloten dat met meer uitgebreide testen andere resultaten hadden bekomen.

Bibliografie

[1] "Material Design for Android," Android Developers.
https://developer.android.com/develop/ui/views/theming/look-and-feel
(Geraadpleegd 12 Dec. 2022).

[2] *"Styles and Themes,"* Android Developers
https://developer.android.com/develop/ui/views/theming/themes
(Geraadpleegd 12, Dec. 2022).

[3] "Stappenchallnge of geen stappenchallenge, that's the question"foodlive.nl
https://www.foodilove.nl/10-000-stappen/#:~:text=Afhankelijk%20van%20je%20gewicht%20verbrand,calorie%C3%ABn%20per%20dag%20door%20beweging. (Geraadpleegd 20, Dec. 2022)

[4] "Aantal stappen omzetten naar kilometers - Hardlopen," Wim Groenendijk.
https://www.groenendijkwim.nl/hardlopen/stappen-naar-kilometers/#:~:text=Hoeveel%20stappen%20is%20een%20kilometer
(Geraadpleegd 12, Dec. 2022).

[5] "StepsApp," steps.app, 2015.
https://steps.app/nl
(Geraadpleegd 06, Dec. 2022).