Mikko Kettunen

Kaatumisten tunnistaminen ranne-kellolla - Datankeruu ja algoritmin kehitys

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikan tutkinto-ohjelma

Insinöörityö

28.2.2022

Tiivistelmä

Tekijä:	Mikko Kettunen
Otsikko:	Kaatumisten tunnistaminen rannekellolla - Datankeruu ja algoritmin kehitys
Sivumäärä:	48 sivua
Aika:	28.2.2022
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Tieto- ja viestintätekniikan tutkinto-ohjelma
Ammatillinen pääaine:	Hyvinvointi- ja terveysteknologia
Ohjaajat:	Lehtori Sakari Lukkarinen Device's manager Tapio Selby Principal engineer Marko Nirhola

---

Insinöörityön tavoitteena oli kerätä rannekellolla dataa kaatumisista ja normaaleista aktiviteeteista, ja tämän datan perusteella kehittää algoritmi tunnistamaan kaatumisia. Insinöörityö toteutettiin Navigil Oy:lle.

Työssä dataa kerättiin Navigilin 580 rannekellolla, joka mittaa sekä kiihtyvyyttä että ilmanpainedataa. Datankeruussa määriteltiin 18 kaatumisskenaariota ja 12 normaalin aktiviteetin skenaariota. Datankeruun tuloksena oli 180 kaatumista sisältävää tiedostoa ja 130 normaaleja aktiviteetteja sisältäviä tiedostoja. Kerättyä dataa tutkittiin Python-ohjelmointikielellä. Datasta löydettiin 6 piirrettä, joiden avulla aktiviteetti voitiin määritellä joko kaatumiseksi tai normaaliksi aktiviteetiksi.

Dataa tutkiessa huomattiin, että jos kaatumisen lähtöasentona oli tuolilla istuminen tai sängyllä makaaminen ei merkittävää ilmanpaineen kasvua useimmiten pystytty tunnistamaan. Koska ilmanpaineen kasvun todettiin olevan hyvä piirre erottamaan kaatuminen normaalista aktiviteetista, ei näitä kaatumisia otettu huomioon algoritmin kehityksessä ja täten algoritmin tulosten esittelyssä.

Kerätyn datan perusteella kehitettiin algoritmi, joka tunnistaa kaatumisia kiihtyvyys- ja ilmanpainedatan perusteella. Algoritmin kehityksessä priorisoitiin väärien hälytysten välttämistä. Tavoitteena oli, että suurin osa mitatuista kaatumisista tunnistettaisiin ilman, että yhtäkään mitattua normaalia aktiviteettia tunnistettaisiin väärin kaatumiseksi. Algoritmin kehityksessä käytettiin Python-ohjelmointikieltä, mutta algoritmia kehitettiin siten, että se olisi mahdollisimman helppo kääntää C-ohjelmointikielelle, jolla Navigilin 580 rannekellon toiminnallisuudet on ohjelmoitu.

Kehitetty algoritmi tunnisti 79 % mitatuista kaatumisista ilman, että yhdestäkään normaalista aktiviteetista tuli väärää hälytystä. Tunnistustarkkuuteen oltiin tyytyväisiä, ja algoritmi tullaan lähitulevaisuudessa implementoimaan Navigil 580 -rannekelloon.

Avainsanat: Kaatumisentunnistus, datankeräys, algoritmi

Abstract

Author:	Mikko Kettunen
Title:	Fall Recognition Watch - Data Collection and Algorithm Development
Number of Pages:	48 pages
Date:	28 February 2022
Degree:	Bachelor of Engineering
Degree Programme:	Information and Communications Technology
Professional Major:	Health Technology
Supervisors:	Sakari Lukkarinen, Senior lecturer Tapio Selby, Device's manager Marko Nirhola, Principal engineer

---

The goal of the study was to collect data from falls and normal activities and based on that data, develop an algorithm to recognize falls. The study was made for Navigil Oy.

The collection of the data was done with Navigil's 580 wristwatch, which measures acceleration and pressure data. 18 scenarios for falls and 12 scenarios for normal activities were defined. Data collection resulted in 180 files, one fall in each file, and 130 files, multiple normal activities in one file. The collected data was analyzed with Python programming language. From the data, 6 features were identified that could together define an activity as either a fall or a normal activity.

During the data analysis, it was noted that any significant rise of atmospheric pressure could not often be detected when falling from a chair or bed. Because the rise of the atmospheric pressure was found out to be a good feature to distinguish falls from normal activities, these falls were ignored when the algorithm was developed, and thus they were not included in the algorithm results.

Based on the collected data an algorithm was developed to detect falls. Avoiding false alarms was prioritized when developing the algorithm. The goal was to identify most of the measured falls without misidentifying any of the measured normal activities as falls. The algorithm was developed with Python, but it was developed in a way that it could be easily translated to C-programming language which is the language used to program any functionalities to the wristwatch.

The developed algorithm detected 79 % of the measured falls without falsely detecting any normal activity as a fall. The accuracy was high enough for the algorithm to be implemented to the wristwatch in near future.

Keywords: Fall detection, data collection, algorithm

Sisällys

Lyhenteet ja käsitteet

Dataframe: Python-ohjelmointikielen Pandas-kirjaston kaksiulotteinen kolumneihin ja riveihin jakautuva tietorakenne.

FIFO: *First in, first out.* Muistin kontekstissa datankäsittelyn menetelmä, jossa muistin vanhin, eli muistiin ensimmäiseksi sisään tullut arvo poistuu, kun muisti on täynnä ja siihen lisätään uusi arvo.

MEMS: *Micro-Electro-Mechanical Systems.* Prosessiteknologia, jonka avulla luodaan mikroskooppisen kokoisia laitteita tai systeemeitä.

mg: *Milli-g-voima*. Kiihtyvyyden yksikkö, jossa 1 g, eli 1000 mg vastaa paikallaan oloa maapallon pinnalla.

1 Johdanto

Maailmanlaajuisesti tapahtuu joka vuosi noin 37,3 miljoonaa kaatumista, jotka vaativat lääkinnällistä hoitoa. Vaikka suurin osa kaatumisista ei johda kuolemaan, on kaatuminen silti toiseksi yleisin kuolemaan johtava tapaturma maailmassa 684 000 vuosittaisella tapauksella. Vain liikenneonnettomuudet ovat tappavimpia noin 1,3 miljoonalla vuosittaisella tapauksella. Ikä on yksi tärkeimmistä riskitekijöistä kaatumisista johtuvissa loukkaantumisissa. Mitä vanhempi henkilö on, sitä todennäköisemmin kaatuminen johtaa jonkinlaiseen vammaan. [1; 2.]

Kaatumisia tunnistaville laitteille on siis tarvetta. Tunnistamalla kaatuminen sen tapahtuessa, voidaan kaatuneelle henkilölle lähettää ajoissa apua. Näin kaatunut henkilö pääsee nopeammin hoitoon, mikä nopeuttaa paranemisprosessia sekä vähentää pysyvien vaurioiden aiheutumisen todennäköisyyttä.

Tässä insinöörityössä kerätään dataa kaatumisista ja normaaleista arjen aktiviteeteista Navigil 580 -rannekellolla ja tämän datan avulla pyritään kehittämään kyseiseen kelloon implementoitua kaatumisentunnistusalgoritmia. Ensimmäisenä tavoitteena on luoda hyvin dokumentoitu datapaketti, jonka perusteella datan keräystä voidaan jatkaa myös tulevaisuudessa. Toisena tavoitteena on kehittää algoritmia siten, että se tunnistaisi kaatumisia paremmin ilman, että vääriä hälytyksiä tulisi lisää.

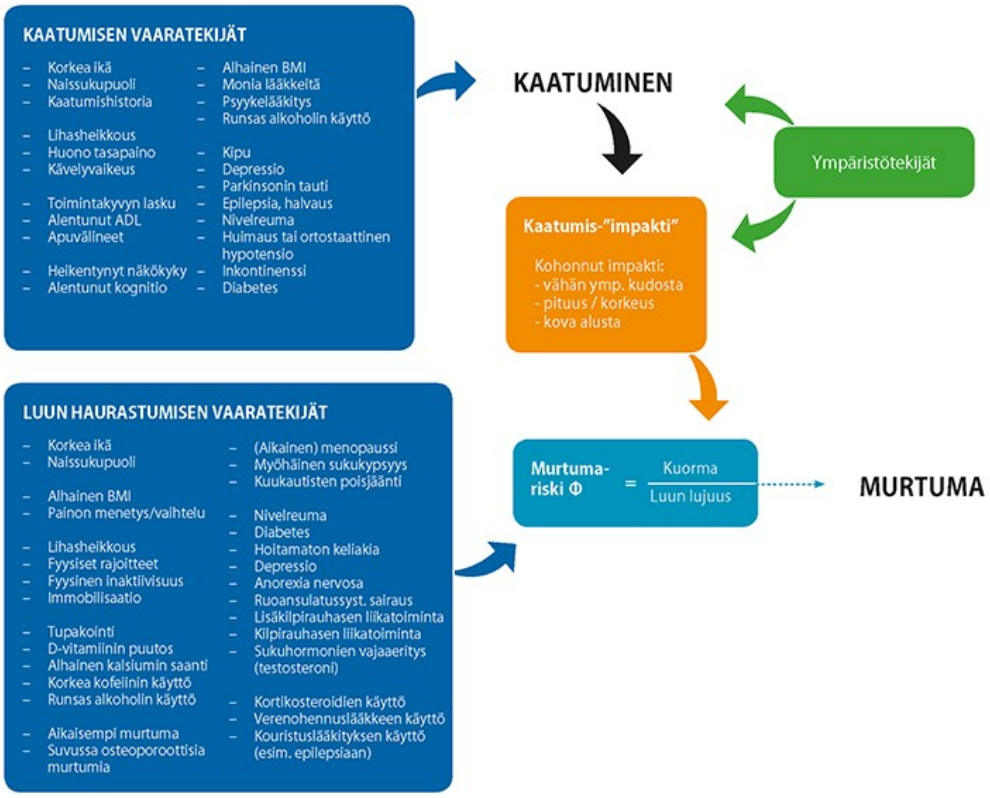
Tämän insinöörityön tilaaja Navigil Oy on suomalainen teknologiayritys, joka valmistaa puettavia hyvinvointituotteita ja palveluita. Yrityksen pääkonttori sijaitsee Espoossa. Navigil 580 -rannekello on pääosin vanhemmille ihmisille tarkoitettu turvalaite, joka pystyy muun muassa tekemään sekä normaaleja, että hätäpuheluita, mittaamaan sykettä ja seuraamaan käyttäjälle asetettuja turva-alueita. Kello toimii yhdessä Navigil Service -nimisen web-sovelluksen kanssa, josta käyttäjälle voidaan asettaa esimerkiksi edellä mainittuja turva-alueita tai seurata sykkeestä muodostettuja hyvinvointiin liittyviä tietoja.

