Mobilna aplikacija za brojanje koraka

Jan, Andrea

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
Diplomski studij računarstva

Diplomski rad

Mobilna aplikacija za brojanje koraka

Rijeka, siječanj 2024.

Andrea Jan
0035216080

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
Diplomski studij računarstva

Diplomski rad

Mobilna aplikacija za brojanje koraka

Mentor: prof. dr. sc. Mladen Tomić

Rijeka, siječanj 2024.

Andrea Jan
0035216080

Umjesto ove stranice umetnuti zadatak za završni ili diplomski rad

Izjava o samostalnoj izradi rada

Izjavljujem da sam samostalno izradila ovaj rad.

Rijeka, siječanj 2024.

_____
Andrea Jan

Zahvala

Zahvaljujem mentoru prof. dr. sc. Mladenu Tomiću na podršci tijekom pisanja ovoga rada i korisnim raspravama i savjetima.

Sadržaj

Popis slika

Popis tablica

Poglavlje 1

Uvod

U posljednjem desetljeću, pametni telefoni postali su nepobitni čimbenik u komercijalnom, društvenom i tehničkom smislu, pružajući korisnicima pristup raznovrsnim uslugama i mogućnostima. Ovi uređaji sastavni su dio svakodnevnog života, dostupni gotovo svakoj osobi i imaju ključnu ulogu u načinu na koji komuniciramo, radimo i zabavljamo se. Jedna od ključnih karakteristika modernih pametnih telefona su ugrađeni inercijski senzori, uključujući akcelerometre, žiroskope i magnetometre, koji čine Inercijsku Mjernu Jedinicu (IMU). Ovi senzori omogućuju uređaju da bilježi informacije o svojoj poziciji, orijentaciji i kretanju, čime se otvaraju široke mogućnosti primjene.

Detekcija i brojanje koraka imaju duboku upotrebu u različitim aspektima današnjeg društva. Od praćenja tjelesne aktivnosti i unaprjeđenja fizičke kondicije, preko praćenja pješačkih rutina u svrhu navigacije, do analize koraka u medicinske svrhe, ovaj koncept postaje ključan za mnoge aplikacije. U domeni zdravlja i fitnessa, brojanje koraka postaje sve popularnije, potičući ljude na održavanje aktivnog načina života. Aplikacije za praćenje tjelesne aktivnosti, zajedno s pametnim narukvicama i uređajima za nošenje, koriste detekciju koraka kako bi pružile korisnicima informacije o njihovim svakodnevnim aktivnostima, poput prijeđenih udaljenosti i potrošenih kalorija. U području navigacije, detekcija koraka ima ključnu ulogu u praćenju korisnika u zatvorenim prostorima gdje GPS signal može biti nedostupan. Tehnike poput Pedestrian Dead Reckoning (PDR) koriste detekciju koraka za procjenu trenutne pozicije korisnika, što je od vitalnog značaja

za aplikacije poput navigacije u trgovačkim centrima, aerodromima ili bolnicama. Medicinska primjena brojanja koraka uključuje praćenje koraka pacijenata kako bi se procijenila njihova rehabilitacija ili tjelesna aktivnost. Osim toga, analiza koraka može pomoći u dijagnosticiranju različitih medicinskih stanja koja utječu na hod i ravnotežu pacijenata.

Ukratko, detekcija i brojanje koraka putem pametnih telefona i inercijskih senzora otvaraju vrata mnogim inovativnim aplikacijama koje poboljšavaju naše zdravlje, kvalitetu života i svakodnevno iskustvo.

U sklopu ovog diplomskog rada, istražit će se upotreba inercijskih senzora pametnih telefona. Proučit će se različite metode i tehnike koje se koriste za svrhu detekcije i brojanja koraka i istražiti njihove prednosti i ograničenja. Također, razvit će se i analizirati Android aplikacija za brojanje koraka koristeći akcelerometar pametnog telefona.

Poglavlje 2

Inercijski senzori pametnog telefona

Senzori pametnog telefona su uređaji koji mjere različite fizičke veličine, kao što su sila koja utječe na uređaj, količina svjetlosti koja dopire do površine ili temperatura u okolini. Ovo su osnovni primjeri fizičkih svojstava koja senzori mogu detektirati. Većina Android telefona opremljena je naprednim senzorima koji su u mogućnosti bilježiti važne informacije, poput relativne vlažnosti, atmosferskog tlaka, magnetskog polja, broja koraka, brzine rotacije uređaja duž osi X, Y i Z, udaljenosti od objekta i mnogo drugih parametara. Većina ovih senzora su MEMS (Mikro Elektro Mehanički Senzori) koji se proizvode u malim razmjerima, obično na silicijskim čipovima, s integriranim mehaničkim i električnim komponentama. Osnovni princip rada MEMS senzora temelji se na mjerenju promjena električnih signala uzrokovanih mehaničkim pokretom. Ove promjene u električnim signalima pretvaraju se u digitalne vrijednosti pomoću električnih krugova. Akcelerometar i žiroskop primjeri su MEMS senzora koji se često koriste. Važno je napomenuti da većina senzora na Android telefonima troši minimalnu količinu baterije i procesorske snage. [1]

Senzori se općenito mogu svrstati u dvije glavne kategorije:

- **Fizički senzori**: Ovi senzori su stvarni hardverski uređaji koji se fizički nalaze na mobilnom uređaju. Poznati su i kao hardverski senzori. Primjeri fizičkih senzora uključuju akcelerometre, žiroskope i magnetometre. Akcelerometar mjeri ubrzanje uređaja, žiroskop mjeri brzinu rotacije, dok

magnetometar mjeri magnetsko polje okoline.

- **Sintetski senzori**: Sintetski senzori, poznati i kao virtualni, sastavljeni ili softverski senzori, nisu stvarni hardverski uređaji na uređaju, već se njihovi podaci izvode iz jednog ili više fizičkih senzora. Na primjer, senzor za gravitaciju pruža informacije o smjeru gravitacijske sile na uređaju, senzor za linearno ubrzanje mjeri promjene brzine bez gravitacijskog utjecaja, a senzor za detekciju koraka identificira korake korisnika.

Važno je napomenuti da platforma Android ne pravi značajnu razliku između fizičkih senzora i sintetskih senzora tijekom njihovog korištenja. Ova podjela uglavnom ima teorijski značaj za razumijevanje porijekla vrijednosti koje senzori pružaju. Oba tipa senzora omogućuju razvoj aplikacija i funkcija koje obogaćuju iskustvo korisnika na Android uređajima. [1]

Vrijednosti senzora mogu se općenito podijeliti u tri kategorije, svaka s vlastitim karakteristikama:

- **Sirove (nekalibrirane) vrijednosti**: Sirove vrijednosti senzora neposredno potječu od samih senzora, i operativni sustav ih prenosi aplikacijama bez dodatne korekcije. Ovi podaci najbliži su onome što senzor izvorno izmjeri. Primjeri senzora koji pružaju sirove vrijednosti uključuju akcelerometre, senzore blizine, senzore svjetlosti i barometre. Ove vrijednosti korisne su za aplikacije koje zahtijevaju neposredan uvid u senzorske podatke.
- **Kalibrirane (prerađene) vrijednosti**: Kalibrirane vrijednosti izračunavaju se od strane operativnog sustava primjenom dodatnih algoritama za korekciju. To uključuje kompenzaciju pogrešaka kao što su drift i pristranost te uklanjanje šuma iz sirovih senzorskih podataka. Na primjer, senzori za detekciju koraka, brojanje koraka i značajno kretanje koriste kalibrirane vrijednosti temeljene na sirovim podacima dobivenim od akcelerometra. Magnetometar i žiroskop posebne su vrste senzora koje često pružaju i sirove i kalibrirane vrijednosti.
- **Fuzirane (kombinirane) vrijednosti**: Fuzirane vrijednosti proizlaze iz kombinacije podataka iz dva ili više senzora. Ove vrijednosti se često iz-

računavaju kako bi se iskoristila prednost jednog senzora za kompenzaciju ograničenja drugih senzora. Na primjer, senzori poput gravitacije i linearnog ubrzanja koriste podatke iz akcelerometra i žiroskopa kako bi pružili fuzirane vrijednosti. Ovo je posebno korisno u aplikacijama gdje je potrebno precizno praćenje orijentacije uređaja i dinamičkih pokreta.

Ova podjela senzorskih vrijednosti pomaže razumjeti kako se senzorski podaci obrađuju prije nego što budu dostupni različitim aplikacijama. Ovisno o specifičnim potrebama aplikacija i zahtjevima, razvijatelji mogu odabrati odgovarajući tip senzorskih vrijednosti za svoje aplikacije kako bi postigli željene funkcionalnosti i preciznost.

Platforma Android podržava tri široke kategorije senzora:

- Senzori pokreta
- Senzori okoline
- Senzori položaja

Svaka od ove tri kategorije pod sobom ima veći broj senzora koje zajedno kategorizira slična funkcionalnost, ali se njihova implementacija razlikuje. [2]

2.1 Senzori pokreta

Senzori pokreta vrše precizna mjerenja ubrzanja i rotacije duž tri osi uređaja. Ova kategorija senzora obuhvaća različite tipove, uključujući akcelerometre, senzore gravitacije, žiroskope i senzore rotacijskog vektora. Platforma Android pruža raznolik izbor senzora za praćenje kretanja uređaja, pri čemu se arhitekture senzora razlikuju ovisno o njihovoj vrsti.

Na primjer, senzori gravitacije, linearnog ubrzanja, rotacijskog vektora, značajnog gibanja, brojača koraka i detektora koraka mogu biti implementirani kao hardverski senzori ili kao softverski senzori, a to ovisi o specifičnom uređaju. S druge strane, senzori akcelerometra i žiroskopa uvijek su hardverski temeljeni.

Senzori pokreta odgovorni su za mjerenje sila koje mogu pokrenuti kretanje

uređaja duž x, y i z osi. To kretanje može biti linearno ili rotacijsko u bilo kojem smjeru. Većina ovih senzora pruža podatke za x, y i z osi, dok senzor rotacijskog vektora dodatno pruža informaciju o četvrtoj osi, što je skalarna komponenta rotacijskog vektora. Ovi senzori igraju ključnu ulogu u praćenju kretanja uređaja i omogućuju korisnicima interakciju s aplikacijama putem gesti i pokreta. [2]

Većina modernih Android uređaja danas je opremljena akcelerometrom, a mnogi čak uključuju i žiroskop. Dostupnost softverskih senzora varira jer oni često ovise o jednom ili više hardverskih senzora kako bi dobili svoje podatke. Ovisno o uređaju, ovi softverski senzori mogu dohvatiti svoje podatke ili iz akcelerometra i magnetometra, ili iz žiroskopa.

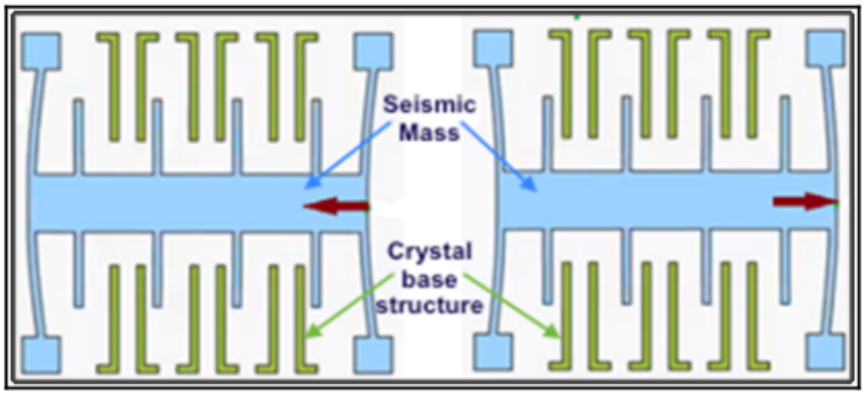
Senzori pokreta imaju široku primjenu za praćenje kretanja uređaja, uključujući naginjanje, trešnju, rotaciju ili ljuljanje. Kretanje obično odražava izravan korisnički unos (na primjer, korisnik upravlja automobilom u igri ili upravlja loptom), ali također može biti posljedica fizičkog okruženja u kojem se uređaj nalazi, poput kretanja uređaja dok se vozi automobilom.

U prvom slučaju, nadzire se kretanje u odnosu na referentni sustav uređaja ili aplikacije, dok u drugom slučaju prati se kretanje u odnosu na svjetski referentni sustav. Sami senzori pokreta obično nisu dizajnirani za praćenje položaja uređaja, ali se često kombiniraju s drugim senzorima, poput senzora geomagnetskog polja, kako bi se odredio položaj uređaja u odnosu na globalni okvir referencije oko uređaja. [1]

Tablica 2.1 prikazuje upotrebu, vrste i potrošnju energije senzora pokreta.

2.1.1 Akcelerometar

S obzirom da je u ovom radu korišten akcelerometar, bit će detaljnije objašnjen u nastavku. Senzor akcelerometra služi za mjerenje ubrzanja duž x, y i z osi koja utječe na uređaj. Izmjereno ubrzanje uključuje fizičko ubrzanje (promjenu brzine) i stalnu gravitaciju koja djeluje na telefon neprekidno. Senzori akcelerometra temelje se na tehnologiji Mikro Elektro Mehaničkog Sustava (MEMS), što je ugradbeni sustav koji integrira elektroničke i mehaničke komponente na izuzetno maloj skali. Osnovni princip rada akcelerometra u pametnom telefonu temelji se na pomaku mikroskopskih kristalnih ploča, poznatih kao seizmička masa (*eng. Seismic Mass*), duž x, y i z osi preko mikroskopske baze kristala (*eng. Crystal base structure*), prikazano na slici 2.1. Kada korisnik generira silu koja pomiče telefon, seizmička masa također se pomakne u odnosu na baznu strukturu smještenu između ploča seizmičke mase, što rezultira promjenom kapaciteta. Ta promjena kapaciteta pretvara se u silu pomoću električnih krugova. Dodatni algoritmi primjenjuju se kako bi se kalibrirale vrijednosti akcelerometra kako bi se kompenzirale temperature, odstupanja i skale. Vrijednost koju senzor akcelerometra prijavljuje izražena je u SI jedinicama (m/s^2). Kada je telefon miran i ne kreće se, senzor akcelerometra prikazuje ubrzanje od 9.81 m/s^2, što predstavlja gravitacijsku silu koja djeluje na telefon. Kada je telefon u slobodnom padu, senzor prikazuje nulu ubrzanja jer nema dodatnih sila koje djeluju na telefon. [1]

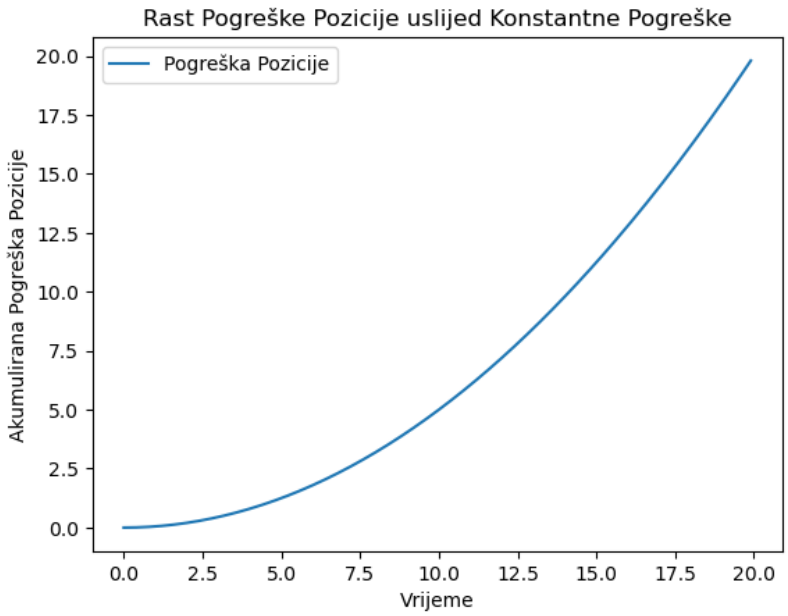


Slika 2.1 Akcelerometar. Preuzeto s [1]

Pogreške u mjerenju

Pri mjerenju ubrzanja pomoću akcelerometra u mobilnim uređajima, suočavamo se s raznim faktorima koji mogu utjecati na preciznost i pouzdanost dobivenih podataka. Ključne izvore pogrešaka možemo podijeliti u nekoliko skupina:

- **Pristranost (*eng. bias*)**: Pristranost akcelerometra predstavlja konstantno odstupanje izlaznog signala od stvarne vrijednosti, izraženo u m/s^2. Konstantna pristranost pogreške, kada se dvostruko integrira, uzrokuje pogrešku u položaju koja kvadratno raste s vremenom, prikazano na slici 2.2. Moguće je procijeniti pristranost mjerenjem prosjeka izlaznog signala akcelerometra kada nije izložen ubrzanju. No, ovo postaje izazovno zbog prisutnosti gravitacije koja stvara dodatnu komponentu pristranosti. Precizna orijentacija uređaja u odnosu na gravitacijsko polje postiže se kalibracijskim rutinama, koje obično uključuju postavljanje uređaja na kontroliranu okretnu ploču. Kalibracija omogućuje mjerenje i korekciju pristranosti, poboljšavajući tako preciznost akcelerometra u aplikacijama gdje točnost položaja ima ključnu ulogu.
- **Termo-mehanički bijeli šum / slučajni hod brzine**: Različiti izvori šuma, uključujući vibracije i elektromagnetske interferencije, mogu pridonijeti sitnim fluktuacijama u mjerenjima ubrzanja. Rezultati dobiveni iz akcelerometra odstupaju zbog sekvence bijelog šuma koji generira slučajni hod brzine. Slučajni hod brzine odnosi se na nepredvidive fluktuacije u brzini koje proizlaze iz stohastičkih promjena u ubrzanju tijekom kretanja. Na slici 2.3 prikazan je učinak koji bijeli šum akcelerometra ima na izračunatu poziciju, gdje se provelo dvostruko integriranje uzoraka dobivenih iz akcelerometra. Bijeli šum akcelerometra stvara drugi red slučajnog hoda u položaju, s nultom srednjom vrijednošću i standardnom devijacijom koja raste proporcionalno s $t^{3/2}$.
- **Flicker šum**: MEMS akcelerometri podložni su "flicker" šumu koji uzrokuje fluktuacije u pristranosti tijekom vremena. Takve fluktuacije obično se modeliraju kao slučajni hod pristranosti. Koristeći ovaj model, "flicker" šum stvara slučajni hod druge razine u brzini čija nesigurnost raste propor-

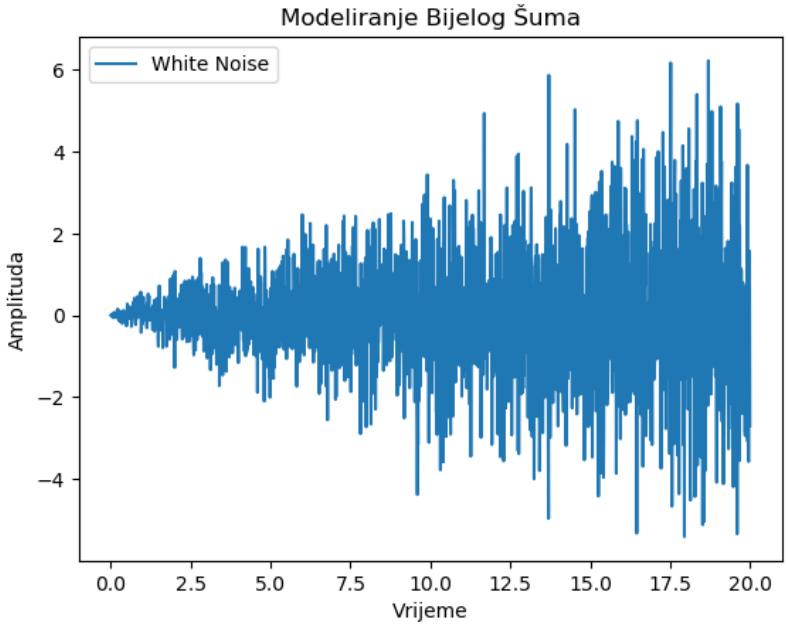


Slika 2.2 Pogreška u položaju kod pristranosti

cionalno s $t^{3/2}$, te slučajni hod treće razine u položaju čija nesigurnost raste proporcionalno s $t^{5/2}$.