2 Teoriaa ja taustaa

2.1 Kaatuminen

Kaatumisia tapahtuu kaikissa ikäryhmissä, mutta ikäihmisillä on suurin riski loukkaantua kaatumisen johdosta. Maailmanlaajuisesti 30 % 65-vuotiaista tai tätä vanhemmista ihmisistä kaatuvat ainakin kerran vuodessa. Sama luku on melkein 40 % yli 85-vuotiailla. Pelkästään Suomessa kuolee vuosittain 1100 ihmistä kaatumisen seurauksena, ja näistä 90 % on ikäihmisiä. Ikäihmisten kaatumisista 12–42 % johtaa jonkinlaiseen loukkaantumiseen, joista jopa 20 % vaatii lääketieteellistä hoitoa. Voidaan siis sanoa, että kaatumiset ovat merkittävä ongelma, ja ne voivat aiheuttaa paljonkin vaivoja ja ongelmia, varsinkin vanhemmissa ikäluokissa. [3, s. 1–3.]

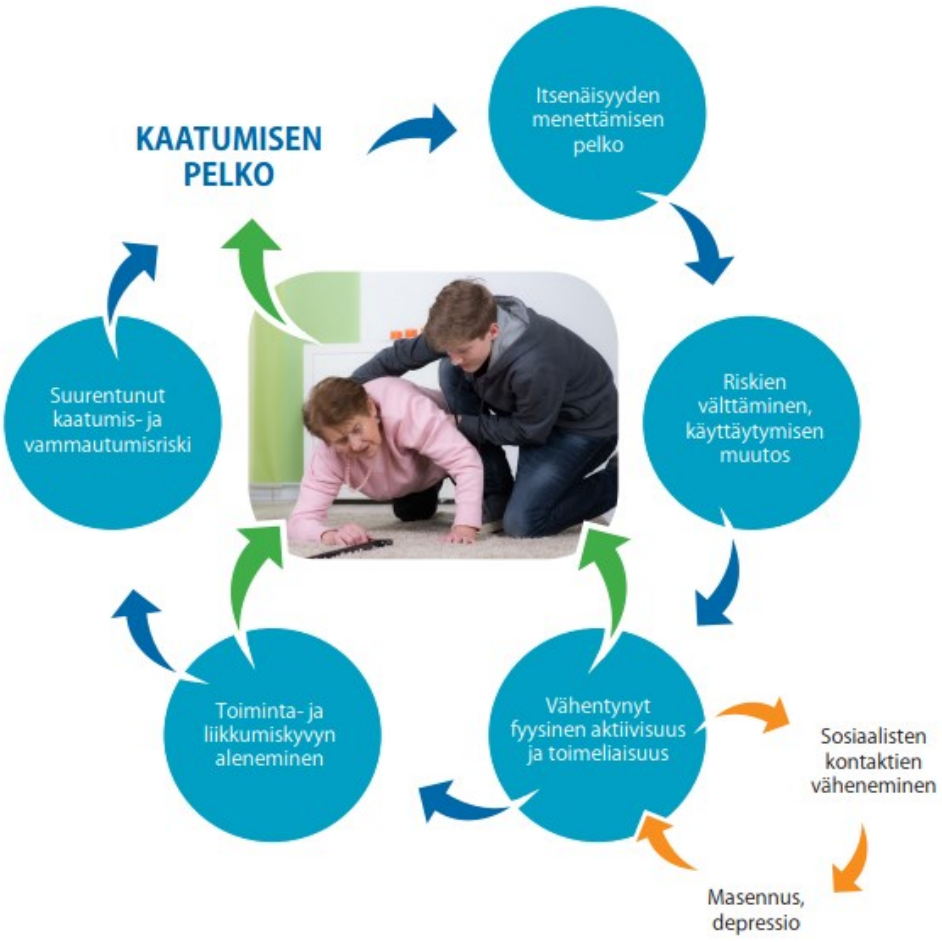
Kaatumisen ja kaatumisesta johtuvan loukkaantumisen riskiä nostavat iän lisäksi myös monet muut tekijät. Kuvassa 1 on Suomen Fysioterapeutit -ammattiliiton tuottama kuva edellä mainituista riskeistä.



Kuva 1. Kaatumisen riskitekijät [4].

Kuvasta 1 voi nähdä, että kaatumisen ja siitä johtuvien loukkaantumisten riskiä nostavia tekijöitä on todella paljon. Yhden tekijän tarkkaa vaikutusta kaatumiseen ei voida tarkasti määritellä, mutta mitä useampi riskitekijä henkilöllä on, sitä suurempi riski hänellä on kaatua ja/tai loukkaantua tulevaisuudessa. Sisäisistä, eli ihmiseen itseensä liittyvät riskitekijöistä tärkeimpiä ovat korkea ikä (75+), heikentynyt terveys, alentunut toiminta- ja liikkumiskyky sekä tasapaino- ja kävelyvaikeudet. Ulkoiset tekijät taas liittyvät henkilön ympäristöön, esimerkkinä ulkoisista riskitekijöistä ovat huono valaistus, liukkaat pinnat ja asuinympäristön sekaisuus. Ulkoisten tekijöiden on arvioitu olevan osasyynä jopa puolissa kaatumistapauksissa.

Pelkästään itse pelko kaatumisesta voi myös lisätä kaatumisen riskiä. Kuvassa 2 on kuvattu, miten tämä pelko voi vaikuttaa kaatumisen riskiin.



Kuva 2. Kaatumisen pelon vaikutus kaatumisriskiin [4].

Kuvassa 2 on havainnollistettu, miten kaatumisen pelko voi aiheuttaa noidankehän, jossa pelko ruokkii kaatumisen riskiä lisääviä käyttäytymismalleja, jotka taas lisäävät pelkoa ja niin edelleen. Kaatumisen pelko on yleistä ikäihmisillä, jopa myös heillä, jotka eivät ole kokeneet kaatumisia.

Kaatumisen pelkoa voidaan siis mahdollisesti vähentää jo sillä, että henkilöllä on jokin laite, joka näitä kaatumisia tunnistaa. Eli ottamatta huomioon tällaisen laitteen tarkkuutta tunnistaa kaatumisia, jo sen käyttöönotto voi mahdollisesti vähentää kaatumisen riskiä ja tuoda mielenrauhaa laitteen käyttäjälle.

2.2 Ongelmat kaatumisentunnistuksessa

Kaatumisentunnistus on osa-alueena hyvin sekava ja hajanainen. Alalta puuttuu yleiset hyvinä pidetyt käytännöt niin datankeruussa kuin algoritmien kehityksessä [5, s. 8–9]. Tässä luvussa käsitellään ongelmia, joita muun muassa näiden käytäntöjen puuttuminen aiheuttaa.

Oikeista kaatumisista on hyvin vaikeaa kerätä dataa, varsinkaan niin paljoa, että sitä pystyisi käyttämään hyödyksi. Kaatumiset ovat harvinaisia tapahtumia, ja kattavan datajoukon kerääminen oikeista kaatumisista vaatisi sen, että dataa tallennettaisiin valtavalta määrältä ihmisiä jatkuvasti hyvin pitkältä ajalta. Datasta pitäisi myös jollakin tapaa saada tietoon ne yksittäiset hetket, jolloin kaatuminen on tapahtunut. Tällainen tapa vaatisi valtavasti resursseja ja työvoimaa, mikä ei yksinkertaisesti ole mahdollisuus suurimmalle osalle tutkijoista tai yrityksistä. Tästä johtuen dataa pitää kerätä niin sanotuissa laboratorio-olosuhteissa, eli X määrä koehenkilöitä suorittaa kaatumisia mahdollisimman todenmukaisesti, ja tämän datan perusteella rakennetaan algoritmit tunnistamaan oikeita kaatumisia. Ongelmana on se, että ei ole olemassa vakiintuneita käytäntöjä sille, miten tämä vaihe suoritetaan. Esimerkkejä datankeräyksen osa-alueista, jotka vaihtelevat suuresti tutkimuksista toiseen ovat datankerääjien määrä, heidän fyysiset ominaisuutensa, käytetyt sensorit, sensorien ominaisuudet, sensorien olinpaikat kehossa, kaatumisskenaariot ja alustat, joihin kaadutaan.

Hyvinä pidettyjen käytäntöjen puute ei ole ainoa tekijä, joka vaikeuttaa laadukkaan datan keräämistä. Toisena tekijänä on se yksinkertainen fakta, että ikäihmiset kaatuvat mahdollisesti eri lailla kuin nuoret ja terveet [6, kappale 4.1.4]. Ikäihmisiä ei kuitenkaan voi käyttää datankeruussa, koska tällaisen koehenkilön loukkaantumisriski olisi liian suuri. Nuorien koehenkilöiden käyttö ei kuitenkaan mitätöi datan laatua, mutta nämä edellä mainitut faktat pitää ottaa huomioon, kun pohditaan valmiin algoritmin varmuudesta tunnistaa kaatuminen.

Kuitenkin, jos kehitettävälle laitteelle, sen sensoreille ja niiden olinpaikoille relevanttia ja laadukasta julkista dataa ei ole saatavilla, niin silloin tutkijan tai yrityksen pitää kerätä dataa itse omalla laitteellaan. Loppujen lopuksi, kaatumisentunnistusalgoritmin pitää perustua joihinkin mittauksiin. Tässä insinöörityössä tilanne on juuri tämä, minkä takia tavoitteena on kerätä dataa itse niin laadukkaasti kuin mahdollista.

Yksi asia, mikä tuottaa ongelmia itse kaatumisentunnistuksessa, on sijainti, josta kaatumisia mitataan. Jos kaatumisentunnistus halutaan toimintona lisätä rannekelloon, on kaatumisten mittaussijaintina ranne, joka on yksi vaikeimmista sijainneista tunnistaa kaatumisia. Koska ranne sijaitsee kahden useaan suuntaan liikkuvan nivelen päässä, on sen mahdolliset liikkeet todella moninaiset. Tämä aiheuttaa ongelmia.