- **Utjecaj temperature**: Promjene temperature zbog promjena u okolišu i samog grijanja senzora uzrokuju fluktuacije u pristranosti izlaznog signala. Odnos između pristranosti i temperature ovisi o specifičnom uređaju, no često je izrazito nelinearan. Ako IMU sadrži senzor temperature, moguće je primijeniti korekcije na izlazne signale kako bi se kompenzirali učinci ovisni o temperaturi.
- **Kalibracija**: Greške u kalibraciji, poput onih u faktorima razmjera, poravnanjima i linearnostima izlaza, manifestiraju se kao pristranosti koje postaju vidljive samo tijekom ubrzanja uređaja. Važno je napomenuti da se ove "privremene" pristranosti mogu primijetiti čak i kada je uređaj miran, zbog djelovanja gravitacijskog ubrzanja.

Razumijevanje ovih pogrešaka ključno je za pravilno tumačenje podataka iz akcelerometara u mobilnim uređajima, a korekcije ili kalibracije mogu biti potrebne



Slika 2.3 Drugi red slučajnog hoda u položaju

ovisno o primjeni. [3]

Senzor	**Tip**	**Vrijednost**	**Temeljni senzori**	**Opis**	**Upotreba**	**Potrošnja**
Akcelerometar	Fizički	Sirova	Akcelerometar	Mjeri ubrzanje sila duž x, y i z osi (uključujući gravitaciju). Jedinica: m/s^2	Može se koristiti za otkrivanje pokreta kao što su podrhtavanje, njihanje, naginjanje i fizičke sile koje se primjenjuju na telefon	Niska
Gravitacija	Sintetski	Fuzirana	Akcelerometar, žiroskop	Mjeri silu gravitacije duž x, y i z osi. Jedinica: m/s^2	Može se koristiti za otkrivanje kada je telefon u slobodnom padu	Srednja
Linearna akceleracija	Sintetski	Fuzirana	Akcelerometar, žiroskop	Mjeri ubrzanje sila duž x, y i z osi (uključujući gravitaciju). Jedinica: m/s^2	Može se koristiti za otkrivanje pokreta kao što su podrhtavanje, njihanje, naginjanje i fizičke sile koje se primjenjuju na telefon	Srednja
Žiroskop	Fizički	Sirova, kalibrirana	Žiroskop	Mjeri brzinu rotacije uređaja duž x, y i z osi. Jedinica: rad/s	Može se koristiti za otkrivanje rotacijskih pokreta poput vrtnje, okretanja i svakog kutnog kretanja telefona	Srednja
Detektor koraka	Sintetički	Kalibrirana	Akcelerometar	Detektira korake	Može se koristiti za otkrivanje početka hodanja korisnika	Niska

Brojač koraka	Sintetički	Kalibrirana	Akcelerometar	Mjeri broj koraka koje je korisnik napravio od posljednjeg ponovnog pokretanja dok je senzor bio aktivan	Prati broj koraka korisnika po danu	Niska
Značajno gibanje	Sintetički	Kalibrirana	Akcelerometar	Otkriva kada postoji značajno gibanje na telefonu zbog hodanja, trčanja ili vožnje	Detektira značajno gibanje	Niska
Rotacijski vektor	Sintetički	Fuzirana	Akcelerometar, žiroskop, magnetometar	Mjeri komponentu rotacijskog vektora duž x-osi $(x \cdot \sin(\theta/2))$, y-osi $(y \cdot \sin(\theta/2))$ i z-osi $(z \cdot \sin(\theta/2))$. Skalarna komponenta rotacijskog vektora je $(\cos(\theta/2))$. Bez jedinica	Može se koristiti u 3D igrama temeljenih na smjeru uređaja	Visoka

Tablica 2.1 Senzori pokreta

2.2 Senzori okoline

Senzori okoline odgovorni su za mjerenje svojstava okoline poput temperature, relativne vlažnosti, svjetlosti i zračnog tlaka u blizini telefona. Za razliku od senzora pokreta i položaja koji pružaju vrijednosti senzora u višedimenzionalnim nizovima, senzori okoline izvješćuju pojedinačne vrijednosti senzora. [2] Tablica 2.2 prikazuje upotrebu, vrste i potrošnju energije senzora okoline.

Senzor	**Tip**	**Vrijednost**	**Temeljni senzori**	**Opis**	**Upotreba**	**Potrošnja**
Temperatura	Fizički	Sirova	Termometar	Mjeri temperaturu okolnog zraka. Jedinica: Stupnjevi Celzijevi	Koristi se za praćenje temperatura.	Srednja
Svjetlost	Fizički	Sirova	Fotometar	Mjeri razinu okolnog svjetla (osvjetljenja). Jedinica: lx	Može se koristiti za prigušivanje svjetline zaslona telefona.	Niska
Barometar	Fizički	Sirova	Barometar	Mjeri tlak okolnog zraka. Jedinica: mPa ili mbar	Može se koristiti za mjerenje visine u odnosu na razinu mora	Srednja
Relativna vlažnost	Fizički	Sirova	Relativna vlažnost	Mjeri postotak relativne vlažnosti okolnog zraka. Jedinica: %	Može se koristiti za izračunavanje točke rosišta te apsolutne i relativne vlažnosti	Srednja

Tablica 2.2 Senzori okoline

2.3 Senzori položaja

Senzori položaja koriste se za određivanje stvarnog položaja telefona u odnosu na globalni koordinatni sustav. Na primjer, koristeći senzor geomagnetskog polja zajedno s akcelerometrom, može se odrediti položaj uređaja u odnosu na magnetski sjeverni pol. Orijentacijski senzor koristi se za utvrđivanje položaja uređaja u okviru aplikacije. Ovi senzori pružaju informacije o položaju uređaja duž x, y i z osi. [2] Tablica 2.3 prikazuje upotrebu, vrste i potrošnju energije senzora položaja.

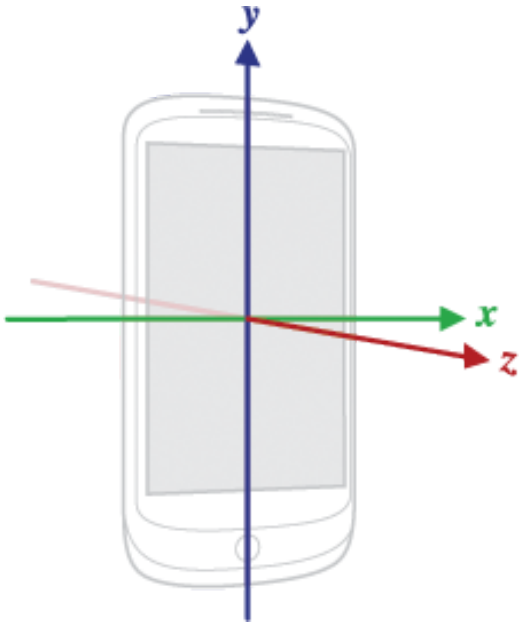
Senzor	**Tip**	**Vrijednost**	**Temeljni senzori**	**Opis**	**Upotreba**	**Potrošnja**
Magnetometar	Fizički	Sirova, kalibrirana	Magnetometar	Mjeri jačinu geomagnetskog polja duž x, y i z osi. Jedinica: T (mikrotesla)	Može se koristiti za izradu kompasa i preciznog izračunavanja stvarnog sjevera.	Srednja
Orijentacija (Zastarjelo)	Sintetički	Fuzirana	Akcelerometar, žiroskop, magnetometar	Mjeri azimut (kut oko z osi), nagib (kut oko x osi) i kotaciju (kut oko y osi). Jedinica: stupnjevi	Koristi se za detekciju položaja i orijentacije uređaja	Srednja
Približnost (*eng. Proximity*)	Fizički	Sirova	Približnost	Mjeri udaljenost objekta u odnosu na zaslon uređaja. Jedinica: cm	Koristi se za određivanje približava li se uređaj uhu osobe dok ga drži	Niska
Vektor Rotacije Igre (*eng. Game Rotation Vector*)	Sintetički	Fuzirana	Akcelerometar, žiroskop	Mjeri komponentu rotacijskog vektora duž x-osi ($x \cdot \sin(\theta/2)$), y-osi ($y \cdot \sin(\theta/2)$) i z-osi ($z \cdot \sin(\theta/2)$). To je skalarna komponenta rotacijskog vektora ($\cos(\theta/2)$). Bez jedinica	Koristi se u 3D igrama temeljenim na orijentaciji uređaja	Srednja

Senzor	**Tip**	**Vrijednost**	**Temeljni senzori**	**Opis**	**Upotreba**	**Potrošnja**
Geomagnetski Vektor Rotacije	Sintetički	Fuzirana	Akcelerometar, magnetometar	Mjeri komponentu rotacijskog vektora duž x-osi ($x \cdot \sin(\theta/2)$), y-osi ($y \cdot \sin(\theta/2)$) i z-osi ($z \cdot \sin(\theta/2)$). To je skalarna komponenta rotacijskog vektora ($\cos(\theta/2)$). Bez jedinica	Koristi se u aplikacijama proširene stvarnosti (*eng. augmented reality*) koje se temelje na orijentaciji uređaja i kompasa	Srednja

Tablica 2.3 Senzori položaja

2.4 Koordinatni sustav senzora

Većina senzora koristi standardni troosni koordinatni sustav za prikaz svojih vrijednosti. Ovaj koordinatni sustav može se usporediti s troosnim koordinatnim sustavom koji se koristi za mjerenje duljine, širine i visine bilo kojeg 3D objekta u prostoru, s razlikom u referentnom sustavu i orijentaciji troosnih osi. [1]



Slika 2.4 Koordinatni sustav senzora. Preuzeto s [1]

Kako je prikazano na slici 2.4, ishodište ovog koordinatnog sustava smješteno je u sredini zaslona uređaja. Kada je uređaj u svojoj zadanoj orijentaciji (obično okomito), x-os se proteže u horizontalnom smjeru, s pozitivnim vrijednostima na desnoj strani i negativnim vrijednostima na lijevoj strani. Slično tome, y-os se proteže vertikalno prema gore, dok se z-os pruža izvan zaslona uređaja. Točke iznad ishodišta u vertikalnom smjeru imaju pozitivne vrijednosti, dok one ispod ishodišta imaju negativne vrijednosti za y-os. Također, točke koje izlaze izvan zaslona uređaja imaju pozitivne vrijednosti, dok one koje su iza zaslona imaju negativne vrijednosti za z-os. Ovaj koordinatni sustav ključan je za interpretaciju senzorskih podataka i omogućuje precizno praćenje orijentacije i kretanja uređaja.

2.5 Android senzorski stog

Na slici 2.5 prikazani su slojevi u Android senzorskom stogu, svaki sa svojim specifičnim zadatkom i sposobnošću komunikacije s idućim slojem. Najviši sloj sastoji se od Android aplikacija, koje su korisnici podataka dobivenih od senzora. Sljedeći sloj je Android SDK (*eng. Software Development Kit*) sloj, koji omogućuje Android aplikacijama pristup senzorima. Ovaj sloj sadrži API-je za popis dostupnih senzora, registraciju senzora i sve druge funkcionalnosti vezane uz senzore. [1]

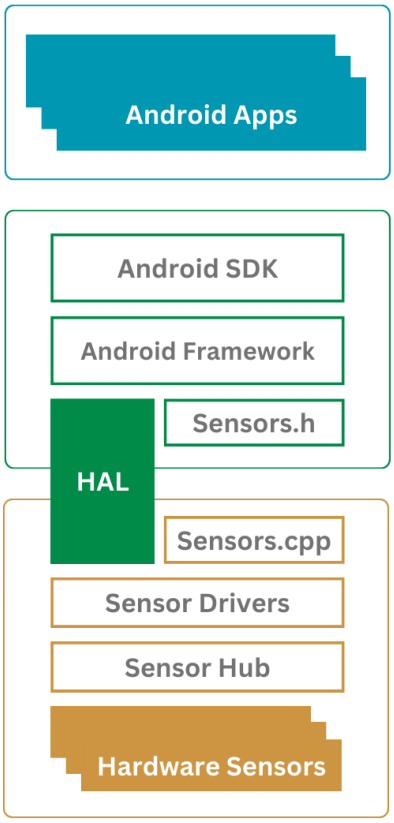
Treći sloj obuhvaća Android okvir (*eng. Framework)*, koji povezuje više aplikacija s jednim HAL (*eng. Sensors' Hardware Abstraction Layer*) klijentom. Okvir se sastoji od različitih komponenti koje omogućuju istodoban pristup više aplikacija senzorima.

Četvrti sloj, poznat kao HAL (*eng. Sensors' Hardware Abstraction Layer*), pruža sučelje između hardverskih upravljačkih programa senzora i Android okvira. HAL se sastoji od sučelja senzora i implementacije HAL-a, poznate kao `sensors.cpp`. Sučelje HAL-a definira Android i doprinositelji AOSP-a (*eng. Android Open Source Project*), dok je implementacija pružena od strane proizvođača uređaja.

Senzorski upravljački programi čine peti sloj stoga te su odgovorni za komunikaciju s fizičkim senzorima. U nekim situacijama, implementacija HAL-a i sami senzorski upravljački programi mogu biti isti softverski entitet, dok u drugim slučajevima proizvođači čipova senzora razvijaju upravljačke programe.

Senzorski čip (*eng. Sensor Hub*) predstavlja šesti, neobavezni sloj stoga. Obično se sastoji od odvojenog, specijaliziranog čipa za obavljanje niskorazinske računalne obrade s niskom potrošnjom energije, dok je aplikacijski procesor u načinu mirovanja. Senzorski čip često se koristi za grupiranje senzorskih podataka i dodavanje hardverskog FIFO reda.

Sedmi i posljednji sloj čini fizički hardverski senzor. Većinom su izrađeni od MEMS (*eng. Micro Electro Mechanical Systems*) silicijskog čipa i obavljaju stvarna mjerenja. [4]



Slika 2.5 Slojevi Android senzorskog stoga

2.6 Komponente senzorskog okvira

Android pruža niz metoda, klasa i sučelja za pristup senzorima i njihovim podacima dostupnim na Android uređajima. Ova skupina metoda, klasa i sučelja zajedno se naziva senzorskim okvirom (*eng. sensor framework*) i čini dio `android.hardware` paketa. Senzorski okvir sastoji se od četiri ključne komponente: `SensorManager`, `Sensor`, `SensorEvent` i `SensorEventListener`.

`SensorManager` glavna je klasa unutar okvira koja omogućuje aplikacijama da zatraže informacije o senzorima i registriraju se za primanje podataka s tih senzora. Kada su aplikacije registrirane, vrijednosti senzorskih podataka šalju se sučelju `SensorEventListener` u obliku klase `SensorEvent` koja sadrži informacije generirane od određenog senzora. [2]

2.6.1 SensorManager

`SensorManager` ključna je klasa za pristup senzorima na uređaju. Stvara instancu sustava senzora i pruža API-je za informacije o senzorima, uključujući popis dostupnih senzora i konstante za izvještavanje o preciznosti, razdoblju uzorkovanja i kalibraciji senzora. Osim toga, `SensorManager` omogućuje registraciju i odjavu slušatelja događaja senzora za pristup određenom senzoru.

2.6.2 SensorEventListener

`SensorEventListener` sučelje je s dvije metode za obradu senzorskih događaja. Prva metoda, `OnSensorChanged()`, aktivira se kada se senzorske vrijednosti promijene i šalje te vrijednosti putem objekta SensorEvent, predanog kao parametar. Druga metoda, `OnAccuracyChanged()`, obavještava o promjeni preciznosti senzora i pruža nove preciznosti putem cjelobrojnih vrijednosti. Podržane su četiri konstante cjelobrojne preciznosti u SensorManageru: `SENSOR_STATUS_ACCURACY_HIGH`, `SENSOR_STATUS_ACCURACY_MEDIUM`, `SENSOR_STATUS_ACCURACY_LOW`, `SENSOR_STATUS_ACCURACY_UNRELIABLE`

2.6.3 Sensor

Klasa `Sensor` koristi se za stvaranje instance određenog senzora. Ova klasa pruža različite metode koje omogućuju utvrđivanje mogućnosti senzora.