Ensimmäisenä ongelmana on, että ranteen asennon perusteella ei voida tehdä mitään johtopäätöksiä siitä, missä asennossa henkilö on. Jos mittauspiste olisi vyötärö, niin silloin jo laitteen asennon perusteella voitaisiin päätellä esimerkiksi, onko henkilö pysty- vai vaaka-asennossa.

Toisena ongelmana on, että ranteella voidaan tehdä lähes ääretön määrä erilaisia liikkeitä tai liikesarjoja, mikä johtaa siihen, että pakollakin jotkin näistä liikkeistä aiheuttavat samankaltaisia kiihtyvyyssignaaleja, joita on todettu syntyvän lähinnä vain kaatumisissa. Mitatuille kiihtyvyysarvoille on siis mahdotonta määritellä sellaisia piirteitä, jotka pätisivät vain ja ainoastaan kaatumisiin, eikä yhteenkään mahdolliseen liikesarjaan, joita ranteella tehdään jossakin normaalissa aktiviteetissa.

Jatkossa voitaisiinkin siis miettiä mahdollisesti jonkin vyötäröön asetettavan lisälaitteen kehittämistä rannekellon rinnalle. Lisälaitetta voitaisiin esimerkiksi markkinoida sellaiselle kohderyhmälle, jonka yksilöillä on suuri riski loukkaantua kaatumisen takia. Ne käyttäjät, jotka eivät kokisi lisälaitteen hankkimista välttämättömänä, voisivat käyttää pelkkää rannekellon kaatumisentunnistustoimintoa.

2.3 Kaatumisentunnistuksessa käytetyt algoritmit

Kaatumisentunnistuksessa käytetyt algoritmit voidaan jakaa karkeasti kahteen ryhmään, jotka ovat perinteiset mitattavien parametrien raja-arvoihin perustuvat algoritmit ja koneoppimisalgoritmit. Tässä luvussa käsitellään näiden algoritmien eroja ja niiden hyviä ja huonoja puolia.

2.3.1 Perinteiset algoritmit

Perinteiset algoritmit ovat sellaisia, joissa kaikki algoritmin toiminnot ovat niin sanotusti ”kovakoodattuja”, eli algoritmin kehittäjä on määritellyt kaikki algoritmin ehdot ja parametrit. Algoritmi ei siis pysty kehittymään tai oppimaan datasta, vaan se toimii vain ennalta määritettyjen sääntöjen puitteissa.

Perinteiset raja-arvoihin perustuvat kaatumisentunnistusalgoritmit seuraavat sensorien mittaamia parametrejä kuten kiihtyvyyttä. Kun parametrit ylittävät tai alittavat tietyt valmiiksi määritellyt raja-arvot, niin algoritmi luokittelee tapahtuman kaatumiseksi. Yksinkertainen esimerkki tällaisesta olisi, että kun kiihtyvyys laskee annetun raja-arvon alle, niin kun tästä hetkestä määritetyn ajan kuluttua kiihtyvyys nousee yli yläraja-arvon, niin algoritmi luokittelisi tapahtuman kaatumiseksi. Todellisuudessa tällaisia raja-arvoja sekä seurattavia parametrejä voi olla paljon enemmän. Perinteisen raja-arvoihin perustuvan algoritmin kehityksessä mitatusta datasta yritetään löytää sellaisia arvoja, jotka ylittyvät tai alittuvat kaatumisissa, mutta eivät muissa arjen tapahtumisissa, kuten hyppimisessä tai käden lyömisessä pöytään.

2.3.2 Koneoppimisalgoritmit

Koneoppimisalgoritmeissa käytetään hyväksi algoritmiin syötettävää dataa, josta algoritmi laskennallisia prosesseja käyttäen muodostaa assosiaatioita syötteen ja tulosten välille. Koneoppimisalgoritmit on koodattu siten, että ne oppivat datasta, jonka avulla ne voivat muuttaa omia parametrejaan. Tällaista oppimista tapahtuu, kun algoritmille syötetään dataa ja datan mukana halutut lopputulokset. Tämä prosessi on algoritmin opetusvaihe. Esimerkiksi algoritmille syötetyssä datassa voi olla vaikkapa kuvia hedelmistä ja joka kuvan mukana annetaan myös hedelmän nimi. Näin algoritmin opetusvaiheessa algoritmi pyrkii itse löytämään kuvista sellaisia yhteisiä piirteitä, joiden avulla se pystyy päättelemään pelkän kuvan perusteella, mikä hedelmä on kyseessä. Koska opetusvaiheessa jokaisen kuvan oikea vastaus on tiedossa, niin algoritmi voi muuttaa mallin parametreja siten, että se löytää parhaan mahdollisen yhteyden kuvan ja oikean vastauksen välillä. Näin ihmisen ei tarvitse koodata jokaista mahdollista ominaisuutta (väri, muoto, koko yms.), joka määrittelee esimerkiksi appelsiinin, vaan algoritmi oppii itse löytämään tämän yhteyden.

Opetusvaiheen jälkeen algoritmille voidaan syöttää uusia kuvia hedelmistä, mutta tällä kertaa ilman oikeita vastauksia. Tätä kutsutaan testausvaiheeksi. Näin saadaan tietoon, kuinka hyvin algoritmin oppimat yhteydet kuvien ja oikeiden vastauksien välillä pätevät, kun oikeaa vastausta ei tiedetä. Testausvaiheen tulosten perusteella voidaan palata takaisin opetusvaiheeseen, jos algoritmi ei ole kehittäjien mielestä oppinut tunnistamaan kuvia tarpeeksi tarkasti. [7, s. 3–5.]

Edellä kuvattu on esimerkki ohjatusta oppimisesta (supervised learning). Tällä tarkoitetaan sitä, että algoritmille annetaan datan lisäksi myös datapisteitä vastaavat oikeat vastaukset. Tämä koneoppimisen kategoria kuvaa mielestäni intuitiivisimmin, mitä koneoppiminen on.

Koneoppiminen voi kuitenkin olla myös ei ohjattua (unsupervised learning). Tässä koneoppimisen kategoriassa algoritmille ei anneta oikeita vastauksia,

vaan algoritmi muodostaa itse syötteiden ja tulosteiden välille yhteyksiä. Klusterointialgoritmit ovat hyvä esimerkki ei ohjatusta koneoppimista. Kyseisille algoritmeille annetaan esimerkiksi datapisteitä koordinaatistossa, ja algoritmi jakaa nämä pisteet omiin ryhmiin, eli klustereihin pisteiden sijaintien perusteella. Tämän jälkeen algoritmi etsii keskipisteen jokaisesta klusterista siten, että keskipisteet ovat keskimäärin mahdollisimman lähellä mahdollisimman montaa datapistettä klusterissa. [8.] Klusterointialgoritmia voidaan käyttää esimerkiksi löytämään hyviä sijainteja yrityksen tuotteiden varastoille, kun tiedossa on asiakkaiden sijainnit. Jos yritys haluaa rakentaa esimerkiksi kolme varastoa, niin klusterointialgoritmilla asiakkaiden sijainnit voidaan jakaa kolmeen klusteriin ja määrittää näiden klustereiden keskipisteet, joihin varastot kannattaisi rakentaa. Toisin kuin ohjatussa koneoppimisessa klusterointialgoritmille ei siis anneta oikeita lopputuloksia ideaaleista sijainneista, vaan algoritmi tekee päätelmät oikeista tuloksista itse annetun datan perusteella.

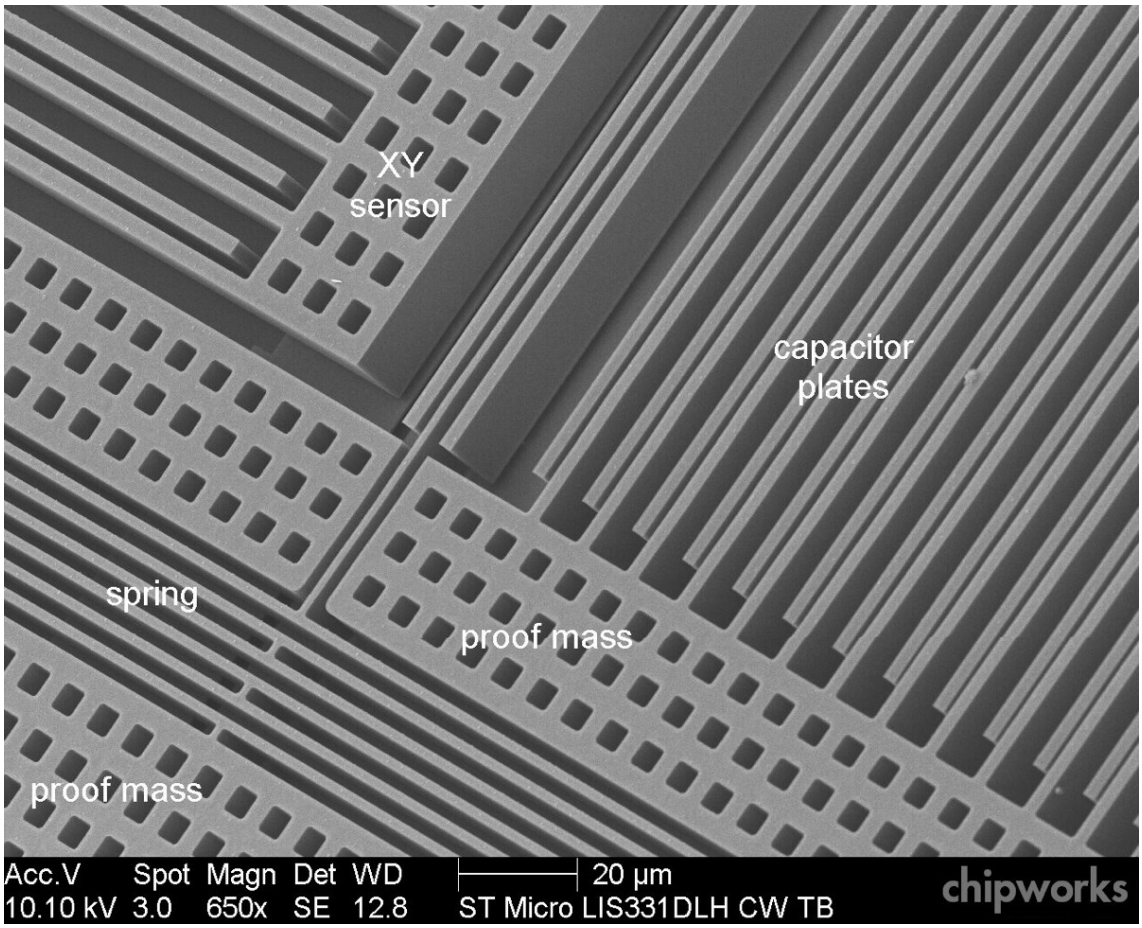
2.3.3 Perinteiset algoritmit vs. koneoppimisalgoritmit

Perinteiset algoritmit vaativat vähemmän laskentatehoa kuin koneoppimisalgoritmit, jolla voi olla paljonkin merkitystä, riippuen laitteen luonteesta, johon algoritmi implementoidaan. Ne ovat myös helpompi ottaa käyttöön, koska koneoppimisalgoritmien kehitys vaatii asiantuntemusta tekoälystä. Perinteisien algoritmien potentiaalinen kaatumisentunnistustarkkuus on kuitenkin huomattu olevan pienempi kuin koneoppimisalgoritmien. Erityyppiset ihmiset kaatuvat eri tavalla ja erilaisia kaatumisskenaarioita on lukemattomia, minkä takia perinteisissä algoritmeissa on hyvin vaikeaa määritellä tiettyjä mitattavien parametrien raja-arvoja, jotka pätisivät kaikkiin mahdollisiin kaatumistyyppeihin. [5, s. 9–14.] Koneoppimisalgoritmien hyvänä puolena on se, että ne pystyvät löytämään sellaisia piirteitä datasta, joita ihmiset eivät välttämättä löytäisi vain tutkimalla dataa.

2.4 Kiihtyvyyssensorin toimintaperiaatteet

Kiihtyvyys on tärkein mitattava suure, jonka perusteella kaatumisia tunnistetaan. Navigil 580 -rannekellon kiihtyvyyssensori perustuu MEMS (Micro-Electro-Mechanical Systems) teknologiaan. MEMS-sensorit ovat kooltaan pieniä, jopa

alle millimetrin kokoisia, joten ne voidaan implementoida mihin tahansa laitteeseen koosta riippumatta. Kuvassa 3 on MEMS-kiihtyvyyssensori kuvattuna elektronimikroskoopilla.



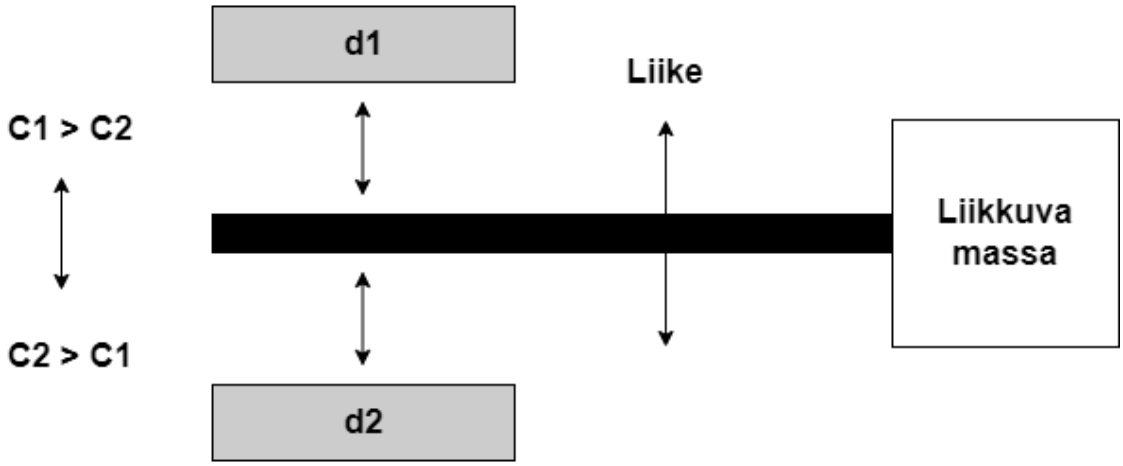
Kuva 3. MEMS-kiihtyvyyssensori kuvattuna elektronimikroskoopilla [9].

Kuvasta 3 saa hyvän käsityksen, kuinka pieniä osia MEMS-sensoreissa voi olla. Kuvan 3 alalaidassa on annettu 0,02 millimetrin (20 µm) mittasuhde, ja monet kuvan osista ovat tätäkin huomattavasti kapeampia. Pienen koon lisäksi muita MEMS-sensoreiden hyviä puolia ovat alhainen hinta ja hyvä mittaustarkkuus. [10.]

Kiihtyvyyssensoreiden mittausperiaate perustuu yleensä joko pietsoelektriseen, pietsoresistiiviseen tai kapasitiiviseen ilmiöön. Pietsoelektrisiä sensoreita käytetään yleisimmin värinän mittaamiseen ja pietsoresistiivisiä sensoreita autojen törmäyksen tunnistamiseen. Kaikkein yleisimmin kiihtyvyyden mittaamiseen

käytetty sensorityyppi on kuitenkin kapasitiivinen sensori, jota myös Navigil 580 -rannekellossa hyödynnetään. [11.]

Kapasitiivinen kiihtyvyyssensori perustuu kapasitanssin muutokseen massan liikkuessa. Kapasitanssilla tarkoitetaan esineen kykyä kerätä ja varastoida energiaa sähkövarauksena. Kuva 4 auttaa ymmärtämään intuitiivisemmin, miten tätä ilmiötä voidaan käyttää hyväksi kiihtyvyysanturissa. Kuvan 4 systeemi on yksinkertaistettu kuvaus kapasitiivisen kiihtyvyyssensorin toimintaperiaatteesta.

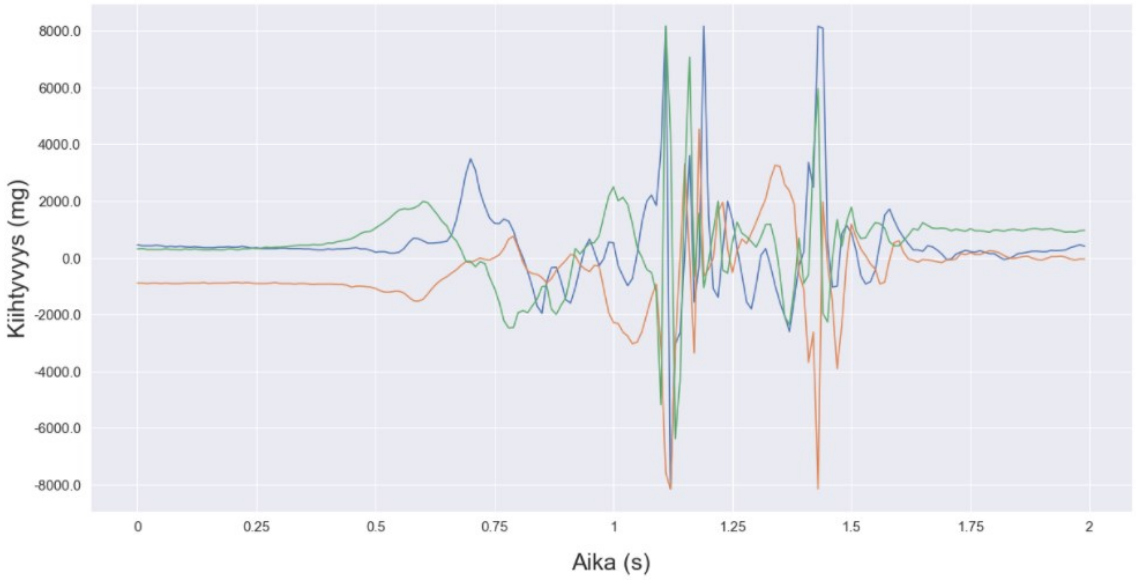


Kuva 4. Kapasitiivisen kiihtyvyyssensorin toimintaperiaate [12, s. 3.]

Kuvassa 4 liike, eli kiihtyvyys aiheuttaa kahden elektrodin välisen massan liikettä kohti jompaakumpaa elektrodia, mikä riippuu kiihtyvyyden suunnasta. Tässä esimerkissä massa on kiinnitettynä jousella kahden elektrodin väliin. Kun massa liikkuu alaspäin, niin etäisyys ylemmästä elektrodista (d1) kasvaa ja etäisyys alemmasta elektrodista (d2) pienenee. Tällöin ylemmän elektrodin kapasitanssi (C1) pienenee ja alemman elektrodin (C2) kasvaa. Näin saadaan tietoa sekä kiihtyvyyden suuruudesta että sen suunnasta. Kapasitanssin muutokset ovat erittäin pieniä, mutta silti mitattavissa. [12, s. 3.]

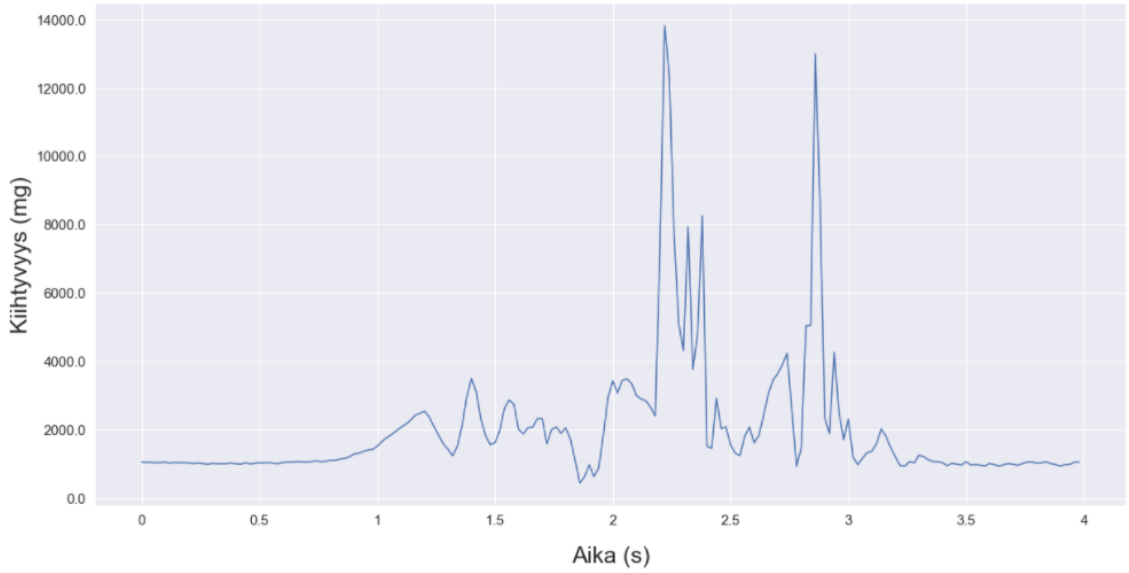
Kuvan 4 systeemissä mittaus tehdään vain 1 akselin suuntaisesti, mutta kiihtyvyyssensoreita on myös 2- ja 3-akselisia. Esimerkiksi Navigil hyödyntää 580 rannekellossaan 3-akselista kiihtyvyyssensoria, joka mittaa kiihtyvyyttä 100

Hz:n taajuudella. Kuvassa 5 on kuvaaja Navigil 580 -rannekellon jokaisen akselin mittaamista kiihtyvyysarvoista kaatumisen aikana. Kuvassa 5 kiihtyvyyden suuruus on kerrottu milli g-voimina (mg).