2.6.4 SensorEvent

`SensorEvent` posebna je vrsta klase koju operativni sustav koristi kako bi obavijestio slušatelje o promjenama u vrijednostima senzora. Objekt `SensorEvent` sadrži sljedeće četiri elementa:

- `values[]`: višedimenzionalno polje koje sadrži vrijednosti senzora
- `timestamp`: vrijeme u nanosekundama kada se događaj dogodio
- `accuracy`: jedna od četiri cjelobrojne konstante preciznosti

- `sensor`: vrsta senzora

Poglavlje 3

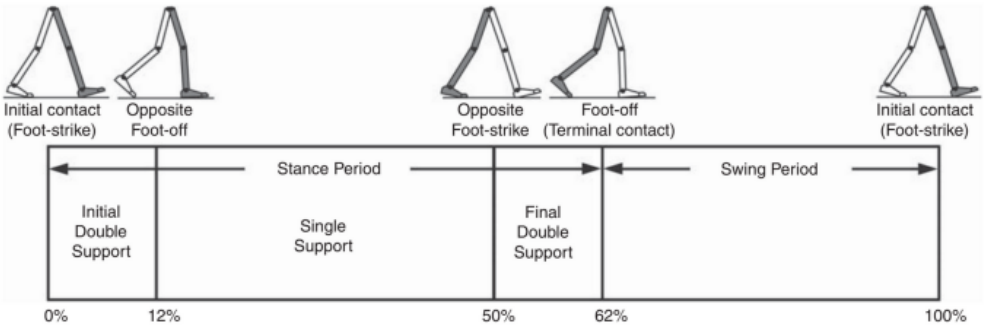
Detekcija koraka

Detekcija koraka tehnika je koja omogućava identifikaciju koraka osobe tijekom hodanja, trčanja ili drugih aktivnosti koje uključuju kretanje. Iako na prvi pogled može izgledati kao jednostavna zadaća, precizno detektiranje koraka ima duboke implikacije u različitim područjima, uključujući biomehaniku, sportsku analizu, praćenje zdravstvenih parametara, sigurnost i mobilnu tehnologiju.
U biomehanici, detekcija koraka pomaže znanstvenicima i istraživačima razumjeti obrasce kretanja, analizirati promjene u hodanju uzrokovane ozljedama ili bolestima te optimizirati učinkovitost u sportskim disciplinama. U medicini, precizna detekcija koraka može se koristiti za praćenje pacijenata s neurološkim poremećajima poput Parkinsonove bolesti ili za procjenu rehabilitacije nakon ozljeda.
Razvoj detekcije koraka usko je povezan s napretkom tehnologije, posebno senzora i računalne obrade podataka. Tradicionalni pristupi koristili su jednostavne senzore poput akcelerometra i žiroskopa kako bi pratili promjene u ubrzanju tijekom hodanja. Međutim, suvremene metode koriste naprednije tehnologije, uključujući duboko učenje, strojno učenje i neuronske mreže, što omogućava precizniju i kontekstualniju detekciju koraka.

3.1 Analiza koraka

Hod se definira kao način kretanja ljudskog tijela. Analiza hoda je kvantitativno mjerenje i procjena ljudskog kretanja, uključujući hodanje i trčanje. Hodni ciklus definira se kao razdoblje od trenutka prvog kontakta (također nazvanog dodir stopala) stopala subjekta s tlom do trenutka sljedećeg prvog kontakta iste noge. Hodni ciklus može se podijeliti na dvije faze: fazu potpore i fazu zamaha, što je prikazano na Slici 3.1. U fazi potpore analizirano stopalo je u kontaktu s tlom, dok je u fazi zamaha stopalo izvan tla. Tijekom svake faze se također mogu definirati dvije vrste vremenskih intervala, pojedinačnu potporu i dvostruku potporu, koji opisuju razdoblje kada je tijelo podržano jednim ili s dvama udovima.
Hodni ciklus također može biti podijeljen na 8 funkcionalnih faza, poznatih kao Rancho klasifikacija [5], koje se odnose na kretanje noge tijekom jednog koraka:

- Početni kontakt - kada stopalo dodiruje tlo, počevši s kontaktom pete s tlom (dodir pete)
- Prihvaćanje opterećenja - prihvaćanje težine i razdoblje dvostruke potpore. Druga noga završava fazu potpore i započinje početni zamah
- Srednja faza - stopalo je dobro postavljeno na tlo kako bi podržalo kretanje suprotnog udova koji je u fazi zamaha (razdoblje jednostruke potpore)
- Terminalna faza - suprotni ud je na kraju faze zamaha (završava s početnim kontaktom suprotnog udova);
- Pred-zamah - analizirani ud počinje napuštati tlo. Završno razdoblje dvostruke potpore
- Početni zamah - počinje trenutak kada stopalo prestaje biti u kontaktu s tlom i započinje zamah. Karakterizira ga najveća fleksija koljena
- Srednji zamah - druga trećina zamaha. Počinje na kraju početnog zamaha i završava na početku terminalnog zamaha. Karakterizira ga maksimalna fleksija kuka
- Terminalni zamah - zadnja trećina razdoblja zamaha. Završava s početnim kontaktom stopala



Slika 3.1 Vremenska podjela hodnog ciklusa. Preuzeto s [5]

3.2 State-of-the-art algoritmi za detekciju koraka

Tijekom proteklih desetljeća, mnoge metode razvile su se s ciljem snimanja i analize ljudskog hoda. Istraživači su koristili različite tehnike, uključujući video snimanje pacijenata pri hodanju, mjerenje sile na podlozi, uporabu inercijskih senzora (IMU) i ostale nosive senzore kako bi proučavali ljudsku lokomociju. S razvojem pametnih telefona opremljenih IMU senzorima, istraživači su počeli koristiti te uređaje u svojim istraživanjima i otkrili nove načine primjene tog znanja o ljudskom hodu.

Danas su brojači koraka i sustavi za praćenje aktivnosti postali neka od najčešćih aplikacija na pametnim telefonima. Problemi vezani uz točnost GPS lokacije unutar zatvorenih prostora potaknuli su istraživače da traže alternativna rješenja za precizno određivanje lokacije unutar zatvorenih prostora. Senzori pametnih telefona pokazali su se kao pouzdana alternativa. Sve više aplikacija zahtijeva visoko precizne i mobilne sustave za detekciju koraka. Kako algoritmi s detekcijom koraka postaju sve složeniji, došlo je od velikog značaja da postanu precizniji kako bi se smanjile kumulativne pogreške.

Najmoderniji algoritmi mogu se razvrstati u dvije glavne kategorije, ovisno o metodama koje koriste. Te kategorije uključuju metode temeljene na parametrima i metode strojnog učenja.

3.2.1 Metode temeljene na parametrima

Najistraženije metode za detekciju koraka su metode temeljene na parametrima. Ove metode uključuju različite tehnike, kao što su detekcija vrhova i dolina, brojanje nula, korištenje Fourierove transformacije na vremenskom otvoru (STFT) za analizu signala, itd.. Ono što ih čini izazovnima je potreba za prilagodbom parametara kako bi se postigli optimalni rezultati u identifikaciji koraka. Na primjer, postavljanje praga ili definiranje frekvencijskih raspona za analizu zahtijeva pažljivo podešavanje kako bi se signal ispravno obradio. [6, 7]

Postavke praga

Metode temeljene na jednostavnim pragovima funkcioniraju tako da procjenjuju ispunjavaju li očitanja akcelerometra određeni prag, što uključuje jednostavne metode temeljene na pragu [8, 9, 10] i metode nulte brzine (Zero Velocity Update - ZUPT) [11, 12]. Jednostavne threshold metode postavljaju prag za određenu karakteristiku ubrzanja. Ove metode posebno su učinkovite kada se ubrzanje mjeri na stopalu, gdje hodanje uzrokuje velika i kratkotrajna ubrzanja, posebno pri udarcima pete. Međutim, postavljanje optimalnih pragova može biti izazovno jer vrijednosti praga variraju između korisnika, vrsta površina i obuće.

U situacijama gdje su pametni telefoni slobodno nošeni od strane korisnika umjesto da su učvršćeni, threshold-based metode, uključujući ZUPT, često ne postižu optimalne rezultate. ZUPT metode iskorištavaju činjenicu da je svaka noga tijekom normalnog hoda povremeno potpuno mirna, a akcelerometar mora bilježiti određeni period neaktivnosti tijekom tog vremena. Ovo postižu praćenjem perioda niske ili nulte brzine ubrzanja. Kada se ovi trenuci detektiraju, broj koraka se povećava za jedan, sugerirajući da je korisnik napravio korak.

Važno je napomenuti da se većina threshold-based metoda, uključujući ZUPT, obično bolje ponaša kada je akcelerometar montiran na nozi korisnika. Međutim, kada su akcelerometri ugrađeni u pametne telefone i korisnici ih koriste na različite načine, postavljanje praga može biti izazovno. Različite metode nošenja pametnog telefona i različiti korisnici mogu zahtijevati različite pragove kako bi se postigli precizni rezultati u brojanju koraka.

Detekcija vrhova i dolina te brojanje nula (*eng. zero-crossing counting*)

Pristupi temeljeni na detekciji vrhova i dolina te brojanje nula predstavljaju ključnu komponentu u detekciji koraka, posebice u kontekstu praćenja aktivnosti korisnika putem pametnih telefona. Ovi pristupi koriste periodične obrasce u magnitudama ubrzanja koje se javljaju tijekom ponavljajućih pokreta hodanja i trčanja. [6, 13]

Detekcija vrhova [14, 15, 16] temelji se na identifikaciji lokalnih maksimuma u magnitudi ubrzanja. Kada se detektira takav vrh, broji se jedan korak. Ova metoda je jednostavna i ima nisku složenost, ali kako bi se osigurala točnost brojanja, postavljeni su pragovi i ograničenja. Na primjer, koraci se broje samo ako magnituda ubrzanja prelazi određeni prag ili ako temporalna dinamika brzine hodanja odgovara očekivanoj brzini hodanja. Isto tako, ograničenja vertikalnog pomaka također mogu biti uključena kako bi se izbjeglo prebrojavanje tijekom promjena nagiba.

Detekcija dolina [17], slično kao i vrhova, identificira lokalne minimume u magnitudi ubrzanja. Kada se detektira takva dolina, također se broji korak. Ova metoda također uključuje pragove i ograničenja kako bi se osigurala točnost brojanja.

Brojanje nula [18, 19], s druge strane, analizira promjene u magnitudi ubrzanja i broji korake na temelju nultih prijelaza u očitanjima ubrzanja. Ova metoda je učinkovita za pronalaženje koraka, no također zahtijeva kalibraciju i podešavanje pragova kako bi se osigurala preciznost.

Iako su ovi pristupi niski po složenosti, važno je napomenuti da njihova učinkovitost može varirati ovisno o uvjetima korištenja, kao i o načinu nošenja uređaja. Preciznost brojanja koraka može opasti u situacijama kada korisnik mijenja način hodanja ili položaj uređaja, posebice ako se uređaj nosi slobodno. Stoga je ključno pažljivo kalibrirati i prilagoditi ove pristupe kako bi se postigli najbolji rezultati u praćenju koraka korisnika.

Korištenje transformacija

Fourierova transformacija na vremenskom otvoru (STFT) [20] može se primijeniti za procjenu frekvencijskog sadržaja uzastopnih prozora podataka. Međutim, važno je napomenuti da ovakav prozorski pristup može donijeti određene probleme vezane uz rezoluciju. S druge strane, valićne transformacije (CWT/DWT) [21, 22] pružaju bolju alternativu. Ove transformacije ponavljaju korelaciju "osnovnog" valovitog oblika sa signalom, privremeno ga komprimirajući ili šireći prema potrebi, čime omogućavaju pregled naglih promjena u ubrzanju. Međutim, važno je napomenuti da su ove transformacije poznate po većoj računalnoj zahtjevnosti. [6]

Korelacijska analiza

Korelacijska analiza broji korake na način da računa i uspoređuje koeficijente korelacije između dva susjedna prozora podataka akcelerometra [6, 23, 24]. U ovim metodama, ubrzanje se transformira iz vremenske u frekvencijsku domenu korištenjem diskretne Fourierove transformacije, dinamičkog vremenskog iskrivljavanja ili autokorelacije. No, ovi pristupi zahtijevaju značajno računalno opterećenje i nisu prikladni za aplikacije na pametnim telefonima. [25]

3.2.2 Metode strojnog učenja

Mnoga istraživanja koriste tehnike strojnog učenja kako bi klasificirali aktivnosti temeljem karakteristika izvučenih iz signala akcelerometra. Međutim, dosad nije pronađena pojedinačna karakteristika niti metoda učenja koja bi se istaknula kao najbolja.

Uobičajene karakteristike koje se koriste u ovim istraživanjima uključuju srednju vrijednost, varijancu ili standardnu devijaciju, energiju, entropiju, korelaciju između različitih osi te koeficijente brze Fourierove transformacije (FFT). [6]

Većina istraživanja koristi ove karakteristike za obuku različitih klasifikatora kao što su neuronske mreže [26], metode potpornih vektora (SVM) [27] ili modeli Gaussovih mješavina (GMM) [28], ili ih koristi kao ulaz za neparametarske metode

klasifikacije poput k-najbližih susjeda [29]. Ove tehnike mogu biti učinkovite, posebno one koje se temelje na Bayesovoj statistici, ali zahtijevaju nadziranu obuku, često za svaku osobu i različite načine nošenja uređaja.

3.2.3 Metode dubokog učenja

Duboko učenje je postalo ključno u analizi koraka i hodanja, zbog svoje sposobnosti učenja iz podataka bez potrebe za ručnim definiranjem pravila. Neki istraživači su koristili modele dubokog učenja za precizno detektiranje koraka i procjenu duljine koraka.

Jedan pristup koristi Bidirekcijsku rekurentnu neuronsku mrežu (BLSTM-RNN) za analizu sirovih senzorskih podataka s pametnih telefona [30]. Ovaj pristup postigao je visoku točnost u detekciji koraka, ali nije u mogućnosti obraditi lažno pozitivne detekcije koraka i različite aktivnosti hodanja poput penjanja po stepenicama.

Drugi pristup koristi senzore u ulošku i konvolucijske neuronske mreže (CNN) za detekciju koraka i usporedbu koraka zdravih i pacijenata s Parkinsonovom bolešću [31]. Ovaj model postigao je visoku preciznost u detekciji koraka za svakodnevne aktivnosti i za pacijente s Parkinsonovom bolešću.

Ova dva pristupa ilustriraju kako duboko učenje može biti korisno za analizu koraka i hodanja, pružajući visoku preciznost i mogućnost prilagodbe različitim situacijama i skupinama pacijenata.

Poglavlje 4

Android aplikacija za brojanje koraka

Nakon pažljivog razmatranja različitih metoda za detekciju koraka, odabrana je primjena metode detekcije vrhova. Ova metoda ističe se zbog relativno niske složenosti u usporedbi s drugim tehnikama, niskih računalnih zahtjeva te zadovoljavajućih rezultata. Prethodno istraživanje [16] demonstriralo je visoku učinkovitost metode, s prosječnom točnošću od 95% i standardnom devijacijom od 4,5%. Važno je napomenuti da to istraživanje nije uzelo u obzir situacije "lažnih koraka", koje se javljaju kada korisnik uređaja obavlja druge svakodnevne aktivnosti, poput pisanja poruka, razgovora ili igranja igrica, dok uređaj bilježi pokrete.

U cilju poboljšanja preciznosti detekcije koraka u stvarnom svijetu, kao inspiracija poslužio je rad [25], u kojem su predstavljena svojstva za identifikaciju "lažnih koraka" i optimizaciju metode detekcije vrhova. Odlučeno je implementirati ova dodatna svojstva zajedno s osnovnom metodom detekcije vrhova kako bi se dobio robustan sustav za brojanje koraka.

4.1 Korištene tehnologije

U procesu razvoja Android aplikacije za brojanje koraka, koristilo se nekoliko ključnih tehnologija koje su imale značajnu ulogu u stvaranju i funkcionalnosti aplika-

cije. U nastavku će biti detaljnije opisane te tehnologije kako bi se bolje razumjele osnovne komponente aplikacije i njezino funkcioniranje.

Android Studio je službeno integrirano razvojno okruženje (IDE) razvijeno od strane Google-a, posebno prilagođeno za kreiranje Android aplikacija. Kroz Android Studio, programeri mogu koristiti kako Java, tako i Kotlin jezik za razvoj aplikacija te iskoristiti intuitivno korisničko sučelje za dizajniranje i upravljanje projektima. Osim toga, nudi širok spektar alata, uključujući one za analizu koda, testiranje na emulatorima i stvarnim uređajima, te podršku za verzioniranje i izgradnju aplikacija.

Kotlin je programski jezik koji se sve više koristi za razvoj Android aplikacija. On pruža veću sigurnost i izražajnost u odnosu na Java jezik, čineći ga iznimno prikladnim za razvoj aplikacija za brojanje koraka. Kotlin također omogućava programerima pisanje čitljivog i sigurnog koda.

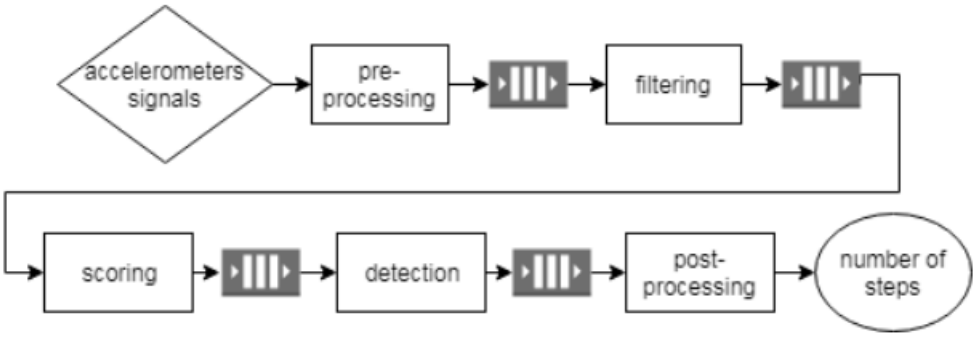
Akcelerometar je hardverski senzor prisutan na većini pametnih telefona. On mjeri ubrzanje uređaja u tri dimenzije (x, y, z) i ključan je za praćenje korisnikovih pokreta. U našoj aplikaciji, akcelerometar ima ključnu ulogu jer omogućava detekciju i praćenje koraka. Putem akcelerometra, aplikacija bilježi promjene brzine i smjera uređaja te na temelju tih podataka detektira korake korisnika.

Sve te tehnologije zajedno čine osnovu za razvoj aplikacije za brojanje koraka. U idućim sekcijama, istražit će se način na koji su ove komponente integrirane i na koji način zajedno omogućavaju praćenje koraka i aktivnosti korisnika.

4.2 Struktura

Ova aplikacija se sastoji od osam ključnih klasa, svaka s precizno određenim zadatkom i svrhom. Proces počinje u `MainActivity` klasi, gdje se prikupljaju podaci s akcelerometra i prosljeđuju `StepCounter` klasi. Ovdje se inicijalizira objekt `DataPoint` koji sadrži sve bitne informacije o trenutnom ubrzanju. Nakon toga, pozivaju se i aktiviraju ostalih pet klasa, svaka odgovorna za specifične korake u obradi podataka akcelerometra. Redoslijed tih klasa je: `PreProcessStage`, `FilterStage`, `ScoringStage`, `DetectionStage`, `PostProcessStage`, respektivno.

Kako bi se omogućila brzu obrada podataka, svaki blok za obradu podataka i signala ubrzanja može biti izrađen u vlastitoj paralelnoj niti. Kako bi se privremeno pohranili podaci između svakog koraka u procesu, koriste se međuspremnici. Slika 4.1 grafički prikazuje redoslijed pozivanja blokova i samog algoritma.



Slika 4.1 Dijagram blokova algoritma za brojanje koraka. Preuzeto s [16]

U nastavku će se detaljnije istražiti svaka od ovih ključnih klasa i njihova specifična uloga u ostvarivanju precizne i pouzdane detekcije i brojanja koraka.

4.2.1 `DataPoint`

Klasa `DataPoint` predstavlja pojedini podatak ili uzorak koji se koristi za obradu u kontekstu brojanja koraka. Ova klasa sadrži nekoliko atributa i metoda za definiranje i dohvaćanje tih atributa.

Atributi:

- `time`: Predstavlja vremenski trenutak kada je uzorak akcelerometra zabilježen. Ovaj atribut se koristi za praćenje vremenskog redoslijeda uzoraka.
- `magnitude`: Predstavlja magnitudu ubrzanja u trenutnom uzorku. Magnituda se koristi za analizu promjena u ubrzanju, što je važno za detekciju koraka.
- `oldMagnitude`: Ovaj atribut predstavlja prethodnu magnitudu ubrzanja. Pohranjuje se kako bi se sačuvala originalna magnituda nakon što se vrši scoring gdje se mijenja vrijednost magnitude.

- `eos`: eos je skraćenica od "End of Sequence" (kraj sekvence) i predstavlja zastavicu koja označava kraj niza podataka. Ova zastavica se postavlja kada se sekvenca podataka završava, na primjer, na kraju hodanja.

4.2.2 `MainActivity`

Klasa `MainActivity` ima ključnu ulogu u aplikaciji, budući da je središnja točka upravljanja senzorima, prikaza rezultata i interakcije s korisnikom. U nastavku su detaljnije razmotrene svaka od njezinih komponenata i funkcionalnosti:

Inicijalizacija

U prvim recima klase `MainActivity`, započinje inicijalizacija potrebnih komponenti. Prvo, uvoze se relevantne biblioteke i klase poput `Sensor`, `SensorEventListener`, `SensorManager` i drugih. Zatim se definiraju različite varijable i konstante koje će biti ključne tijekom izvođenja aplikacije. To uključuje:

- `sensorManager`: Instanca `SensorManager` klase koja će se koristiti za upravljanje senzorima uređaja
- `accelerometer`: Predstavlja akcelerometar uređaja, a dobiva se iz `sensorManager`
- `stepCounter`: Instanca klase `StepCounter`, koja je odgovorna za brojanje koraka i obradu senzorskih podataka
- Konstanta `SAMPLING_FREQUENCY`: Ova konstanta definira frekvenciju uzorkovanja senzora (u Hertzima) i ima ključnu ulogu u preciznosti brojanja koraka
- Razne varijable za praćenje broja koraka, trenutnog statusa aplikacije i komponenti korisničkog sučelja, kao što su `tv_stepCount` (tekstualno polje za prikaz broja koraka) i `btn_toggleStepCounter` (gumb za pokretanje/zaustavljanje brojanja koraka)

Metoda `onCreate`

Ova metoda se izvršava prilikom pokretanja aktivnosti (`MainActivity`). U njoj se provode sljedeće ključne radnje:

- Provjera dozvola: Prilikom izvođenja Android aplikacije koja koristi određene senzore, kao što je u ovom slučaju akcelerometar, potrebno je osigurati da aplikacija ima potrebne dozvole za pristup tim senzorima. U suprotnom, aplikacija neće moći koristiti senzore i neće moći funkcionirati kako je predviđeno. U dijelu koda koji se odnosi na dozvole, koristi se Android funkcionalnost za provjeru dozvola. Provjerava se dozvola `ACTIVITY_RECOGNITION` koja je potrebna za prepoznavanje aktivnosti korisnika, uključujući korake. Ako ta dozvola nije odobrena, korisniku se šalje zahtjev za odobrenje te dozvole pomoću `requestPermissions` metode
- Inicijalizacija grafičkih elemenata: Postavlja se sučelje aplikacije inicijalizacijom komponenata kao što su `tv_stepCount` i `btn_toggleStepCounter`. To uključuje i postavljanje prvotnih vrijednosti i teksta na sučelju.
- Priprema `StepCounter`-a: Inicijalizira se instanca klase `StepCounter`, postavljaju se metode za ažuriranje broja koraka i završetak obrade podataka te se postavljaju slušatelji događaja
- Inicijalizacija senzora: Pomoću `sensorManager`-a dobiva se pristup senzoru akcelerometra kako bi se omogućila obradu senzorskih podataka

Metoda `startClickListener`

Ova metoda obrađuje događaje klika na gumbu `btn_toggleStepCounter`. Ovisno o trenutnom stanju aplikacije, ova funkcija omogućuje korisniku pokretanje ili zaustavljanje brojanja koraka. Ako je brojanje koraka već u tijeku (tj. korisnik želi zaustaviti brojanje), senzor se isključuje i brojanje prestaje. Ako korisnik želi započeti brojanje koraka, omogućuje se brojanje i uključuje se senzor za prikupljanje senzorskih podataka.

Metoda `accelerometerEventListener`

`accelerometerEventListener` je implementacija sučelja `SensorEventListener` koja obrađuje promjene u senzoru akcelerometra. Ova funkcija reagira na svaku promjenu u podacima koje šalje akcelerometar uređaja, a te promjene odnose se na ubrzanje koje uređaj mjeri u određenom trenutku. U njoj se pozivaju dvije osnovne metode:

- `onSensorChanged(event: SensorEvent)`: Ova metoda se poziva svaki put kada senzor akcelerometra zabilježi promjenu u podacima. Parametar `event` sadrži informacije o toj promjeni, uključujući vrijednosti ubrzanja po osima (x, y i z). Ubrzanje se mjeri u metrima po sekundi kvadratnoj (m/s^2) i ovisi o brzini i smjeru kretanja uređaja.
- `stepCounter.processSample(event.timestamp, event.values)`: Ovdje se poziva metoda `processSample` objekta `stepCounter`, koji je instanca klase `StepCounter`. Ova metoda prima dva argumenta: `event.timestamp` koji predstavlja vremenski žig trenutka kada je uzorak akcelerometra zabilježen te `event.values` koji sadrži tri vrijednosti ubrzanja (po x, y i z osima) za taj trenutak. Ova funkcija osigurava da se svaki uzorak sa senzora akcelerometra obradi i proslijedi `StepCounter` klasi na daljnje procesiranje. Ovdje se vrši osnovna obrada podataka iz senzora prije nego što se proslijede različitim fazama za konačno brojanje koraka. Osim toga, korištenje ovog slušatelja omogućava da se obrada pokreće svaki put kad senzor zabilježi promjenu u ubrzanju, što je ključno za praćenje koraka tijekom kretanja. Na taj način, `accelerometerEventListener` ima ključnu ulogu u kontinuiranom praćenju ubrzanja uređaja i osigurava da svi relevantni podaci šalju `StepCounter` klasi za daljnju obradu i brojanje koraka.

Sveukupno, `MainActivity` je odgovorna za upravljanje senzorima, prikaza broja koraka na korisničkom sučelju i omogućava korisnicima jednostavno pokretanje i zaustavljanje brojanja koraka putem gumba. Ova klasa također usko surađuje sa `StepCounter` klasom koja obavlja kompleksniju obradu senzorskih podataka za precizno brojanje koraka.

4.2.3 `StepCounter`

Klasa `StepCounter` predstavlja srce Android aplikacije za brojanje koraka. Ova klasa je odgovorna za prikupljanje podataka sa senzora akcelerometra, obradu tih podataka i brojanje koraka na temelju njih. Slijedi detaljno objašnjenje svake komponente klase:

Sučelja

Klasa StepCounter koristi dva sučelja:

- `OnStepUpdateListener`: Služi za obavještavanje aplikacije kada se otkrije novi korak.
- `OnFinishedProcessingListener`: Služi za obavještavanje aplikacije kad završe sve faze obrade podataka.

Varijable i nizovi

- `steps`: Varijabla koja čuva broj otkrivenih koraka i ažurira se kako se otkrivaju novi koraci.
- Različiti nizovi poput `rawData`, `ppData`, `smoothData`, `peakScoreData`, `peakData` i `stepsData` koriste se za pohranu podataka tijekom različitih faza obrade podataka. Ovi nizovi su thread-safe, što omogućava siguran pristup iz različitih niti.

Metode

- Metoda `setUp`: Ova metoda poziva se prije početka brojanja koraka kako bi postavila inicijalno stanje brojača i pripremila nizove za pohranu podataka.
- Metoda `start`: Ova metoda pokreće algoritam za brojanje koraka. Resetira stanje i pokreće različite faze obrade podataka u zasebnim nitima (npr. `PreProcessStage`, `FilterStage`, ...). Različite faze služe za obradu podataka i prepoznavanje koraka, osiguravajući točnost i brzinu brojanja koraka.

- Metoda `stop`: Ova metoda zaustavlja brojanje koraka. Signalizira kraj prikupljanja podataka dodavanjem posebnog podatka ("end of stream") u listu podataka. To označava kraj obrade.
- Metode za dodavanje slušatelja: `addOnStepUpdateListener` omogućava dodavanje slušatelja koji prate ažuriranje broja koraka.
  `setOnFinishedProcessingListener` postavlja slušatelja koji će se pozvati nakon završetka obrade podataka.
- Metoda `incSteps`: Ova metoda povećava broj koraka i obavještava sve registrirane slušatelje o ažuriranju broja koraka. To omogućava ažuriranje prikaza broja koraka na sučelju aplikacije.
- Metoda `processSample`: Ova metoda omogućava dodavanje novog uzorka sa senzora akcelerometra u algoritam za obradu podataka. Računa magnitudu uzorka, odnosno ukupno ubrzanje u tri smjera, s obzirom da fizička orijentacija uređaja nije poznata.Računa se po formuli: mag = $\sqrt{x^2 + y^2 + z^2}$

Na slici 4.2 prikazan je graf neobrađenog signala prikupljenog sa senzora i bez obrade.

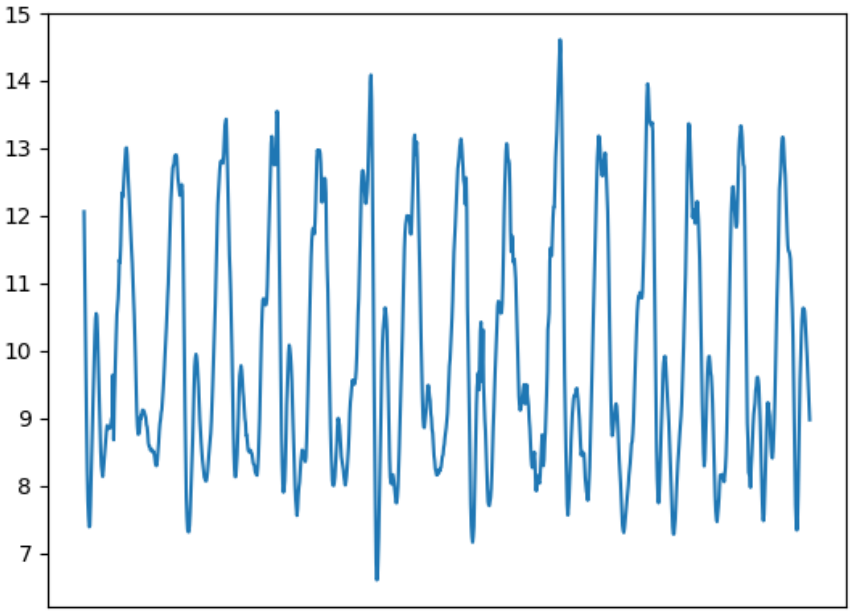
4.2.4 `PreProcessStage`

Klasa `PreProcessStage` obavlja pred-procesiranje niza podataka ubrzanja koristeći linearnu interpolaciju. U nastavku će biti objašnjen koncept linearne interpolacije te njena implementacija u algoritmu.

Linearna interpolacija

Linearna interpolacija je jedna od osnovnih metoda interpolacije u matematici i računarstvu. Koristi se za aproksimaciju vrijednosti između dvije poznate točke na temelju linearnog modela, što znači da se pretpostavlja da postoji linearna veza između tih dviju točaka.

U ovom algoritmu, linearna interpolacija je ključna za glatko povezivanje podataka između uzastopnih točaka, što je bitno za praćenje koraka. Koristeći očita-



Slika 4.2 Neobrađeni signal akcelerometra

nja senzora u diskretnim vremenskim trenucima, linearna interpolacija omogućuje uzorkovanje podataka u određenom periodu. Time se postiže stvaranje neprekidne krivulje koja precizno odražava promjene u podacima tijekom vremena, osiguravajući dosljednost u analizi i preciznost u detekciji koraka.

Matematička formula za linearnu interpolaciju između dvije točke (x1, y1) i (x2, y2) te odabrane vremenske točke x između x1 i x2 je:

$$y = y_1 + \frac{(x - x_1) \cdot (y_2 - y_1)}{x_2 - x_1}$$

Gdje su:

- x - ciljana vremenska točka za interpolaciju
- x_1 i x_2 - vremenske točke dviju poznatih podataka

- y_1 i y_2 - vrijednosti dviju poznatih točaka
- y - interpolirana vrijednost za vremensku točku x

Linearna interpolacija omogućuje glatko povezivanje podataka i aproksimaciju vrijednosti između dviju poznatih točaka, što je korisno za analizu i praćenje promjena u podacima tijekom vremena.

Varijable i inicijalizacija

Ova klasa prihvaća dva glavna argumenta: `inputQueue` s ulaznim podacima i `outputQueue` koji će sadržavati interpolirane podatke. Također, koristi sljedeće varijable: lista `window` koja pohranjuje podatke za interpolaciju, `dp` kao trenutni podatak koji se obrađuje, `interpolationTime` koji služi kao vremenski interval za interpolaciju definiran na 10 milisekundi, `timeScalingFactor` - faktor skaliranja za vremenske oznake, `interpolationCount` kao brojač interpoliranih podataka i `startTime` koja označava vrijeme početka za skaliranje.

Interpolacija

Interpolacija se izvodi na sljedeći način:

- Provjerava se ima li novih podataka u `inputQueue`. Ako takvi podaci postoje, uzima se prvi podatak iz liste.
- Vrijeme tog podatka skalira se pomoću funkcije `scaleTime()`, pri čemu se vrijednosti izražene u nano sekundama pretvaraju u milisekunde.
- Provjerava se postoji li u prozoru (`window`) najmanje dvije točke. Za provedbu interpolacije potrebne su najmanje dvije točke kako bi se mogle izračunati nove vrijednosti između njih.
- Zatim se uzimaju vremenske oznake (`time1` i `time2`) prve dvije točke u prozoru.
- Proračun broja potencijalnih interpoliranih točaka između dvije vremenske točke izvodi se tako da se razlika između `time2` i `time1` podijeli s vremenskim intervalom za interpolaciju (`interpolationTime`), a rezultat se zaokruži na

najbliži viši cijeli broj korištenjem funkcije `ceil`. Ovaj postupak omogućava precizno određivanje koliko točaka će biti interpolirano između dvije poznate točke, osiguravajući glatko povezivanje podataka. Matematički, ovaj postupak može se izraziti kao:

$$\text{numberOfPoints} = \lceil \frac{\text{time2} - \text{time1}}{\text{interpolationTime}} \rceil$$

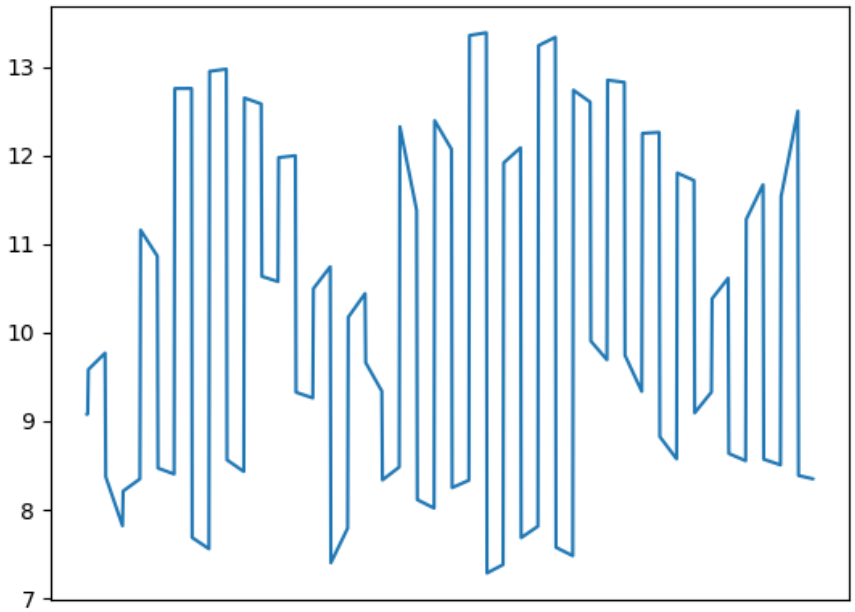
- Nakon što je određeno koliko potencijalnih interpoliranih točaka treba izračunati, slijedi petlja koja obrađuje svaku od tih točaka
- U svakoj iteraciji ove petlje izračunava se vremenska oznaka interpolirane točke (`interpTime`) množeći brojač interpoliranih podataka (`interpolationCount`) s vremenskim intervalom između točaka koje se interpoliraju. Ovime se dobiva precizno vrijeme za trenutnu interpoliranu točku.
- Zatim se provjerava nalazi li se vremenska oznaka interpolirane točke (`interpTime`) između vremenskih oznaka dviju poznatih točaka (`time1` i `time2`). Ako se nalazi, računa se interpolacija.
- Koristi se funkcija `linearInterpolate()` kako bi se izračunale vrijednosti za interpoliranu točku između početne točke i završne točke za odabrano vrijeme (`interpTime`). Ova funkcija primjenjuje linearnu interpolaciju kako bi izračunala vrijednosti na osnovu vremena između dvije poznate točke.
- Nakon izračuna vrijednosti za interpoliranu točku, dodaje se u `outputQueue` kako bi se dalje koristila u obradi podataka.

Na slici 4.3 prikazan je graf signala nakon primjene koraka interpolacije.

Na ovaj način osigurava se uzorkovanje podataka na intervalu od 10 ms, osiguravajući kontinuitet i glatko povezivanje između dviju poznatih točaka. Ovaj postupak garantira dosljednost i kontinuitet u analizi podataka.