Kuva 5. Navigil 580 -rannekellon 3-akselisen kiihtyvyyssensorin dataa kaatumisen aikana.

Kuvassa 5 sininen viiva kuvaa x-akselin, oranssi y-akselin ja vihreä z-akselin kiihtyvyysdataa. Koska ranteen asento voi olla lähes mikä tahansa kaatumisen aikana, ei näiden yksittäisen akselin mittaamien arvojen perusteella voida vielä sanoa, kuinka paljon kiihtyvyyttä on kokonaisuudessaan tapahtunut. Navigilin kaatumisentunnistusalgoritmi ei siis tarkastele jokaiselta akselilta mitattua dataa erikseen, vaan näistä jokaisesta 3 akselin mittaamasta kiihtyvyysarvosta muodostetaan yksi ainut kiihtyvyysarvo. Ilman tätä muunnosta joutuisi algoritmi tekemään jatkuvasti laskennallisesti raskaita toimintoja, jotta kokonaiskiihtyvyysarvot, ja niistä tehdyt johtopäätökset saataisiin tietoon. Tämä taas lisäisi merkittävästi virrankulutusta ja veisi resursseja pois muilta Navigil 580 -rannekellon toiminnoilta. Kuvassa 6 näkyy kuvaaja, joka esittää kuvan 5 datasta muodostettua kokonaiskiihtyvyyttä.

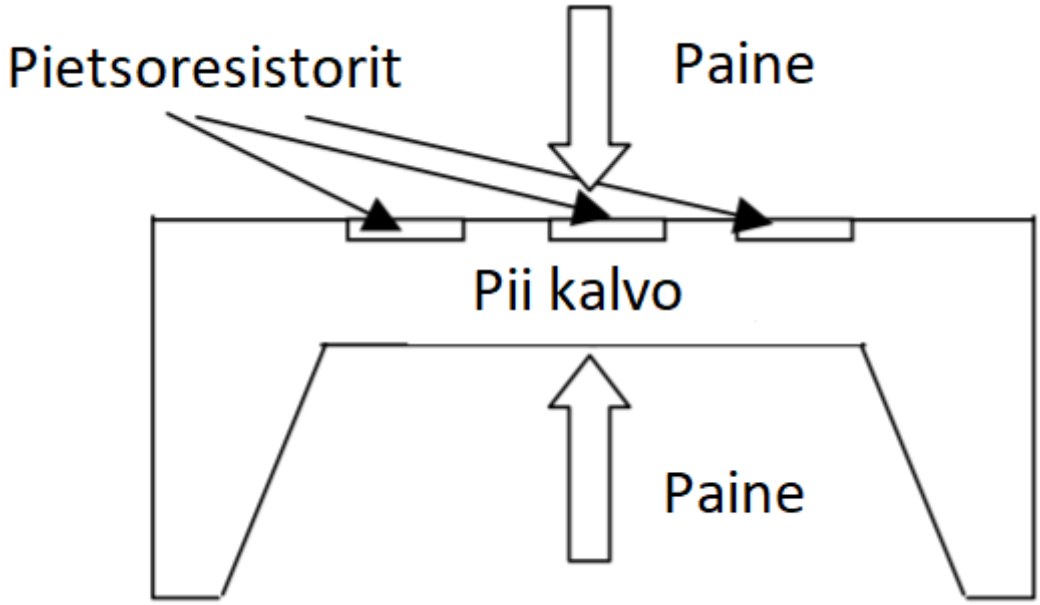


Kuva 6. Navigil 580 -rannekellon kiihtyvyyssensorin kokonaiskiihtyvyyden dataa kaatumisen aikana.

Kuvasta 6 näkyy, kuinka kaikki 3 akselin mittaamat kiihtyvyysarvot on saatu muunnettua yksittäisiksi arvoiksi. On selkeää, että tällainen muunnos hävittää tiedon käyttäjän mahdollisesta asennosta kaatumisen aikana, jota voitaisiin käyttää hyödyksi kaatumisentunnistuksessa. Tämä on kuitenkin kompromissi, joka joudutaan tekemään, jotta kaatumisentunnistus ei toimintona veisi liikaa resursseja.

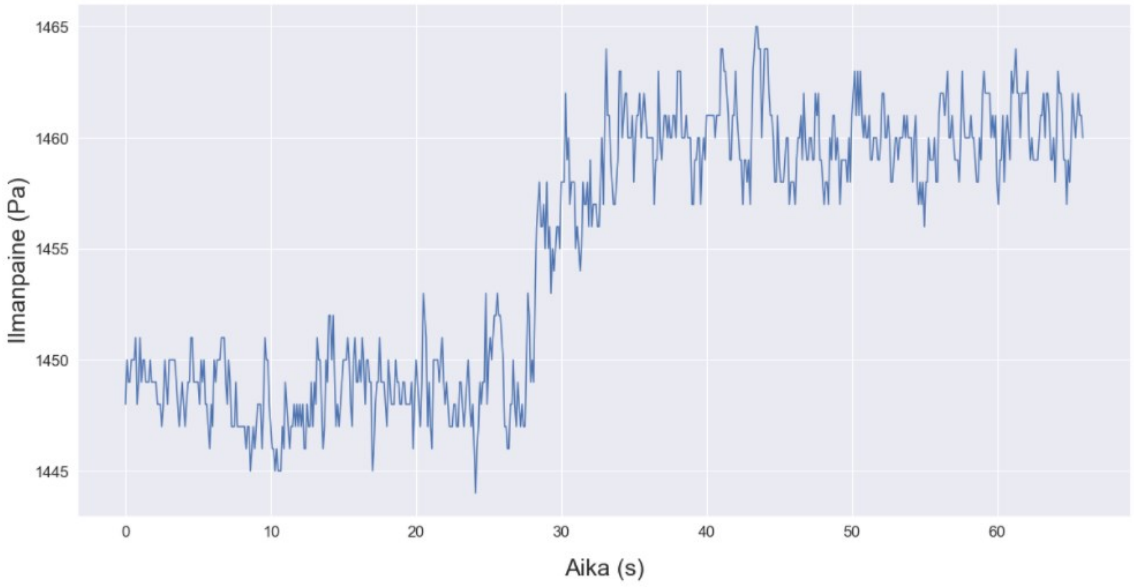
2.5 Painesensorin toimintaperiaatteet

Toinen sensori, jonka dataa Navigil 580 -rannekello käyttää kaatumisen tunnistamiseen, on pietsoresistiivinen MEMS-painesensori. Pietsoresistiivisyys on ilmiö, jossa materiaalin sähköinen resistanssi muuttuu, kun se altistuu mekaaniselle rasitukselle. Kuvassa 7 on esitetty, kuinka pietsoresistiivistä ilmiötä hyödynnetään käytännössä.



Kuva 7. Pietsoresistiivisen sensorin toimintaperiaate [13, s. 1].

Kuvassa 7 piikalvo taipuu ylöspäin tai alaspäin paineen kasvaessa tai laskiessa. Tämä taipuminen muuttaa kalvon muotoa, joka muuttaa pietsoresistorien kokemaa rasitetta, joka muuttaa resistorien resistanssia. Tämä resistanssin muutos on mitattavissa, jonka perusteella pystytään päättelemään resistanssin muutoksen aiheuttaneen paineen suuruus. Kuvassa 7 alhaalta tuleva referenssipaine voidaan aiheuttaa esimerkiksi tyhjiön avulla. Kun tiedetään, kuinka paljon referenssipaine vaikuttaa piikalvon muotoon, niin ulkoinen paine voidaan laskea sen kalvoon aiheuttaman muutoksen perusteella. [13, s. 1–2.] Navigil 580 rannekellon painesensori käyttää tätä tekniikkaa mittaamaan ilmanpainetta. Ilmanpaine kasvaa keskimäärin 11,8 Pascalia per jokainen alaspäin liikuttu metri, joka on enemmän tai vähemmän se matka, jonka kello liikkuu alaspäin kaaduttaessa, riippuen tietenkin kellon käyttäjän pituudesta. Kuvassa 8 esitellään Navigil 580 -rannekellon painesensorin mittaamaa dataa kaatumisen aikana.



Kuva 8. Navigil 580 -rannekellon painesensorin dataa kaatumisen aikana.

Navigil 580 -rannekellon painesensori mittaa dataa 10 Hz:n taajuudella. Kuvasta 8 näkyy, kuinka mitatut arvot vaihtelevat jatkuvasti noin 10 Pascalin välillä, eli ilmanpaineen mittaus ei ole täysin tarkkaa. Noin 30 sekunnin kohdalla tapahtuu kaatuminen, jonka jälkeen ilmanpaineen vaihteluväli siirtyy noin 10 Pascalia ylöspäin.

2.6 Navigilin nykyinen algoritmi

Koska Navigilin nykyinen algoritmi on perinteinen raja-arvoihin perustuva algoritmi enkä henkilökohtaisesti ole tekoälyn asiantuntija, niin tässä insinöörityössä en lähde kehittämään koneoppimismalleja, vaikka tällaisella voitaisiin teoriassa saavuttaa parempi kaatumisentunnistustarkkuus. Koneoppimisalgoritmi vaatisi todennäköisesti myös enemmän laskentatehoa rannekellolta kuin perinteinen raja-arvoihin perustuva algoritmi, jossa piirteitä kiihtyvyys- ja painesignaaleista määritellään lähinnä summaamalla ja erottamalla arvoja. Koneoppimisalgoritmeista kuitenkin löytyy nykyään myös sulautettuihin järjestelmiin tarkoitettuja alhaisen virrankulutuksen versioita, joiden käyttömahdollisuuksia voitaisiinkin tulevaisuudessa arvioida [14].