4.2.5 `FilterStage`

Klasa `FilterStage` izvršava se nakon klase `PreProcessStage` te prima interpolirane podatke koje zatim filtrira kako bi se smanjio utjecaj šuma na podatke. Prije



Slika 4.3 Signal nakon faze pred-procesiranja

samog objašnjenja koda, važno je objasniti niskopropusni FIR filtar i Gaussovu krivulju koji su korišteni u fazi filtriranja signala.

Niskopropusni FIR (Finite Impulse Response) filtar

Niskopropusni FIR filtar je vrsta digitalnog filtra koji se koristi za uklanjanje visokih frekvencija iz signala, čime se postiže glađenje i uklanjanje šuma. Naziv "niskopropusni" sugerira da ovaj filtar dopušta prolazak niskofrekventnih komponenata signala dok blokira visokofrekventne komponente. Ovaj tip filtra ima končani impulsni odgovor, što znači da ima konačan broj koeficijenata za filtriranje signala.

FIR filtri se često koriste u raznim primjenama obrade signala, uključujući podatke akcelerometra. U kontekstu brojanja koraka, niskopropusni FIR filtar

ima važnu ulogu u uklanjanju visokofrekventnih šumova i nepotrebnih oscilacija iz signala, čime se omogućuje preciznija detekcija koraka.

Glavna svrha niskopropusnog FIR filtra u obradi podataka akcelerometra je sljedeća:

- Uklanjanje šuma: Akcelerometri su osjetljivi na različite izvore šuma, kao što su mehanički, električni i termalni šumovi. Niskopropusni filtar uklanja visokofrekventne komponente ovog šuma, čime se poboljšava kvaliteta podataka.
- Glađenje signala: Filtrirani signal postaje glađi i manje oscilira. To je važno za precizno brojanje koraka jer su oscilacije i šumovi u podacima nepoželjni.
- Poboljšanje detekcije koraka: Uklanjanjem visokofrekventnih komponenti koje ne odražavaju stvarne korake, filtar omogućava algoritmima za brojanje koraka da preciznije detektiraju pokrete i korake osobe.

Gaussova krivulja

Gaussova krivulja, također poznata kao normalna krivulja, je matematički oblik krivulje koji se često koristi za modeliranje različitih prirodnih i društvenih fenomena. Gaussova krivulja ima simetričan oblik oko svoje središnje vrijednosti i karakterizira ju glatko opadanje prema rubovima. Ova krivulja je ključna u mnogim područjima, uključujući statistiku, inženjerstvo i znanstvena istraživanja.

Matematički izraz za Gaussovu krivulju definira se sljedećom funkcijom gustoće vjerojatnosti (PDF - Probability Density Function):

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

- $f(x)$ predstavlja vrijednost Gaussove funkcije za dani ulaz x.
- σ je standardna devijacija koja kontrolira širinu krivulje.
- μ je srednja vrijednost (ili očekivanje) i predstavlja vrhunac funkcije.
- π je matematička konstanta Pi (približno 3.14159).

- e je matematička konstanta Eulerova broja (približno 2.71828).

Ovaj izraz opisuje oblik Gaussove krivulje, koja je simetrična oko srednje vrijednosti μ i ima širinu koja je proporcionalna standardnoj devijaciji σ. Gaussova funkcija često se koristi za modeliranje distribucija u statistici i kao osnova za dizajniranje niskopropusnih filtra te je česta u mnogim matematičkim i inženjerskim primjenama.

Gaussova krivulja ključna je za generiranje koeficijenata FIR filtra, s ključnom ulogom u poboljšanju učinkovitosti detekcije koraka. Pažljivo oblikovani koeficijenti ne samo da čine filtriranje signala učinkovitim već i čuvaju bitne informacije signala, smanjujući istovremeno smetnje od šuma i nepotrebnih oscilacija. Uz to, zahvaljujući Gaussovoj krivulji, postiže se elegantno zaglađivanje signala, eliminirajući visokofrekventne šumove i oscilacije.

Varijable i Inicijalizacija

Klasa prima dvije glavne liste, `inputQueue` i `outputQueue`, koje sadrže podatke prije i nakon filtriranja. Također koristi nekoliko drugih varijabli, uključujući `window` za čuvanje podataka u prozoru i `filterCoefficients` za koeficijente filtra.

Prilikom inicijalizacije, klasa izračunava sumu koeficijenata filtra (`filter_sum`). Koeficijenti filtra generiraju se koristeći funkciju `generateCoefficients()`. Ovi koeficijenti su bitni za proces filtriranja i pravilno definiraju karakteristike niskopropusnog filtra.

Korištenje funkcije za generiranje koeficijenata filtra

U statičkom bloku `companion object`, nalazi se metoda `generateCoefficients()` koja generira koeficijente filtra koristeći matematičku formulu za Gaussovu funkciju.

- Definira se konstanta `FILTER_LENGTH`, koja predstavlja duljinu filtra. Duljina filtra određuje koliko će se uzoraka prošlih ulaznih podataka koristiti. Ako je duljina filtra mala, filtar će imati manji utjecaj na prošle uzorke, što

može rezultirati brzim odzivom na promjene u ulaznom signalu. Međutim, kratki filtri mogu biti osjetljivi na šum i neželjene oscilacije. Dulji filtri mogu pružiti bolje izglađivanje signala i smanjiti utjecaj šuma. Međutim, to također može dovesti do duljih zakašnjenja u odzivu na promjene u signalu. U ovom slučaju, duljina filtra postavljena je na 13 s obzirom da se ta vrijednost pokazala optimalnom u radu [16].

- Definira se konstanta `FILTER_STD` = 0.35, koja predstavlja standardnu devijaciju filtra. Vrijednost je također uzeta iz rada [16] kao optimalna. Standardna devijacija Gaussove funkcije kontrolira širinu zvonaste krivulje filtra. Veća standardna devijacija rezultirat će širom i spljoštenom krivuljom, dok će manja standardna devijacija rezultirati uskom i oštrijom krivuljom. Smanjivanje standardne devijacije može pojačati odziv filtra na određene frekvencije, ali također može povećati osjetljivost na šum. Povećanje standardne devijacije čini krivulju širom i manje oštrom. To može rezultirati boljim odbacivanjem šuma, ali također može smanjiti preciznost u filtriranju specifičnih frekvencija.
- Uz pomoć matematičke formule za Gaussovu funkciju, generiraju se koeficijenti te dodaju u listu `coeff`.

Filtriranje podataka

Unutar metode `run()`, klasa aktivno obrađuje podatke. Ako postoji niz podataka u `inputQueue`, uzima prvi element iz reda. Također se provjerava poseban uvjet za kraj podataka (end of stream - eos). Ako je detektiran kraj, klasa završava s filtriranjem i šalje posljednji podatak u `outputQueue`.

Proces filtriranja

Kako bi se primijenio niskopropusni FIR filtar, koristi se prozor određene duljine (13 u ovom slučaju) i primjenjuju se koeficijenti filtra na svaku vrijednost u tom prozoru. Korišten je FIR filtar, što znači da se svaka izlazna vrijednost računa kao težinska suma ulaznih vrijednosti i odgovarajućih koeficijenata filtra. U ovom slu-

čaju, duljina prozora i koeficijenti filtra odabrani su tako da stvore niskopropusni filtar s određenom karakteristikom rezanja (cut-off frequency) kako bi se smanjio utjecaj visokofrekvencijskih komponenti koje nisu relevantne za korake i ljudski pokret.

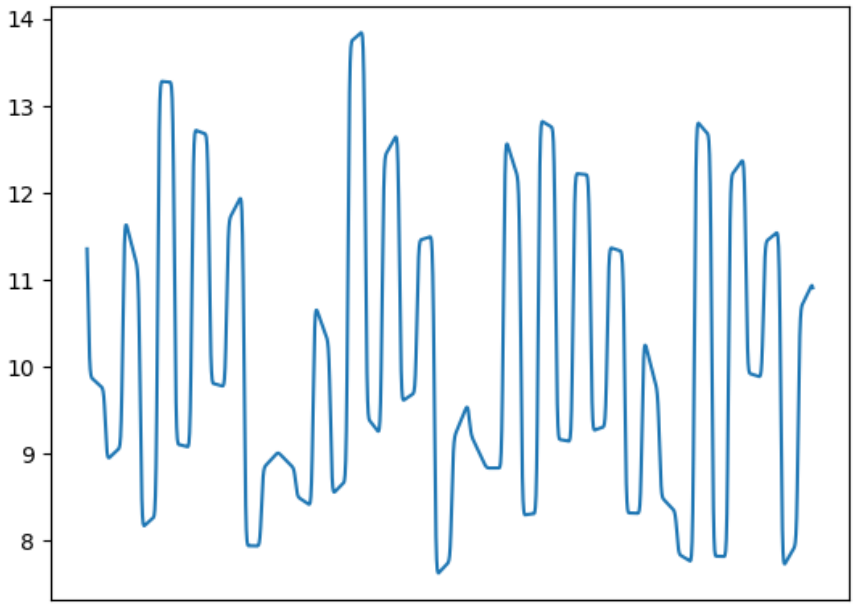
- Prvo se provjerava ima li prozor `window` dovoljan broj podataka za primjenu filtra. `FILTER_LENGTH` određuje se duljinu prozora, a kada prozor sadrži `FILTER_LENGTH` elemenata, znači da se može primijeniti filtar.
- Inicijalizira se varijablu `sum` na 0, koja će se koristiti za akumulaciju vrijednosti izračunate primjenom filtra.
- Prolazi se kroz sve koeficijente filtra. Za svaki podatak u prozoru (`window[i]`), množi se njegova magnituda s odgovarajućim koeficijentom filtra (`filterCoefficients[i]`) i dodaje se ta vrijednost na sumu. Ovaj korak predstavlja primjenu filtra na ulazne podatke.
- Nakon što je zbrojen doprinos svih elemenata, stvara se novi `DataPoint` objekt koji će predstavljati izlaz iz filtriranja. Srednja točka u prozoru uzima se kao vrijeme novog podatka, a izlazna vrijednost je suma podataka podijeljena s faktorom normalizacije `filter_sum`.
- Dodaje se novi podatak u `outputQueue`, koji će se koristiti za daljnju obradu.
- Kako bi se održala konstantna duljina prozora, uklanja se najstariji podatak iz prozora tako da bude spreman za sljedeći korak filtriranja.

Na slici 4.4 prikazan je graf signala nakon primjene koraka filtriranja.

4.2.6 `ScoringStage`

Klasa `ScoringStage` ima zadatak izračunati ocjenu koraka temeljenu na ubrzanju koraka i dodati te ocjene u izlazni red. U nastavku je predstavljen pregled i redoslijed odvijanja aktivnosti:

- Ulazni i izlazni podaci pohranjuju se u listama `inputQueue` i `outputQueue`. Ulazni podaci sadrže filtrirane vrijednosti ubrzanja, dok se izlaz koristi za pohranu izračunatih ocjena.



Slika 4.4 Signal nakon faze filtriranja

- `run` metoda obavlja glavnu obradu podataka, provjerava nove podatke u `inputQueue` i dodaje ih u trenutni prozor. Ako je trenutni podatak označen kao kraj niza, dodaje ga u izlaz, inače nastavlja obradu.
- Kad trenutni prozor (`window`) sadrži dovoljno podataka (veličina prozora definirana je kao `WINDOW_SIZE = 35`), poziva se metoda `scorePeak` za izračun ocjene koja će nastavno biti detaljnije objašnjena.
- Rezultirajuća ocjena dodaje se u izlaznu listu kao novi `DataPoint`. Središnja točka trenutnog prozora koristi se kao referenca za vrijeme tog podatka.

Metoda `scorePeak`

Metoda `scorePeak` koristi se za procjenu izraženosti vrhova u danom uzorku, što pomaže u identifikaciji koraka ili pokreta stopala u signalu. Analizira razlike u

magnitudi između središnjeg vrha i okolnih podataka kako bi ocijenila vrhove i olakšala detekciju koraka.

Koristi se metoda srednje razlike za identifikaciju izraženih vrhova u signalu koji se obično pojavljuju tijekom svakog koraka. Ova metoda razmatra razliku između središnjeg vrha i okolnih podataka kako bi odredila koliko su ti vrhovi izraženi u odnosu na svoje okoline. Računa se po formuli:

$$P_i = \frac{1}{2N} \sum_{k=-N, k \neq i}^{N} (x_i - x_{i+k})$$

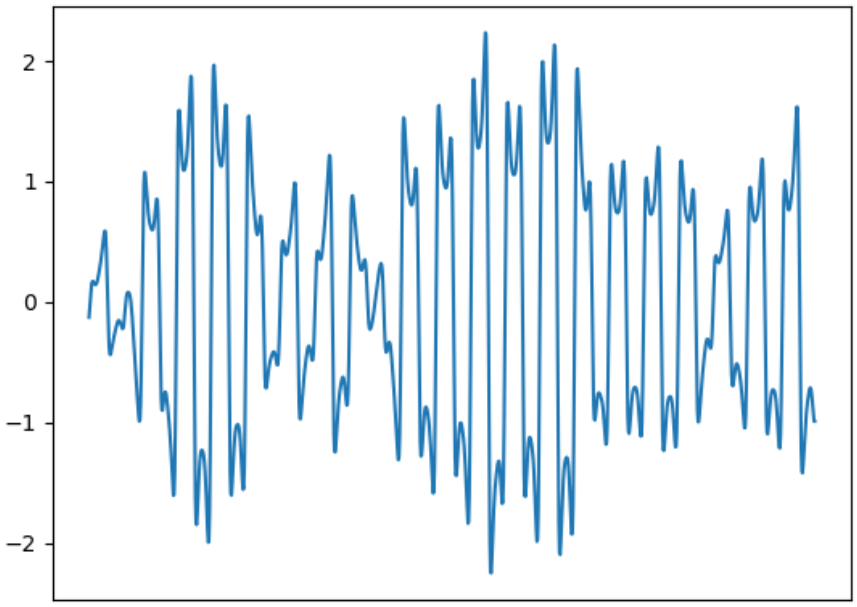
Gdje su:

- P_i predstavlja izračunanu vrijednost srednje razlike za i-ti podatak
- N označava polovicu širine prozora
- k predstavlja indeks susjednih podataka u prozoru
- x_i predstavlja vrijednost središnjeg podatka u prozoru
- x_{i+k} označava vrijednost susjednih podataka u prozoru

Redoslijed odvijanja metode `scorePeak` odvija se na sljedeći način:

- Prvo, metoda određuje sredinu (točku `midpoint`) danog niza podataka `data`, koja predstavlja vrh (peak) u signalu. Ova točka nalazi se na polovici niza.
- Zatim, metoda izračunava razlike u magnitudi (intenzitetu) između središnjeg vrha i svih podataka s lijeva i zdesna u odnosu na središnju točku (`midpoint`).
- Varijabla `diffLeft` predstavlja kumulativnu razliku u magnitudi između središnjeg vrha i svih podataka na lijevoj strani središnje točke.
- Varijabla `diffRight` predstavlja kumulativnu razliku u magnitudi između središnjeg vrha i svih podataka na desnoj strani središnje točke.
- Konačno, metoda vraća prosječnu razliku između lijevih i desnih magnituda vrhova, podijeljenu s `WINDOW_SIZE - 1`.

Na slici 4.5 prikazan je graf signala nakon primjene koraka ocjenjivanja.



Slika 4.5 Signal nakon faze ocjenjivanja

4.2.7 `DetectionStage`

Ova faza identificira potencijalne vrhove koji bi se mogli povezati s korakom putem statističke detekcije iznimnih vrijednosti. Tijekom obrade signala, algoritam prati pomičnu srednju vrijednost i standardnu devijaciju. Ove dvije veličine koriste se kako bi se odredilo je li dani uzorak iznimno velik. Ako je razlika između uzorka i srednje vrijednosti veća od c puta standardne devijacije, označava se kao potencijalni korak.

Klasa `DetectionStage` prima izlazne vrijednosti `ScoringStage` klase te se može podijeliti u nekoliko koraka:

- Inicijalizacija varijabli: Varijabla `dp` za trenutni podatak, `count` za praćenje broja prikupljenih podataka, `mean` za srednju vrijednost i `std` za standardnu devijaciju. Također, postavlja se prag (`threshold = 1.2`) koji će se koristiti

za otkrivanje potencijalnih vrhova.

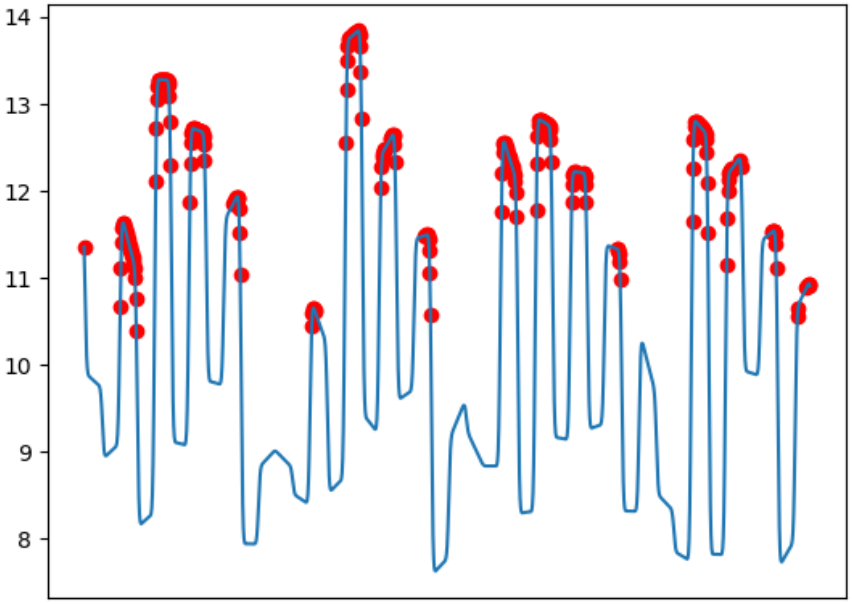
- Prikupljanje podataka i ažuriranje srednje vrijednosti i standardne devijacije: Algoritam neprestano prikuplja podatke iz ulazne liste `inputQueue`. Za svaki novi podatak ažurira se brojač (`count`) i računa srednja vrijednost (`mean`) i standardna devijacija (`std`).
- Računanje potencijalnih vrhova: Kada se prikupi dovoljno podataka (u ovom slučaju, više od 15), svaki novi podatak uspoređuje se s trenutnom srednjom vrijednošću i standardnom devijacijom. Ako razlika između novog podatka i srednje vrijednosti premaši prag (`threshold`) pomnožen sa standardnom devijacijom (`std`), označava se kao potencijalni korak. To znači da su vrhovi u signalu koji premašuju prag označeni kao koraci ili pokreti stopala. Ova faza je ključna za otkrivanje koraka u analizi signala.