Navigilin nykyinen algoritmi seuraa jatkuvasti sekä kiihtyvyys- että painesensorin mittaamaa dataa. Kun kiihtyvyysdatasta tunnistetaan sellaisia piirteitä, jotka on mielletty aiheutuneen vapaasta pudotuksesta, niin algoritmi seuraa, nouseeko kiihtyvyys tämän vapaapudotuksen jälkeen yli valmiiksi määritellyn raja-arvon tietyn ajan sisällä. Jos tämä kiihtyvyyspiikki on tarpeeksi korkea, niin viimeinen asia, jonka algoritmi tarkastaa on ilmanpaineen muutos tämän kiihtyvyyspiikin jälkeen. Jos paine on kasvanut yli valmiiksi määritellyn raja-arvon kiihtyvyyspiikin jälkeen, niin tapahtuma luokitellaan kaatumiseksi.

3 Datankeruun menetelmät

Datankeräys jaetaan kahteen eri osioon. Ensimmäisessä osiossa dataa kerätään kaatumisista ja toisessa normaaleista arjen aktiviteeteista, joissa käden liikkeet voitaisiin mahdollisesti tulkita virheellisesti kaatumiseksi. Näin algoritmin kehityksessä pystytään tarkastelemaan sekä kaatumisentunnistustarkkuutta että väärien tunnistusten määrää. Dataa kerätään Navigil 580 -rannekellolla ja molemmissa osioissa toimin itse datankeräyksen koehenkilönä.

3.1 Kaatumisdatan keräys

Kaikki kaatumiset suoritetaan sisätiloissa ja kaatumisalustaksi on valittu Gymstickin jumppapatja, joka näkyy kuvassa 9.

Kuva 9. Kaatumisalustana käytetty Gymstickin jumppapatja [15].

Patja on 2 metriä pitkä, 1 metriä leveä ja 50 millimetriä paksu. Alustan valinnassa kriteereinä oli, että alusta on tarpeeksi kova, jotta koehenkilön automaattiset refleksit tasapainon menetyksen jälkeen olisivat samankaltaiset kuin lattialle tai maahan kaaduttaessa. Myös kaatumisesta aiheutuva isku, jota mitataan kiihtyvyysanturilla, on näin todenmukaisempi, kuin mitä se olisi pehmeämmällä patjalla. 50 mm:n paksuinen patja on kuitenkin tarpeeksi pehmeä, jotta koehenkilölle ei pitäisi aiheutua ruhjeita kaatumisista.

Seuraavaksi on listattu kaatumisdatan keräyksen vaiheet:

1. Kellon asetuksista laitetaan jatkuva kiihtyvyys- ja painedatan tallennus päälle.

2. Koehenkilö asettaa kellon vasempaan käteensä ja suorittaa valitsemansa kaatumisskenaarion. Kaatumisen jälkeen koehenkilö jää makaamaan maahan noin 20-30 sekunnin ajaksi.

3. Koehenkilö nousee maasta, ja lopettaa datantallennuksen.

4. Tallennettu data ladataan tietokoneelle, ja sille annetaan kaatumista vastaava tiedostonimi, joka sisältää kaatumisen lähtöasennon, syyn, suunnan ja monesko mittaus kyseisestä skenaariosta on kyseessä. Esimerkkinä tiedostonimestä olisi: ”walk_trip_forward_3”.

5. Ladatun tiedoston kuvaajasta etsitään kaatumisen hetki 5 sekunnin tarkkuudella.

6. Lopuksi kaatumisesta merkitään ylös metatiedot tekstitiedostoon JSON-formaatissa. Metatiedot sisältävät kaatumistiedoston nimen ja sen luontipäivän, kaatumisen ajankohdan 5 sekunnin tarkkuudella, kaatumisen lähtöasennon, kaatumisen syyn ja kaatumisen suunnan, sekä mahdolliset lisätiedot kaatumistapahtumasta, jotka olisi hyvä ottaa huomioon. Esimerkki kaatumistiedoston metatiedoista:

```
{
     "filename": "walk_slip_forward_6.sdr",
     "time": "2021-09-25  15:00:06",
     "starting": "walk",
     "reason": "slip",
     "direction": "forward",
     "description": ""
}
```


Kaatumisdatan keräyksessä ei siis pelkästään tallenneta yksittäisiä kaatumisdataa sisältäviä tiedostoja, vaan jokaisesta tiedostosta tallennetaan ylös tärkeitä tietoja myöhemmin tapahtuvaa algoritmin kehitystä varten.

Kohdassa 2 syy sille, että maahan jäädään hetkeksi aikaa makaamaan, on se, että tämän datan perusteella kehitettävän algoritmin tavoitteena on tunnistaa kaatumiset, joissa henkilö on mahdollisesti satuttanut itseään. Tämä on tilanne, jota mittauksissakin halutaan simuloida. Jos käyttäjä nousisi heti ylös kaatumisen jälkeen, ei tällaisesta tapauksesta olisi tarvetta tehdä kaatumishälytystä.

Kaatumisdatan keräyksessä on käytetty 18 eri kaatumisskenaariota, jotka on listattu taulukossa 1.

Taulukko 1. Kaatumisskenaariot.

Skenaarioiden määrä	**Lähtöasento**	**Kaatumisen syy**	**Kaatumisen suunta**	**Loppuasento**
4	Seisten	Luhistuminen	Eteen, taakse, vasen, oikea	Mahalleen, selälleen, oikealle kyljelle, vasemmalle kyljelle
3	Kävely	Luhistuminen	Eteen, vasen, oikea	
2	Kävely	Liukastuminen	Eteen, taakse	
1	Kävely	Kompastuminen	Eteen	Loppuasento määrittyy kaatumisen suunnan perusteella
2	Hölkkäys	Liukastuminen	Eteen, taakse	
1	Hölkkäys	Kompastuminen	Eteenpäin	
3	Istuminen	Tuolin keikkaaminen	Taakse, vasen, oikea	
2	Makaaminen	Sängystä tippuminen vierien	Vasen, oikea	
Yhteensä: 18				

Kaatumisskenaarioiden keksimisessä käytettiin apuna Majd SALEH ym. tutkimusta, jossa dataa kerättiin sekä kaatumisista että normaaleista arjen aktiviteeteistä [16, s. 5]. Jokainen skenaario sisältää lähtöasennon ennen kaatumista, kaatumisen syyn ja kaatumisen suunnan, ja nämä ovat hyvin yksiselitteisiä. Kaatumisen syy pyritään aina simuloimaan aidosti, eli tavoitteena on, että tasapaino menetetään oikeasti. Esimerkiksi kompastumisessa ei vain tökätä jalkaa maahan ja hypätä eteenpäin, vaan jalan pitää oikeasti osua esteeseen, jonka johdosta tasapaino menetetään ja kaatuminen tapahtuu. Näin itse kaatumista ei tarvitse näytellä, vaan tasapainon menetyksen johdosta kehon refleksit toimivat automaattisesti lähes samoin tavoin kuin oikeassa kaatumistapauksessa.

3.2 Normaalien aktiviteettien datankeräys

Normaalien aktiviteettien datankeräyksen tarkoituksena on saada dataa sellaisista liikkeistä, joita käyttäjä saattaa suorittaa osana normaalia arkea. Näiden skenaarioiden keksimisessä pidetään myös mielessä mahdollinen kehitettävä algoritmi. Esimerkiksi, jos tiedetään, että algoritmi tulee todennäköisesti etsimään datasta korkeita kiihtyvyyspiikkejä, niin skenaarioiden kannattaa olla sellaisia, joissa tällaisia kiihtyvyyspiikkejä voisi syntyä. Näin voidaan alkaa tutkimaan, mitkä tekijät datassa erottavat nämä skenaariot oikeista kaatumisista, kun näitä korkeita kiihtyvyyspiikkejä esiintyy molemmissa tapauksissa. Tämä mielessä pitäen keksittiin 12 normaalia aktiviteettiä, jotka on listattu alla.

1. hyppiminen
2. x-hyppiminen
3. käden nostaminen
4. käden laskeminen
5. heiluttaminen
6. taputtaminen
7. käden lyöminen pöytään
8. tuolille istuminen
9. sohvalle istuminen
10. kävely
11. hölkkäys
12. käsien pesu, kuivaus ja ravistelu.

Kaikissa skenaarioissa otetaan huomioon tulevan algoritmin mahdolliset ehdot kaatumiselle. Jokainen skenaario suoritetaan siten, että se synnyttäisi korkean kiihtyvyyspiikin. Esimerkiksi skenaariossa 3 käsi nostetaan ylös todella nopeasti ja skenaariossa 8 sohvalle rojahdetaan, eikä vain istuta rauhallisesti. Liikkeiden jälkeen pidetään myös ajoittain taukoja, jotta niitä ei voisi luokitella normaaliksi aktiviteetiksi vain sillä perusteella, että kiihtyvyyspiikin jälkeen tapahtuu vielä useampia kiihtyvyyspiikkejä. Esimerkiksi skenaariossa 6 taputuksen jälkeen käsien liikkeet pysäytetään välillä hetkeksi, välillä taputetaan pidempään, ja välillä tehdään vain yksittäisiä taputuksia. Skenaariossa 8 taas tuolille jäädään välillä

hetkeksi istumaan ja välillä noustaan nopeammin ylös. Mittauksissa näitä liikkeitä toistetaan n. minuutin ajan, ja tänä aikana liikkeitä suoritetaan useita kertoja.

4 Algoritmin kehitys

4.1 Datan tutkiminen

Kun dataa on mitattu tarpeeksi, niin sen jälkeen voidaan aloittaa sen tutkiminen. Ennen kuin algoritmia voidaan alkaa kovin pitkälle kehittämään, niin datasta pitää löytää sellaisia piirteitä, jotka pätisivät mahdollisimman moniin kaatumisiin, ja samanaikaisesti mahdollisimman vähiin normaaleihin aktiviteetteihin. Näiden piirteiden löytämisessä käytetään kahta menetelmää.

Ensin perehdytään kerätystä datasta muodostettuihin kiihtyvyys- ja painesignaalien kuvaajiin. Perehtymällä näihin kuvaajiin saadaan hyvä yleiskuva siitä, miten nämä signaalit käyttäytyvät kaatumisissa ja normaaleissa aktiviteeteissa. Näin pystytään päättelemään, minkälaista tietoa koko datasta halutaan saada irti.

Kun on saatu hyvä yleiskuva siitä, miten signaalit käyttäytyvät eri tilanteissa, niin koko datasta voidaan alkaa muodostamaan hyödyllistä tilastotietoa. Näin saadaan tietoa esimerkiksi siitä, kuinka korkeita kiihtyvyyspiikkejä kaatumisissa tapahtuu, kuinka paljon paine kasvaa kiihtyvyyspiikkien jälkeen, kuinka alas kiihtyvyyssignaali tippuu ennen kaatumista yms. Tilastotiedoista saadaan esimerkiksi edellisessä lauseessa mainittujen tapahtumien minimi-, maksimi- ja keskiarvoja. Samankaltaisia tietoja kerätään myös normaaleista aktiviteeteista. Näin kiihtyvyys- ja painedatasta saadaan ylös arvoja, jotka pätevät kaatumisiin, mutta eivät normaaleihin aktiviteetteihin. Esimerkiksi tilastoista voidaan huomata, että 80 % kaatumisissa kiihtyvyyspiikit nousevat yli määrätyn raja-arvon ja paine kasvaa määrätyssä ajassa tietyn verran eikä tätä tapahdu missään normaaleissa aktiviteeteissa. Tätä tietoa voidaan sitten käyttää hyväksi, kun aletaan määrittelemään algoritmin ehtoja ja raja-arvoja.

Tilastotietojen keräämisessä käytetään Python-ohjelmointikieltä ja kehitysympäristönä toimii Anaconda Individual Edition (versio 4.11.0) [17]. Anaconda on avoimen lähdekoodin kehitysympäristö, joka sisältää lukuisia datatieteissä tarvittavia kirjastoja, kuten pandas ja numpy. Avoin lähdekoodi tarkoittaa, että kuka vaan voi tarkastella Anacondan lähdekoodia. Pandas ja numpy taas ovat Pythonin kirjastoja, jotka sisältävät datankäsittelyssä hyödyllisiä työkaluja, joiden avulla varsinkin isoja datamääriä on helpompi ja nopeampi käsitellä.

4.2 Havainnot datasta

Kaatumisia mitattiin 10 kappaletta jokaisesta skenaariosta, eli yhteensä 180 kappaletta. Normaaleja aktiviteetteja mitattiin 11 kappaletta per skenaario, eli 132 kappaletta. Normaaleja aktiviteetteja sisältävissä tiedostoissa on kuitenkin useampia liikkeitä per tiedosto, kun taas kaatumisia sisältävissä tiedostoissa on vain yksi kaatuminen per tiedosto.

Alaluvuissa 4.21–4.25 kerrotaan, minkälaisia piirteitä sekä kiihtyvyys-, että painesignaaleista löydettiin ja miten näitä voidaan käyttää algoritmissa hyödyksi erottamaan kaatuminen normaalista aktiviteetista. Näitä piirteitä löydettiin sekä datan tutkimisen alkuvaiheessa että algoritmin kehitysvaiheessa, kun tutkittiin ja kokeiltiin erilaisia tapoja erottaa kaatumiset normaaleista aktiviteeteista.

Kiihtyvyydestä löydettiin 5 piirrettä, jotka erottavat kaatumisen normaalista aktiviteetista. Näitä ovat korkea kiihtyvyyspiikki, aktiviteetin kesto, korkeiden kiihtyvyyspiikkien määrä aktiviteetin aikana, matalien kiihtyvyyspiikkien määrä aktiviteetin aikana ja korkeiden kiihtyvyyspiikkien määrä ennen ja jälkeen aktiviteetin. Ilmanpainesignaalista löydettiin yksi piirre, joka on ilmanpaineen muutos aktiviteetin jälkeen. Tässä kontekstissa aktiviteetilla tarkoitetaan sellaista tapahtumaa kiihtyvyyssignaalissa, josta on löydetty tarpeeksi korkea kiihtyvyyspiikki ja jossa aktiviteetin alku ja loppu on pystytty määrittelemään.

Alaluvuissa 4.21–4.25 kuvaillut arvot on määritelty siten, että kun kiihtyvyysdatasta löydettiin tarpeeksi korkea kiihtyvyyspiikki, niin tästä ajanhetkestä määriteltiin aktiviteetin alku ja loppu. Jos puhutaan arvoista aktiviteetin aikana, niin

silloin tarkoitetaan näiden kahden ajan väliltä määriteltyjä arvoja. Jos puhutaan arvoista ennen tai jälkeen aktiviteetin, niin silloin tarkoitetaan näiden aikojen ulkopuolelta määriteltyjä arvoja.

Alkuvaiheessa huomattiin kuitenkin, että jos kaatumisen lähtöasentona on istuminen tai makaaminen, niin silloin ilmanpaineen kasvu kaatumisen aikana on keskimäärin hyvin pieni. Koska ilmanpaineen kasvun todettiin jo varhain olevan hyvä piirre erottamaan kaatumiset normaaleista aktiviteeteista, jätettiin edellä mainitut tapaukset pois tarkemmista tutkimuksista. Tästä johtuen seuraavissa alaluvuissa ilmoitetut tulokset eivät sisällä näitä kaatumisia.

Seuraavat havainnot sisältävät 130 kaatumista ja 260 normaalia aktiviteettia. Koska aktiviteetin alku ja loppu on määritelty siten, että niillä voitaisiin mahdollisimman tarkasti arvioida kaatumisen kesto, niin jo tämän määritelmän takia tulokset eivät sisällä jokaista yksittäistä normaalia aktiviteettia. Tämä johtuu siitä, että aktiviteetin alkua ja loppua tarkasteltiin vain tietyltä etäisyydeltä löydetyn kiihtyvyyspiikin ympäriltä, ja jos näitä ei pystytty määrittelemään, niin datassa siirryttiin vain eteenpäin. Algoritmissa aktiviteetti voitaisiin määritellä normaaliksi aktiviteetiksi jo sillä perusteella, että aktiviteetin kestoa ei ole pystytty määrittelemään.

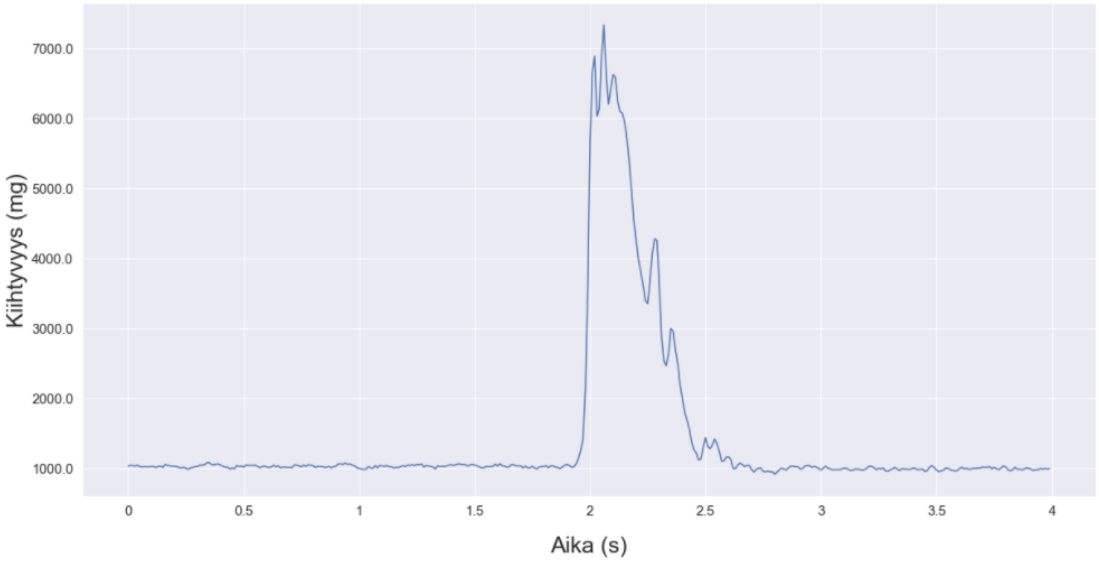
4.2.1 Aktiviteetin kesto

Jotta aktiviteetin kestoa voidaan mitata, pitää määritellä aktiviteetin alku ja loppu. Aktiviteetin alku määritellään siten, että ensimmäisestä löydetystä korkeasta kiihtyvyyspiikistä lasketaan arvoja ajassa taaksepäin kahden sekunnin ajalta. Kun on löydetty tarpeeksi monta peräkkäistä matalaa arvoa, niin näistä viimeisen arvon kohta datassa, eli sen, joka on lähinnä löydettyä korkeaa kiihtyvyyspiikkiä, määritellään aktiviteetin aluksi.

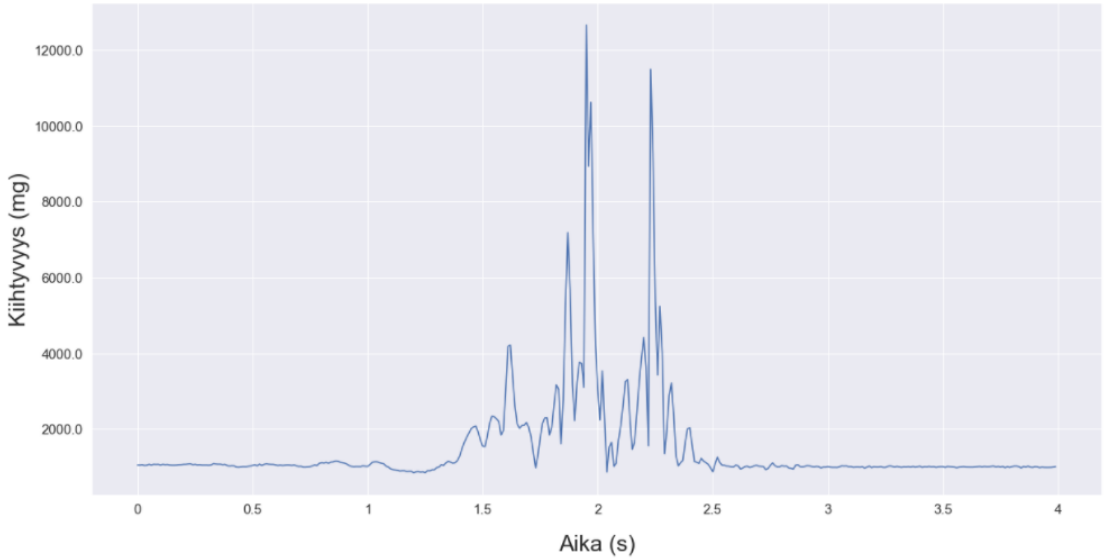
Aktiviteetin loppu määritellään siten, että samasta kiihtyvyyspiikistä lasketaan arvoja ajassa eteenpäin kahden sekunnin ajalta. Kun on löydetty tarpeeksi monta peräkkäistä matalaa arvoa, niin näistä ensimmäisen arvon kohta datassa

määritellään aktiviteetin lopuksi. Aktiviteetin kesto on yksinkertaisesti aktiviteetin alun ja lopun välinen etäisyys datassa.