Kroz ove korake, `DetectionStage` identificira potencijalne kandidate za korake putem statističkog otkrivanja odstupanja od uobičajenog obrasca u signalu. Ova faza značajno pomaže u razdvajanju stvarnih koraka od ostalih promjena u signalu.

Na slici 4.6 prikazan je graf na kojem crvene točke označavaju potencijalne vrhove nakon primjene koraka detekcije.

4.2.8 `PostProcessStage`

U ovoj fazi postprocesiranja, prozor fiksne veličine klizi preko potencijalnih vrhova u signalu i pažljivo bira samo najizraženiji vrh unutar tog prozora. Svi potencijalni vrhovi koncentrirani su u blizini glavnog vrha. Ova faza identificira lokalni maksimum među tim vrhovima i odabire ga. Prilagodbom veličine prozora moguće je kontrolirati broj koraka koji se bilježe unutar tog prozora, čime se sprječava detekcija vrhova koji su previše blizu jedan drugome. Ovaj pristup može se prilagoditi periodičnosti ljudskog hoda. Duljina prozora definirana je na 200 ms, što odgovara maksimalnoj brzini hodanja od 5 koraka u sekundi, što je brže od najbržeg hoda (3 koraka u sekundi) [16]. Također, računa se sličnost između prethodna dva vrha kako bi se izbjegle velike oscilacije u ubrzanju što sugerira na "lažni korak". Ovo osigurava preciznu i pouzdanu detekciju koraka tijekom različitih brzina kretanja.



Slika 4.6 Detektirani potencijalni vrhovi

U nastavku će biti detaljnije objašnjene svojstva periodičnoti i sličnosti iz rada [25].

Periodičnost

Periodičnost se definira kao vremenska razlika između dva susjedna vrha u očitanjima akcelerometra, odnosno $T_i = t_{peak_{i+1}} - t_{peak_i}$, gdje je T_i periodičnost, a $t_{peak_{i+1}}$ i t_{peak_i} su vremena kada se pojavljuje vrh i kada se uzdiže vrh, redom. Eksperimentalna ispitivanja pokazuju da je periodičnost za isto stanje kretanja (npr. hodanje, trčanje, penjanje ili silaženje) relativno konstantna jer korisnici hodaju ili trče s relativno konstantnom brzinom. Međutim, periodičnost varira kada korisnik miruje dok koristi telefon na različite načine, kao što su slanje poruka ili igranje igara na telefonu. Ograničavanjem raspona periodičnosti hodanja na određeni interval, djelomično se može riješiti problem pretjeranog brojanja koraka. [25]

Sličnost

Vrhunci ubrzanja za dva koraka su prilično slični kada korisnici prirodno hodaju ili trče. Ova sličnost može se izmjeriti razmakom između dva prozora ubrzanja i izražava se kao $s_i = -\|\text{peakacc}(i+2) - \text{peakacc}(i)\|$.

Ovdje, peakacc$(i+2)$ i peakacc(i) predstavljaju dva susjedna vrhunca ubrzanja za lijevi ili desni korak. Razlog za usporedbu senzorskih podataka s iste noge leži u blagim razlikama u očitanjima između lijeve i desne noge. Vrijednost sličnosti obrnuto je proporcionalna udaljenosti između vrijednosti tih dvaju ubrzanja, pri čemu manja udaljenost rezultira većom sličnošću. Kako bi se spriječila visoka sličnost u stanju mirovanja, jednadžba je definirana kao:

$$s_i = \begin{cases} -\infty, & \text{ako je } m_i \text{ ili } m_{i+2} \text{ u stanju mirovanja} \\ -\|\text{peakacc}(i+2) - \text{peakacc}(i)\|, & \text{inače} \end{cases}. \tag{4.1}$$

Ovdje, m_i i m_{i+2} označavaju odgovarajuća stanja kretanja za peakacc(i) i peakacc$(i+2)$, redom. [25]

Logika faze postprocesiranja

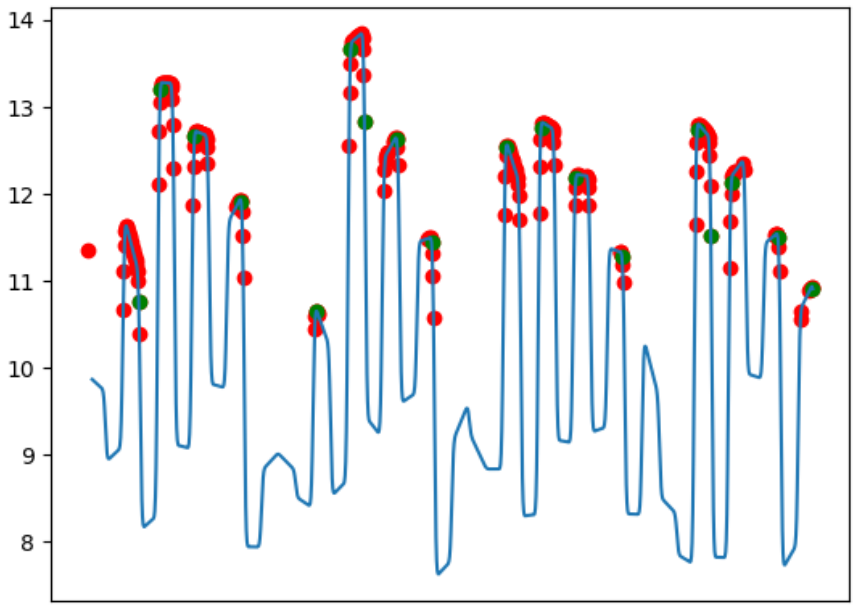
Glavna logika faze radi na sljedeći način:

- Ulazni podaci se kontinuirano čitaju iz `inputQueue` koji sadrži detektirane vrhove iz `DetectionStage`.
- Kada se novi podatak (`dp`) čita, provjerava se je li označen kao kraj podataka. Ukoliko je označen, obavještava se kraj podataka putem sučelja `OnEndOfData` i petlja se završava.
- Ako nije označen kraj podataka, obrađuju se podaci. Postavljaju se inicijalne vrijednosti `current` i `last2` koje služe za prozor podataka.
- Zatim se provjerava razlika u vremenu između trenutnog očitanja (`dp`) i prethodnog koraka (`current`). Ako je ta razlika veća od `timeThreshold` postavljen na 200 milisekundi, pretpostavlja se da je korak potvrđen. Ovdje se

računa i sličnost između trenutnog (`dp`) i prethodna dva koraka (`last2`) pomoću funkcije `calculateSimilarity`.

- Ako sličnost prelazi `similarityThreshold` definiran na -5, korak se bilježi, broji se putem sučelja `OnNewStepDetected`, a trenutni podatak (`dp`) postaje novi korak.
- Ako razlika u vremenu nije dovoljno velika ili sličnost ne zadovoljava uvjet, provjerava se koji od trenutnog podatka (`dp`) i prethodnog koraka (`current`) ima veću magnitudu. Tada se taj podatak označava kao trenutni korak.

Na slici 4.7 prikazan je graf na kojem zelene točke označavaju detektirane vrhove, tj. korake nakon primjene koraka post-procesiranja.



Slika 4.7 Detektirani vrhovi (koraci)

S ovom fazom završava cijela logika algoritma za brojanje koraka. Nakon prolaska kroz sve prethodne faze obrade podataka, uključujući prikupljanje podataka

iz senzora, interpolaciju, filtriranje, ocjenjivanje, detekciju vrhova, vrednovanje sličnosti i analizu periodičnosti kretanja, faza `PostProcessStage` donosi konačnu odluku o broju koraka. Korištenjem parametara kao što su vremenski pragovi i pragovi sličnosti, algoritam identificira relevantne korake i omogućuje precizno mjerenje koraka korisnika u stvarnom vremenu.

Poglavlje 5

Analiza točnosti realizirane aplikacije

Usporedba točnosti razvijene aplikacije s vodećim pedometar aplikacijama, kao što su "Pedometer App" i "Mi Fitness", predstavlja bitan segment u procjeni kvalitete aplikacije za praćenje koraka. Kroz ovu temeljitu analizu, pruža se prilika dublje sagledati performanse i pouzdanost algoritma razvijenog za precizno brojanje koraka te usporediti ih s već etabliranim i popularnim rješenjima za praćenje fizičke aktivnosti dostupnim na tržištu. Ova usporedba pruža vrijedan uvid u sposobnosti aplikacije, njen konkurentni potencijal i mjesto na tržištu mobilnih aplikacija za praćenje zdravlja i tjelesne aktivnosti, te omogućuje bolje razumijevanje njenih prednosti i ograničenja.

5.1 Testiranje

U svrhu evaluacije performansi razvijene aplikacije za brojanje koraka, provelo se testiranje na uzorku od deset sudionika. Ovaj korak je ključan kako bi se dobio dublji uvid u točnost i pouzdanost algoritma, te kako bi se usporedili rezultati s već postojećim pedometar aplikacijama poput "Pedometer App" i "Mi Fitness". Sve tri aplikacije brojale su korake istovremeno. Ovaj odjeljak opisuje metodologiju testiranja, uključujući sudionike, okruženje i provedbu testova.

- **Sudionici**: Testiranje je uključivalo deset različitih sudionika različite dobi i spola (četiri muškarca i šest žena, dobi između 20 i 55 godina). Odabrani sudionici nisu profesionalni sportaši, već obični korisnici koji su predstavljali široku populaciju kako bi se osigurala relevantnost aplikacije za širi krug korisnika.
- **Različiti položaji i okruženja**: Kako bi se simulirale različite svakodnevne situacije, svaki sudionik je testirao tri različita položaja nosivosti mobilnog uređaja. To uključuje držanje uređaja u ruci, nošenje uređaja u džepu jakne i u džepu hlača. Kako bi se eliminirali mogući utjecaj redoslijeda postavljanja, korišten 12x12 balansirani latinski kvadrat za određivanje varijacije redoslijeda položaja i vrste testa kod različitih sudionika. Ovo osigurava da redoslijed postavljanja uređaja i vrste aktivnosti nije pristran i ne utječe na rezultate testiranja, čime se osigurava pouzdanost dobivenih podataka.
- **Testiranje aktivnosti**: Svaki od sudionika je prošao kroz niz testnih aktivnosti, uključujući:
  - Hodanje: Svaki sudionik je hodao na udaljenosti od 150 koraka. Ovo predstavlja uobičajenu svakodnevnu aktivnost i temeljnu svrhu aplikacije.
  - Trčanje: Sudionici su trčali na udaljenosti od 50 koraka. Ova aktivnost simulira brže korake i veći intenzitet.
  - Hodanje po stepenicama: Sudionici su se popeli i spustili 2 kata stepenicama, simulirajući svakodnevnu aktivnost penjanja stepenicama koja zahtijeva dodatni napor.
  - "Treasure Hunt" aktivnost: U ovoj aktivnosti, sudionici su morali slijediti specifične upute na pet različitih lokacija međusobno udaljenih 20 koraka. To uključuje brojanje do 10, izvođenje 3 čučnja, brojanje do 5, izvođenje kruga oko sebe i podizanje papira s poda. Ova aktivnost kombinira različite kretnje i promjene ritma što bi trebalo simulirati prirodno ljudsko kretanje.
- **Brojanje koraka**: Tijekom svih aktivnosti, aplikacija je kontinuirano pra-

tila i bilježila broj koraka korisnika, dok su sudionici također sami brojali vlastite korake kako bi se omogućila usporedba i provjera preciznosti rezultata.

Ovo testiranje rezultiralo je bogatim skupom podataka koji će omogućiti dublje razumijevanje performansi naše aplikacije u različitim scenarijima i aktivnostima. Prikupljeni podaci služit će kao osnova za usporedbu s rezultatima postignutim od strane vodećih pedometar aplikacija. Analizom ovih rezultata moći će se ocijeniti preciznost našeg razvijenog algoritma te istražiti njegove prednosti i ograničenja u usporedbi s konkurencijom.

5.2 Statistička analiza i rezultati

S obzirom na prirodu našeg eksperimenta, odlučeno je statistički obraditi podatke pomoću testa Two-way Repeated Measures ANOVA. Two-way RM ANOVA je statistički test koji se koristi kada isti ispitanici sudjeluju u različitim postavkama eksperimenta s više nezavisnih varijabli. Točnije, koristi se ANOVA test kako bismo saznali postoji li interakcija između dviju nezavisnih varijabli na zavisnu varijablu. U ovom eksperimentu postoje dvije nezavisne varijable: položaj uređaja i vrsta testne aktivnosti, za koje se želi utvrditi utječu li na zavisnu varijablu, odnosno izmjerenog broja koraka korisnika. Faktor položaja uređaja ima tri razine, ruka, džep jakne i džep hlača, dok faktor vrste aktivnosti ima četiri razine, hodanje 150 koraka, trčanje 50 koraka, hodanje stepenicama 76 koraka te treasure hunt od 80 koraka.

Različit broj koraka među različitim vrstama aktivnosti predstavlja izazov prilikom provođenja statističke analize putem Two-way Repeated Measures ANOVA. Kako bi se olakšala procjena utjecaja na zavisnu varijablu, odlučena je prilagodba prikupljenih podataka. Umjesto korištenja apsolutnih brojeva koraka, koji variraju ovisno o vrsti aktivnosti, odlučen je izračun postotka izmjerenog broja koraka u odnosu na unaprijed zadani broj. Ova prilagodba omogućuje stvaranje zajedničke mjere koja omogućava usporedbu između različitih aktivnosti i olakšava analizu njihovog utjecaja na zavisnu varijablu. Postotak izmjerenih koraka od zadane

vrijednosti pomaže u eliminiranju inherentnih razlika u broju koraka među aktivnostima, omogućujući bolje sagledavanje relativne performanse sudionika neovisno o specifičnostima svake vrste aktivnosti.

Statistički alat IBM SPSS detaljno je obradio dobivene podatke. SPSS je generirao različite tablice povezane sa zavisnom varijablom broja koraka. Neke od tih tablica su: tablica normalnosti, tablica s deskriptivnom statistikom koja sadrži srednje vrijednosti podataka i standardnu devijaciju, Mauchlyev test sferičnosti koji pruža informacije o narušenoj sferičnosti, tablica Testova između-subjektnih efekata koja daje uvid u značajnost i F podatke za svaku nezavisnu varijablu i njihovu interakciju, parne usporedbe itd. Nakon opisa tehnika korištenih za analizu, prikazani su dobiveni rezultati i donesen je zaključak.

Naša aplikacija

Kako bi se ispravno izveo Two-way RM ANOVA test, raspodjela zavisne varijable u svakoj kombinaciji povezanih grupa trebala bi biti približno normalno distribuirana. Stoga se koristi Shapiro-Wilk test normalnosti pomoću čijih se koeficijenata može odrediti jesu li podaci normalno distribuirani. Na slici 5.1 se vidi kako su svi podaci normalno distribuirani ako se gleda da su koeficijenti u stupcu Shapiro-Wilk veći od 0.05 ($p > 0.05$).

U tablici na slici 5.2, dovivene su prosječne vrijednosti i standardne devijacije za svih 12 postavki našeg eksperimenta.

Korišten je Mauchlyev test sferičnosti kako bi se utvrdilo je li sferičnost prekršena ili ne. Mauchlyev test sferičnosti provjerava je li matrica kovarijanca pogrešaka ortogonalno transformiranih zavisnih varijabli proporcionalna jediničnoj matrici. Ovaj test pokušava utvrditi jednakost varijanci između različitih uvjeta ili razina faktora u eksperimentalnom dizajnu. Ako je sferičnost prekršena, stupnjevi slobode moraju se prilagoditi upotrebom jednog od korekcijskih testova (npr. Greenhouse-Geisser). Tablica na slici 5.3 prikazuje sve parametre korištene za Mauchlyev test (Mauchlyev W, aproksimacija hi-kvadrat, stupnjevi slobode i p-vrijednost), kao i epsilon vrijednosti za Greenhouse-Geisser, Huynh-Feldt i Lower-bound testove. S obzirom da oba faktora imaju više od 2 razine, proučava se

Tests of Normality

	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
hod_ruka	,137	10	,200*	,972	10	,911
hod_jakna	,154	10	,200*	,975	10	,933
hod_hlace	,193	10	,200*	,940	10	,555
trc_ruka	,129	10	,200*	,966	10	,849
trc_jakna	,197	10	,200*	,916	10	,321
trc_hlace	,155	10	,200*	,969	10	,886
step_ruka	,146	10	,200*	,948	10	,646
step_jakna	,178	10	,200*	,907	10	,258
step_hlace	,226	10	,158	,929	10	,441
treas_ruka	,244	10	,093	,888	10	,160
treas_jakna	,200	10	,200*	,932	10	,466
treas_hlace	,147	10	,200*	,933	10	,479

Slika 5.1 Test normalnosti

Descriptive Statistics

	Mean	Std. Deviation	N
hod_ruka	98,13	2,150	10
hod_jakna	106,67	1,832	10
hod_hlace	92,27	1,481	10
trc_ruka	85,60	3,978	10
trc_jakna	81,20	4,541	10
trc_hlace	74,60	3,777	10
step_ruka	95,79	2,617	10
step_jakna	96,71	2,721	10
step_hlace	99,47	2,646	10
treas_ruka	91,38	2,665	10
treas_jakna	90,00	2,700	10
treas_hlace	84,00	2,622	10

Slika 5.2 Deskriptivna statistika

sferičnost i za vrstu testa i za položaj uređaja. Mauchlyev test sferičnosti pokazao je da pretpostavka sferičnosti nije prekršena: W = 0,310; χ^2 = 9,044; p = 0,110 za

Mauchly's Test of Sphericity[a]

Measure: MEASURE_1

Within Subjects Effect	Mauchly's W	Approx. Chi-Square	df	Sig.	Epsilon[b] Greenhouse-Geisser	Huynh-Feldt	Lower-bound
test	,310	9,044	5	,110	,616	,768	,333
polozaj	,994	,045	2	,978	,994	1,000	,500
test * polozaj	,115	14,682	20	,825	,644	1,000	,167

Slika 5.3 Mauchlyev test sferičnosti

vrstu testa i W = 0,994; $\chi^2 = 0{,}045$; p = 0,978 za položaj. Budući da p-vrijednosti nisu manje od 0,05, nema potrebe za modificiranjem stupnjeva slobode.

Tests of Within-Subjects Effects

Measure: MEASURE_1

Source		Type III Sum of Squares	df	Mean Square	F	Sig.	Partial Eta Squared
test	Sphericity Assumed	4801,925	3	1600,642	193,702	<,001	,956
	Greenhouse-Geisser	4801,925	1,847	2599,852	193,702	<,001	,956
	Huynh-Feldt	4801,925	2,303	2085,500	193,702	<,001	,956
	Lower-bound	4801,925	1,000	4801,925	193,702	<,001	,956
Error(test)	Sphericity Assumed	223,113	27	8,263			
	Greenhouse-Geisser	223,113	16,623	13,422			
	Huynh-Feldt	223,113	20,723	10,767			
	Lower-bound	223,113	9,000	24,790			
polozaj	Sphericity Assumed	608,021	2	304,010	52,522	<,001	,854
	Greenhouse-Geisser	608,021	1,989	305,706	52,522	<,001	,854
	Huynh-Feldt	608,021	2,000	304,010	52,522	<,001	,854
	Lower-bound	608,021	1,000	608,021	52,522	<,001	,854
Error(polozaj)	Sphericity Assumed	104,189	18	5,788			
	Greenhouse-Geisser	104,189	17,900	5,821			
	Huynh-Feldt	104,189	18,000	5,788			
	Lower-bound	104,189	9,000	11,577			
test * polozaj	Sphericity Assumed	516,793	6	86,132	14,188	<,001	,612
	Greenhouse-Geisser	516,793	3,866	133,667	14,188	<,001	,612
	Huynh-Feldt	516,793	6,000	86,132	14,188	<,001	,612
	Lower-bound	516,793	1,000	516,793	14,188	,004	,612
Error(test*polozaj)	Sphericity Assumed	327,827	54	6,071			
	Greenhouse-Geisser	327,827	34,796	9,421			
	Huynh-Feldt	327,827	54,000	6,071			
	Lower-bound	327,827	9,000	36,425			

Slika 5.4 Testovi efekata unutar subjekata

Na tablici na slici 5.4 mogu se vidjeti testovi efekata unutar subjekata, koji pokazuju ispravak za prekršaj sferičnosti. U ovom slučaju, ispravak nije potreban, pa se razmatra redak "Sferičnost pretpostavljena" za vrstu testa i položaj. Two-way

RM ANOVA utvrdila je da se točnost broja koraka statistički značajno razlikuje između četiri vrste testne aktivnosti, $F(3, 27)=193.702$, $p<0.001$. Stoga je potrebna post-hoc analiza kako bi se detaljnije utvrdilo između kojih vrsta aktivnosti postoji značajna razlika.

Estimates

Measure: MEASURE_1

test	Mean	Std. Error	95% Confidence Interval	
			Lower Bound	Upper Bound
1	94,544	,376	93,694	95,395
2	80,467	1,014	78,173	82,760
3	96,795	,405	95,879	97,710
4	88,458	,586	87,132	89,784

Slika 5.5 Procjene vrste aktivnosti

Pairwise Comparisons

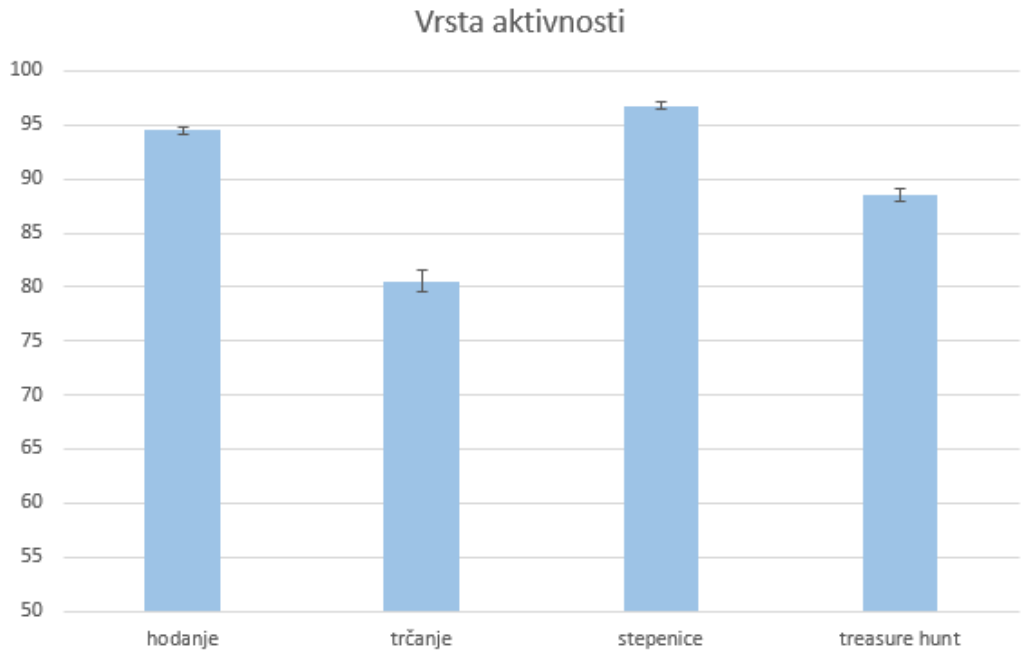
Measure: MEASURE_1

(I) test	(J) test	Mean Difference (I-J)	Std. Error	Sig.[b]	95% Confidence Interval for Difference[b]	
					Lower Bound	Upper Bound
1	2	14,078*	,780	<,001	11,454	16,702
	3	-2,250*	,449	,004	-3,762	-,739
	4	6,086*	,498	<,001	4,411	7,762
2	1	-14,078*	,780	<,001	-16,702	-11,454
	3	-16,328*	1,029	<,001	-19,791	-12,865
	4	-7,992*	,872	<,001	-10,926	-5,057
3	1	2,250*	,449	,004	,739	3,762
	2	16,328*	1,029	<,001	12,865	19,791
	4	8,336*	,653	<,001	6,139	10,534
4	1	-6,086*	,498	<,001	-7,762	-4,411
	2	7,992*	,872	<,001	5,057	10,926
	3	-8,336*	,653	<,001	-10,534	-6,139

Slika 5.6 Parne usporedbe vrsta aktivnosti

Post-hoc testovi korištenjem Bonferroni korekcije tablica na slici 5.6 otkrili su da je aktivnost hodanja značajno točnija od aktivnosti trčanja (94.544±0.376% protiv 80.467±1,014%), kao i značajno točnija od treasure hunt-a (94.544±0.376% protiv 88.458±0.586%) budući da je p-vrijednost manja od Bonferroni-korektirane alfa razine: α=0.0125. Također, sve tri aktivnosti značajno su točnije od aktivnosti trčanja (hodanje 94.544±0.37%, stepenice 96.795±0.405%, treasure hunt 88.458±0.458% protiv 80.467±1,014%). Još je pronađena statistički značajna razlika između aktivnosti hodanja po stepenicama i hodanja (96.795±0.405% protiv 94.544±0.376%), kao i treasure hunt-a (96.795±0.405% protiv 88.458±0.458%), gdje je aktivnost hodanja po stepenicama točnija.

Rezultati prosječne točnosti u odnosu na vrstu aktivnosti, kao i njihove standardne pogreške, prikazani su na grafu sa slike 5.7, a prikupljeni su iz tablice sa slike 5.5.



Slika 5.7 Graf točnosti vrsta aktivnosti

Tablica sa slike 5.4 također pokazuje da postoji značajna razlika u točnosti i između tri vrste položaja uređaja $F(2, 18)=52.522$, $p<0.001$.

Estimates

Measure: MEASURE_1

polozaj	Mean	Std. Error	95% Confidence Interval	
			Lower Bound	Upper Bound
1	92,683	,570	91,393	93,973
2	90,328	,597	88,978	91,678
3	87,188	,496	86,066	88,309

Slika 5.8 Procjene položaja uređaja

Pairwise Comparisons

Measure: MEASURE_1

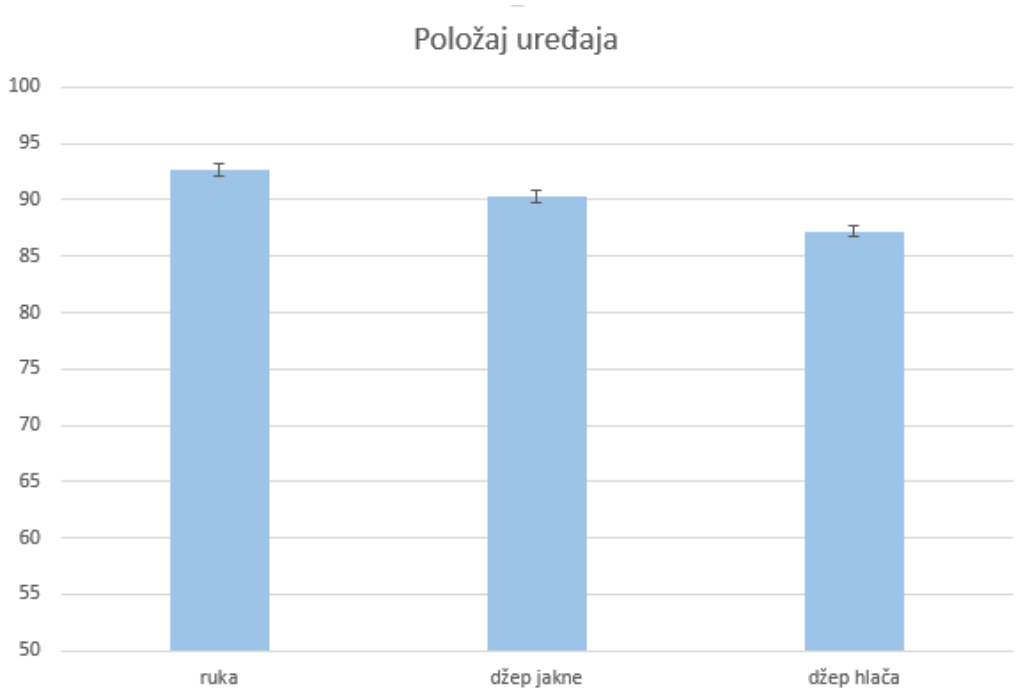
(I) polozaj	(J) polozaj	Mean Difference (I-J)	Std. Error	Sig.[b]	95% Confidence Interval for Difference[b]	
					Lower Bound	Upper Bound
1	2	2,355*	,547	,006	,751	3,960
	3	5,495*	,549	<,001	3,885	7,105
2	1	-2,355*	,547	,006	-3,960	-,751
	3	3,140*	,518	<,001	1,622	4,658
3	1	-5,495*	,549	<,001	-7,105	-3,885
	2	-3,140*	,518	<,001	-4,658	-1,622

Slika 5.9 Parne usporedbe položaja uređaja

Iz tablice sa slike 5.9 može se vidjeti da su položaji uređaja u ruci i u džepu jakne značajno točniji od položaja uređaja u džepu hlača (ruka 92.683±0.570%, jakna 90.328±0.5597% protiv 87.188±0.496%) budući da je p-vrijednost manja od Bonferroni-korektirane alfa razine: α=0.0167. Također, položaj uređaja u ruci značajno je točniji od položaja uređaja u džepu jakne (92.683±0.570% protiv 90.328±0.597%).

Rezultati prosječne točnosti u odnosu na položaj uređaja, kao i njihove standardne pogreške, prikazani su na grafu sa slike 5.10, a prikupljeni su iz tablice sa slike 5.8.

Također, tablica sa slike 5.4 pokazuje da postoji značajna razlika u točnosti interakcije vrste aktivnosti i položaja (test*polozaj) $F(6, 54)=14.188$, $p<0.001$.



Slika 5.10 Graf točnosti položaja uređaja

Može se vidjeti kako je u položaju uređaja u ruci aktivnost hodanja značajno točnija od trčanja i treasure hunt-a (97.967±0.629% protiv 85.600±1.258% i 91.375±0.843%), dok u usporedbi s hodanjem po stepenicama nije statistički značajna. Aktivnost trčanja značajno je netočnija od ostale 3 aktivnosti (85.600±1.258% protiv - hodanje 97.967±0.629%, stepenice 95.7895±0.828% i treasure hunt 91.375±0.843%). Razlika između aktivnosti hodanja po stepenicama i treasure hunt-a nije statistički značajna.

Što se tiče položaja uređaja u džepu jakne, aktivnost trčanja značajno je netočnija od ostale tri aktivnosti (81.200±1.436% protiv hodanje 93.400±0.581%, hodanje po stepenicama 96.711±0.861% te treasure hunt 90.000±0.854%). Također, za razliku od položaja uređaja u ruci, aktivnost hodanja po stepenicama statistički je točnija od aktivnosti treasure hunt-a (96.711±0.861% protiv 90.000±0.854%). Razlika između aktivnosti hodanja po stepenicama i hodanja nije statistički značajna.

U položaju uređaja u džepu od hlača, pronađena je značajna razlika u toč-

nosti između svih aktivnosti. Pa tako se može vidjeti da je značajno najtočnija aktivnost hodanja po stepenicama (97.884±0.503%), zatim aktivnost hodanja (92.267±0.468%), nakon nje aktivnost treasure hunt-a (84.000±0.829%) te je značajno najnetočnija aktivnost trčanja (74.600±1.194%).

Usporedba točnosti s vodećim pedometar aplikacijama

Za usporedbu točnosti, kao što je i spomenuto ranije, odabrane su dvije vodeće pedometar aplikacije, "Mi Fitness" i "Pedometer App". Razlog odabira aplikacije "Mi Fitness" proizlazi iz činjenice da je proizvođač testiranog uređaja Xiaomi, istovremeno i razvio tu aplikaciju, što može pružiti relevantan uvid u performanse. S druge strane, aplikacija "Pedometer App" je odabrana zbog iznimno pozitivnih ocjena i komentara korisnika u trgovini aplikacija, što ukazuje na visok stupanj zadovoljstva i povjerenja korisnika u njenu preciznost i funkcionalnosti. Kombinacija ove dvije aplikacije za usporedbu doprinosi sveobuhvatnoj evaluaciji performansi razvijene aplikacije za brojanje koraka. Kako bi se vidjelo ima li značajne razlike između sve tri aplikacije, radi se test Three-way repeated measures ANOVA. Srednje vrijednosti točnosti prikazane su u tablici na slici 5.11. Brojevi označavaju redom: aplikacija (naša, "Mi Fitness", "Pedometer App"), aktivnost (hodanje, trčanje, stepenice, treasure hunt) i položaj (u ruci, u džepu jakne, u džepu hlača).

Detaljnije su proučene značajne razlike unutar svake vrste aktivnosti i položaja uređaja:

- Hodanje: U položaju uređaja u ruci nema značajne razlike u točnosti između aplikacija, međutim u položaju džepa jakne aplikacija "Mi Fitness" značajno je točnija od naše aplikacije. U položaju džepa hlača aplikacija "Pedometer App" značajno je točnija od obje aplikacije te je naša aplikacija značajno točnija od aplikacije "Mi Fitness". Podaci su izvučeni iz tablice sa slike 5.12.

- Trčanje: U svim položajima uređaja aplikacija "Pedometer App" značajno je točnija od naše aplikacije i "Mi Fitness", dok između naše aplikacije i "Mi Fitness" nema značajne razlike u točnosti, kao što se vidi u tablici sa slike

5.13.

- Stepenice: Kao što se vidi na tablici sa slike 5.14, u svim položajima uređaja naša aplikacija i "Pedometer App" značajno su točnije od aplikacije "Mi Fitness", a između njih nema značajne razlike u točnosti.
- Treasure hunt: Kao i kod aktivnosti penjanja po stepenicama, naša aplikacija i "Pedometer App" značajno su točnije od aplikacije "Mi Fitness", dok između njih nema značajne razlike u točnosti. Rezultati su prikazani na tablici sa slike 5.15.

Na tablicama 5.1, 5.2 i 5.3 prikazane su točnosti aplikacija i vrste aktivnosti za svaki položaj uređaja. Može se zaključiti da u većini slučajeva aplikacija "Pedometer App" ima najtočnije rezultate. Međutim, u situaciji kada je uređaj u položaju džepa jakne pri aktivnosti hodanja, najtočnija je aplikacija "Mi Fitness". Također, u aktivnosti treasure hunt-a kada je uređaj u položaju ruke ili džepa hlača, naša aplikacija ima najveću točnost.

	Hodanje	**Trčanje**	**Stepenice**	**Treasure hunt**
Naša	98%	86%	96%	92%
Mi Fitness	94%	81%	89%	80%
Pedometer App	98%	96%	98%	87%

Tablica 5.1 Tablica točnosti u položaju ruke

	Hodanje	**Trčanje**	**Stepenice**	**Treasure hunt**
Naša	93%	81%	97%	90%
Mi Fitness	99%	78%	90%	69%
Pedometer App	98%	97%	98%	89%

Tablica 5.2 Tablica točnosti u položaju džepa jakne

Ova statistička analiza bit će zaključena s analizom točnosti među aplikacijama. Može se zaključiti da je aplikacija "Pedometer App" značajno točnija od naše i "Mi Fitness" aplikacije, dok je naša aplikacija značajno točnija od aplikacije "Mi

	Hodanje	**Trčanje**	**Stepenice**	**Treasure hunt**
Naša	92%	75%	98%	84%
Mi Fitness	78%	72%	82%	69%
Pedometer App	99%	96%	98%	89%

Tablica 5.3 Tablica točnosti u položaju džepa hlača

Fitness". Prosječna točnost aplikacije "Pedometer App" je 95,1%, naše aplikacije 90,1% te aplikacije "Mi Fitness" 82,1%. Treba uzeti u obzir da su ovdje uzete srednje vrijednosti svih modaliteta te da je na ovaj način prikazan generalni pregled točnosti aplikacija. Kao što je i prikazano u tablicama iznad, svaka aplikacija ima prednosti za specifičnu kombinaciju modaliteta.

Rezultati njihove prosječne točnosti, kao i njihove standardne pogreške, prikazani su na grafu sa slike 5.17, a prikupljeni su iz tablice sa slike 5.16.

Estimates

Measure: MEASURE_1

aplikacija	aktivnost	polozaj	Mean	Std. Error	95% Confidence Interval	
					Lower Bound	Upper Bound
1	1	1	97,967	,629	96,543	99,391
		2	93,400	,581	92,085	94,715
		3	92,267	,468	91,207	93,326
	2	1	85,600	1,258	82,755	88,445
		2	81,200	1,436	77,951	84,449
		3	74,600	1,194	71,898	77,302
	3	1	95,789	,828	93,917	97,662
		2	96,711	,861	94,764	98,657
		3	97,884	,503	96,746	99,022
	4	1	91,375	,843	89,469	93,281
		2	90,000	,854	88,068	91,932
		3	84,000	,829	82,124	85,876
2	1	1	93,600	1,368	90,506	96,694
		2	98,700	,335	97,942	99,458
		3	77,800	,841	75,898	79,702
	2	1	80,800	,727	79,155	82,445
		2	77,700	,883	75,704	79,696
		3	72,100	,737	70,433	73,767
	3	1	88,500	,872	86,526	90,474
		2	89,800	,680	88,262	91,338
		3	81,700	1,300	78,759	84,641
	4	1	79,900	,706	78,302	81,498
		2	75,100	1,080	72,658	77,542
		3	69,200	,757	67,487	70,913
3	1	1	97,500	,373	96,657	98,343
		2	97,800	,593	96,460	99,140
		3	98,700	,300	98,021	99,379
	2	1	96,200	,757	94,487	97,913
		2	97,100	,640	95,652	98,548
		3	95,600	,833	93,716	97,484
	3	1	98,000	,365	97,174	98,826
		2	98,200	,416	97,258	99,142
		3	97,700	,517	96,529	98,871
	4	1	87,100	,722	85,467	88,733
		2	88,400	,968	86,209	90,591
		3	88,900	1,090	86,435	91,365

Slika 5.11 Graf točnosti

aktivnost	polozaj	(I) aplikacija	(J) aplikacija	Mean Difference (I-J)	Std. Error	Sig.[b]	95% Confidence Interval for Difference[b]	
							Lower Bound	Upper Bound
1	1	1	2	4,367	1,581	,066	-,271	9,005
			3	,467	,759	1,000	-1,759	2,693
		2	1	-4,367	1,581	,066	-9,005	,271
			3	-3,900	1,345	,053	-7,846	,046
		3	1	-,467	,759	1,000	-2,693	1,759
			2	3,900	1,345	,053	-,046	7,846
	2	1	2	-5,300*	,803	<,001	-7,657	-2,943
			3	-4,400*	,945	,004	-7,172	-1,628
		2	1	5,300*	,803	<,001	2,943	7,657
			3	,900	,526	,364	-,643	2,443
		3	1	4,400*	,945	,004	1,628	7,172
			2	-,900	,526	,364	-2,443	,643
	3	1	2	14,467*	,900	<,001	11,826	17,107
			3	-6,433*	,526	<,001	-7,976	-4,890
		2	1	-14,467*	,900	<,001	-17,107	-11,826
			3	-20,900*	,737	<,001	-23,062	-18,738
		3	1	6,433*	,526	<,001	4,890	7,976
			2	20,900*	,737	<,001	18,738	23,062

Slika 5.12 ANOVA rezultati za hodanje

2	1	1	2	4,800*	1,227	,011	1,199	8,401
			3	-10,600*	1,077	<,001	-13,759	-7,441
		2	1	-4,800*	1,227	,011	-8,401	-1,199
			3	-15,400*	1,147	<,001	-18,764	-12,036
		3	1	10,600*	1,077	<,001	7,441	13,759
			2	15,400*	1,147	<,001	12,036	18,764
	2	1	2	3,500*	1,118	,036	,220	6,780
			3	-15,900*	1,320	<,001	-19,773	-12,027
		2	1	-3,500*	1,118	,036	-6,780	-,220
			3	-19,400*	,521	<,001	-20,927	-17,873
		3	1	15,900*	1,320	<,001	12,027	19,773
			2	19,400*	,521	<,001	17,873	20,927
	3	1	2	2,500	1,258	,235	-1,191	6,191
			3	-21,000*	1,868	<,001	-26,479	-15,521
		2	1	-2,500	1,258	,235	-6,191	1,191
			3	-23,500*	1,067	<,001	-26,630	-20,370
		3	1	21,000*	1,868	<,001	15,521	26,479
			2	23,500*	1,067	<,001	20,370	26,630

Slika 5.13 ANOVA rezultati za trčanje

3	1	1	2	7,289*	1,330	,001	3,390	11,189
			3	-2,211	1,040	,187	-5,261	,840
		2	1	-7,289*	1,330	,001	-11,189	-3,390
			3	-9,500*	,910	<,001	-12,169	-6,831
		3	1	2,211	1,040	,187	-,840	5,261
			2	9,500*	,910	<,001	6,831	12,169
	2	1	2	6,911*	,984	<,001	4,023	9,798
			3	-1,489	1,085	,609	-4,672	1,693
		2	1	-6,911*	,984	<,001	-9,798	-4,023
			3	-8,400*	,846	<,001	-10,881	-5,919
		3	1	1,489	1,085	,609	-1,693	4,672
			2	8,400*	,846	<,001	5,919	10,881
	3	1	2	16,184*	1,447	<,001	11,941	20,427
			3	,184	,651	1,000	-1,724	2,093
		2	1	-16,184*	1,447	<,001	-20,427	-11,941
			3	-16,000*	1,300	<,001	-19,812	-12,188
		3	1	-,184	,651	1,000	-2,093	1,724
			2	16,000*	1,300	<,001	12,188	19,812

Slika 5.14 ANOVA rezultati za stepenice

4	1	1	2	11,475*	1,195	<,001	7,969	14,981
			3	4,275*	1,086	,010	1,091	7,459
		2	1	-11,475*	1,195	<,001	-14,981	-7,969
			3	-7,200*	1,052	<,001	-10,286	-4,114
		3	1	-4,275*	1,086	,010	-7,459	-1,091
			2	7,200*	1,052	<,001	4,114	10,286
	2	1	2	14,900*	1,502	<,001	10,494	19,306
			3	1,600	1,364	,813	-2,402	5,602
		2	1	-14,900*	1,502	<,001	-19,306	-10,494
			3	-13,300*	1,391	<,001	-17,380	-9,220
		3	1	-1,600	1,364	,813	-5,602	2,402
			2	13,300*	1,391	<,001	9,220	17,380
	3	1	2	14,800*	1,199	<,001	11,282	18,318
			3	-4,900*	1,494	,029	-9,281	-,519
		2	1	-14,800*	1,199	<,001	-18,318	-11,282
			3	-19,700*	1,334	<,001	-23,612	-15,788
		3	1	4,900*	1,494	,029	,519	9,281
			2	19,700*	1,334	<,001	15,788	23,612

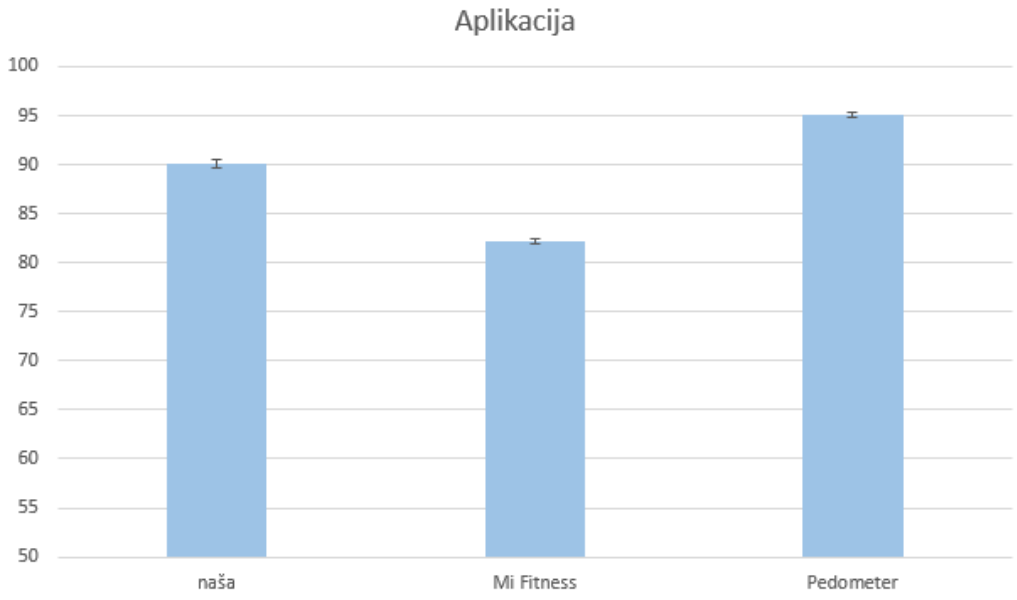
Slika 5.15 ANOVA rezultati za treasure hunt

Estimates

Measure: MEASURE_1

aplikacija	Mean	Std. Error	95% Confidence Interval	
			Lower Bound	Upper Bound
1	90,066	,461	89,023	91,109
2	82,075	,258	81,491	82,659
3	95,100	,216	94,611	95,589

Slika 5.16 Procjene aplikacija



Slika 5.17 Graf točnosti aplikacija

Poglavlje 6

Zaključak

U okviru ovog diplomskog rada detaljno su istraženi inercijski senzori pametnih telefona te primijenjeni najnoviji algoritmi za detekciju koraka s ciljem razvoja Android aplikacije koja bi omogućila precizno brojanje koraka. Kroz temeljitu analizu dostupne literature i pažljiv odabir znanstvenih članaka, uspješno su implementirani algoritmi koji se oslanjaju isključivo na akcelerometar, što je omogućilo postizanje visoke točnosti u praćenju koraka putem mobilnih uređaja.

Naša aplikacija rezultat je integracije algoritama iz dva znanstvena članka, čime se postigla sinergija različitih pristupa u detekciji koraka. Unatoč postignutim napredcima, tijekom provedenoga testiranja i usporedbe s popularnim pedometar aplikacijama, primijećena su određena ograničenja u točnosti naše aplikacije. Statistička analiza rezultata pokazala je manju preciznost u odnosu na jednu referentnu aplikaciju, dok je istovremeno nadmašena drugu, što sugerira potrebu za daljnjim poboljšanjima.

Ovaj rad nije samo dao svoj doprinos tehnološkom napretku u praćenju tjelesnih aktivnosti putem pametnih telefona, već je također naglašeno koliko je bitno stalno istraživanje, usavršavanje algoritama i suradnja s korisnicima da bi se dobile korisne povratne informacije. Uz to, skrenuta je pažnja na to koliko su aplikacije za brojanje koraka važne za podršku ljudima u održavanju zdravog života i podizanju svijesti o važnosti kretanja.

Bibliografija

[1] Varun Nagpal, *Android Sensor Programming By Example*, Packt Publishing Ltd., 2016, ISBN 978-1-78528-550-9.

[2] Android Developers. "Sensors Overview". https://developer.android.com/guide/topics/sensors/sensors_overview

[3] O. Woodman. *An introduction to inertial navigation.* Technical Report 696, University of Cambridge, Computer Laboratory, 2007.

[4] Yin Yan, Shaun Cosgrove, Ethan Blanton, Steven Y. Ko, Lukasz Ziarek: *Real-Time Sensing on Android.* SUNY Buffalo, Fiji Systems Inc.

[5] J. J. Carollo and D. J. Matthews. "Chapter 16: The Assessment of Human Gait, Motion, and Motor Function." In: Pediatric Rehabilitation: Principles & Practice. Demos Medical Publishing, LLC., 2009, pp. 461 –491. isbn: 9781933864372.

[6] Brajdic, A.; Harle, R. Walk detection and step counting on unconstrained smartphones. In Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing, Zurich, Switzerland, 8–12 September 2013.

[7] Lee, H.; Choi, S.; Lee, M. Step Detection Robust against the Dynamics of Smartphones.

[8] Sheu, J.S.; Huang, G.S.; Jheng, W.C.; Hsiao, C.H. Design and Implementation of a Three-Dimensional Pedometer Accumulating Walking or Jogging Motions. In Proceedings of the 2014 International Symposium on Computer, Consumer and Control (IS3C), Taichuang, Taiwan, 10–12 June 2014; pp. 828–831.

[9] Bai, Y.W.; Yu, C.H.; Wu, S.C. Using a three-axis accelerometer and GPS module in a smart phone to measure walking steps and distance. In Proceedings

of the 2014 IEEE 27th Canadian Conference on Electrical and Computer Engineering (CCECE), Toronto, ON, Canada, 4–7 May 2014; pp. 1–6.

[10] Zhong, S.; Wang, L.; Bernardos, A.; Song, M. An accurate and adaptive pedometer integrated in mobile health application. In Proceedings of the IET International Conference on Wireless Sensor Network, Beijing, China, 15–17 November 2010; pp. 78–83.

[11] Song, W.; Lee, J.W.; Lee, B.S.; Schulzrinne, H. Finding 9-1-1 callers in tall buildings. In Proceedings of the IEEE 15th International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM), Sydney, NSW, Australia, 19 June 2014; pp. 1–9.

[12] Ojeda, L.; Borenstein, J. Personal Dead-reckoning System for GPS-denied Environments. In Proceedings of the IEEE International Workshop on Safety, Security and Rescue Robotics, Rome, Italy, 27–29 September 2007; pp. 1–6.

[13] Meng-Shiuan Pan and Hsueh-Wei Lin, "A Step Counting Algorithm for Smartphone Users: Design and Implementation," *IEEE SENSORS JOURNAL*, vol. 15, no. 4, pp. 1234-1241, April 2015.

[14] J. Chon and H. Cha, "LifeMap: A smartphone-based context provider for location-based services," *IEEE Pervasive Comput.*, vol. 10, no. 2, pp. 58-67, Apr./Jun. 2011.

[15] H.-J. Jang, J. W. Kim, and D. H. Hwang, "Robust step detection method for pedestrian navigation systems," *IET Electron. Lett.*, vol. 43, no. 14, pp. 749-751, Jul. 2007.

[16] D. Salvi, C. Velardo, J. Brynes, and L. Tarassenko, "An optimized algorithm for accurate step counting from smartphone accelerometry," in *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, IEEE, 2018, pp. 4423-4427.

[17] H. Wang, S. Sen, A. Elgohary, M. Farid, M. Youssef, and R. R. Choudhury, "No need to war-drive: Unsupervised indoor localization," in Proc. 10th Int. Conf. Mobile Syst., Appl., Services (MobiSys), 2012, pp. 197-210.

[18] S. Beauregard, "A helmet-mounted pedestrian dead reckoning system," in IFAWC '06, 2006, pp. 1-11.

[19] P. Goyal, V. Ribeiro, H. Saran, and A. Kumar, "Strap-down pedestrian dead-reckoning system," in IPIN '11, 2011, pp. 1-7.

[20] M. Vetterli and J. Kovacevic, *Wavelets and Subband Coding*, Prentice Hall PTR, 1995.

[21] C. Torrence and G. P. Compo, "A practical guide to wavelet analysis," *Bull. Amer. Meteor. Soc.*, vol. 79, no. 1, pp. 61-78, 1998.

[22] S. Mallat, *A wavelet tour of signal processing*, Academic Press, 1999.

[23] M. S. Pan and H. W. Lin, "A step counting algorithm for smartphone users: Design and implementation," *IEEE Sensors Journal*, vol. 15, no. 4, pp. 2296-2305, Apr. 2015.

[24] A. Rai, K. K. Chintalapudi, V. N. Padmanabhan, and R. Sen, "Zee: Zero-effort crowdsourcing for indoor localization," in *Proc. 18th Annu. Int. Conf. Mobile Comput. Netw. (MobiCom)*, Aug. 2012, pp. 293-304.

[25] Fuqiang Gu, Kourosh Khoshelham, Jianga Shang, Fangwen Yu, and Zhuo Wei, "Robust and Accurate Smartphone-Based Step Counting for Indoor Localization," *IEEE Sensors Journal*, vol. 17, no. 11, pp. 3454-3463, Jun. 1, 2017.

[26] J. Kwapisz, G. Weiss, and S. Moore, "Cell phone-based biometric identification," in BTAS '10, pages 1–7, 2010.

[27] Z. He, Z. Liu, L. Jin, L.-X. Zhen, and J.-C. Huang, "Weightlessness feature - a novel feature for single tri-axial accelerometer based activity recognition," in ICPR '08, pages 1–4, 2008.

[28] R. Ibrahim, E. Ambikairajah, B. Celler, and N. Lovell, "Time-frequency based features for classification of walking patterns," in DSP '07, pages 187–190, 2007.

[29] S. Pirttikangas, K. Fujinami, and T. Nakajima, "Feature selection and activity recognition from wearable sensors," in UCS '06, pages 516–527, 2006.

[30] M. Edel and E. Köppe. "An advanced method for pedestrian dead reckoning using BLSTM-RNNs." In: 2015 International Conference on Indoor Positioning and Indoor Navigation (IPIN). IEEE. 2015, pp. 1–6.

[31] T. Steinmetzer, I. Bönninger, M. Reckhardt, F. Reinhardt, D. Erk, and C. M. Travieso. "Comparison of algorithms and classifiers for stride detection using wearables." In: Neural Computing and Applications (2019), pp. 1–12.

Sažetak

Diplomski rad istražuje primjenu inercijskih senzora pametnih telefona u kontekstu praćenja koraka, s posebnim naglaskom na akcelerometar kao ključni senzor za preciznost mjerenja. Analiziraju se i ostali relevantni senzori poput senzora pokreta, okoline i položaja, istražujući njihovu međusobnu interakciju. Razmatraju se najnoviji algoritmi za detekciju koraka, obuhvaćajući metode temeljene na parametrima, strojnom učenju i dubokom učenju. Cilj je analizirati prednosti i ograničenja svake metode s naglaskom na odabir optimalnog pristupa za implementaciju u aplikaciji. Proces implementacije Android aplikacije detaljno je opisan, vođen relevantnim znanstvenim člancima. Prikazani su ključni koraci, uključujući metodu detekcije vrhova i pet koraka obrade podataka, od linearnog interpoliranja do odabira pravih vrhova putem periodičnosti i sličnosti. Provedeno testiranje na deset sudionika obuhvatilo je različite aktivnosti i položaje uređaja, s usporedbom rezultata s popularnim aplikacijama poput "Mi Fitness" i "Pedometer App". Statistička analiza rezultata pruža uvid u točnost, utjecaj uvjeta testiranja te ukupnu pouzdanost razvijenog sustava za brojanje koraka.

Ključne riječi **— akcelerometar, detekcija koraka, Android aplikacija**

Abstract

This master's thesis explores the application of inertial sensors in smartphones within the context of step tracking, with a particular emphasis on the accelerometer as a pivotal sensor for precise measurements. Other pertinent sensors such as motion, environmental, and positional sensors are analyzed, investigating their mutual interaction. The latest algorithms for step detection are examined, encompassing methods based on parameters, machine learning, and deep learning. The objective is to analyze the advantages and limitations of each method, with a focus on selecting the most optimal approach for implementation in the application. The process of implementing the Android application is comprehensively described, guided by pertinent scientific articles. Key steps are illustrated, including the

peak detection method and five data processing stages, ranging from linear interpolation to selecting accurate peaks through periodicity and similarity analysis. Testing involving ten participants covered various activities and device positions, with results compared to popular applications like "Mi Fitness" and "Pedometer App". Statistical analysis provides insights into accuracy, the influence of testing conditions, and the overall reliability of the developed step-counting system.

Keywords **— accelerometer, step detection, Android application**