Näillä edellä mainituilla menetelmillä pystyttiin luotettavimmin määrittelemään kaatumisen kesto. Kuvissa 10 ja 11 ovat esimerkit, miten aktiviteetin kestolla voidaan erottaa normaali aktiviteetti kaatumisesta.



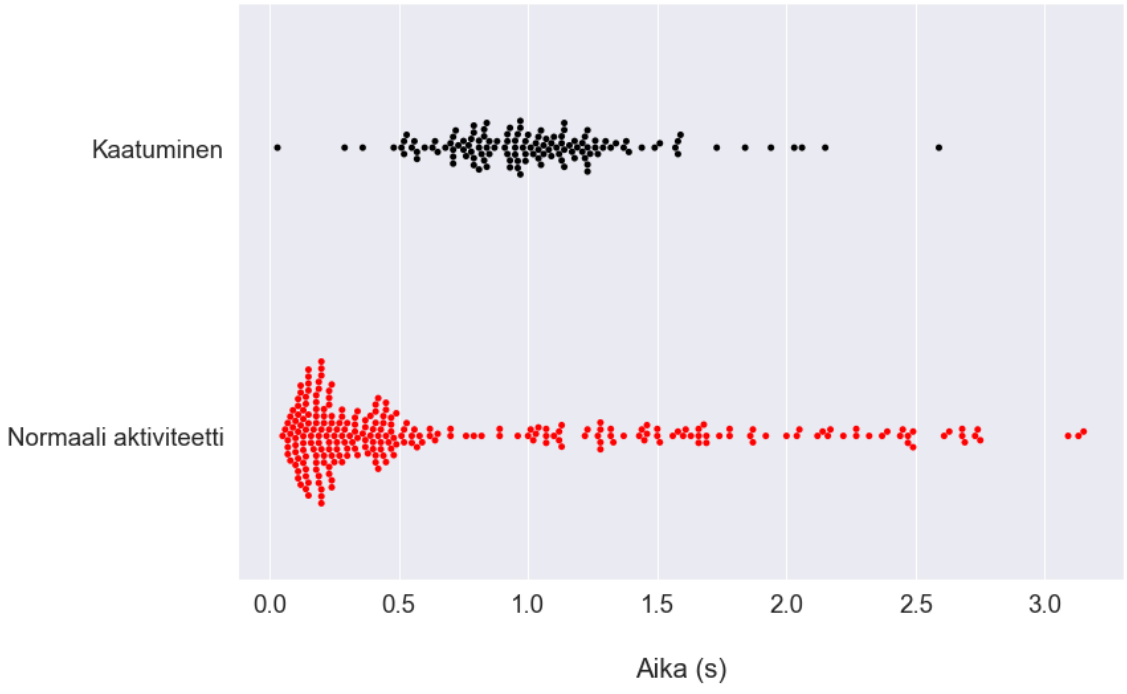
Kuva 10. Kiihtyvyyssignaali. Käden nopea nostaminen.



Kuva 11. Kiihtyvyyssignaali. Kaatuminen eteenpäin, lähtöasentona paikallaan seisominen.

Kuten kuvasta 10 voi huomata, käden nopea nostaminen voi aiheuttaa kiihtyvyyspiikin, joka voi olla yhtä korkea tai korkeampi, kuin mitä syntyisi kaatumisessa. Tällainen kiihtyvyyspiikki voidaan saavuttaa käden nostamisella vain, jos liike on hyvin nopea, mikä johtaa myös siihen, että koko tapahtuma on hyvin lyhyt, esimerkiksi kuvan 11 kaatuminen kestää noin sekunnin, kun taas kuvan 10 käden nostaminen noin puoli sekuntia. Mittaamalla aktiviteetin kestoa voidaan tällaiset tapahtumat määritellä normaaleiksi aktiviteeteiksi.

Kuvassa 12 esitellään, miten aktiviteetin kestot jakautuvat kaatumisissa ja normaaleissa aktiviteeteissa. Jokainen yksittäinen tapaus on mallinnettu yhtenä pisteenä kuvaan: kaatumiset ovat mustia ja normaalit aktiviteetit punaisia.



Kuva 12. Kaatumisten ja normaalien aktiviteettien kestot.

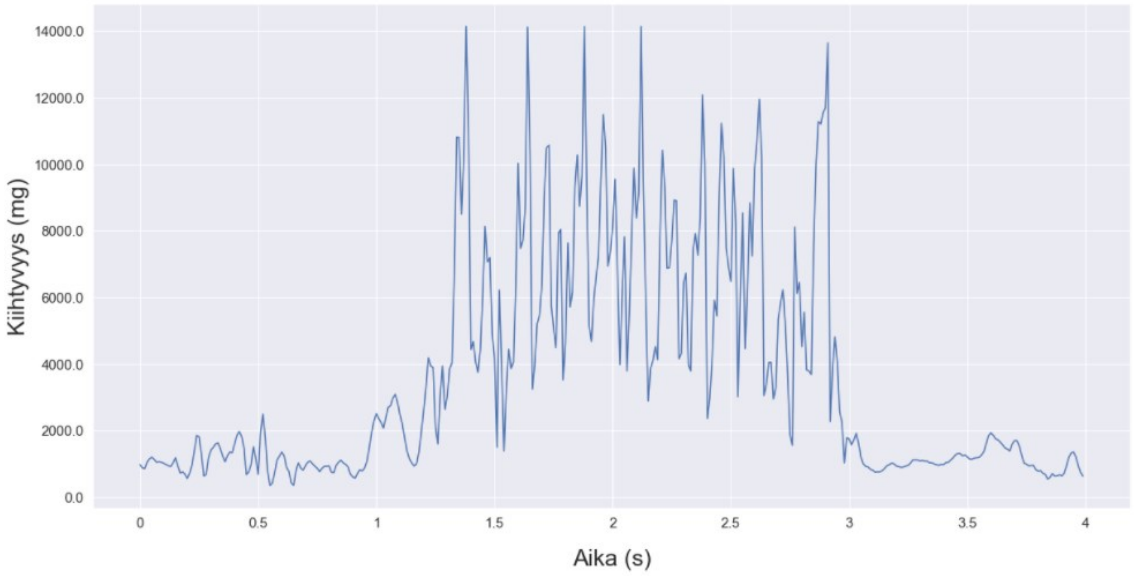
Kuvasta 12 voi huomata, että kaatumisissa aktiviteetin kestojen jakauma on huomattavasti kapeampi kuin normaaleissa aktiviteeteissa. Jos algoritmissa määriteltäisiin, että aktiviteetin pitäisi olla vähintään 0,5 sekuntia pitkä, jotta se voitaisiin luokitella kaatumiseksi, niin jo tällä perusteella voitaisiin yli puolet mitatuista normaaleista aktiviteeteista luokitella oikein, eli ei kaatumisiksi.

Kuvasta 12 näkyy, että muutaman kaatumisen kesto on ollut todella lyhyt, eli näissä tapauksissa itse aktiviteetin määritelmä ei ole sopinut kyseiseen kaatumistapaukseen. Esimerkiksi jos näissä tapauksissa on ensimmäisen korkean kiihtyvyyspiikin jälkeen tullut poikkeuksellisen pitkä aika, jossa kaikki kiihtyvyysarvot ovat olleet matalia, on kaatuminen virheellisesti määritelty loppuvan huomattavasti aikaisemmin, kuin se on oikeasti loppunut.

Muutamien kaatumisten mitattu kesto on taas ollut hieman todellista pidempi. Esimerkiksi tarpeeksi tiheätahtinen hölkkäys juuri ennen kaatumista yhdistettynä liukastumiseen tai kompastumiseen voi aiheuttaa sen, että juuri ennen kaatumista datasta ei löydykään tarpeeksi montaa matalaa kiihtyvyysarvoa. Tämä johtaa siihen, että kaatumisen alku määritellän virheellisesti todellista aikaisemmaksi, mikä kasvattaa kaatumisen mitattua kestoa.

4.2.2 Korkeiden kiihtyvyyspiikkien määrä aktiviteetin aikana

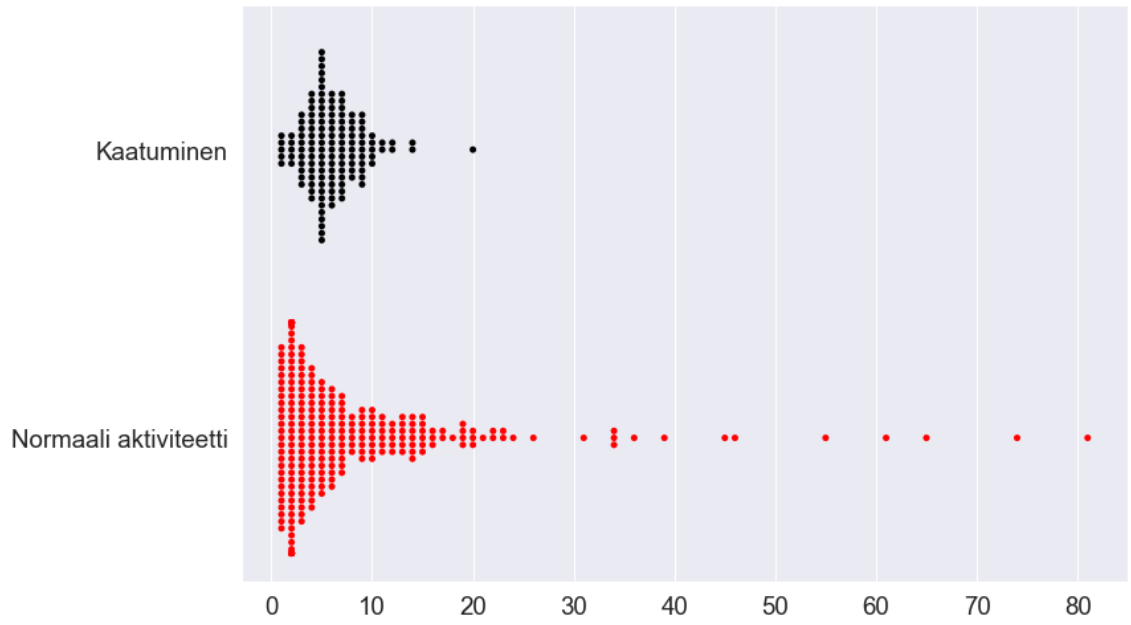
Laskemalla korkeiden kiihtyvyyspiikkien määrää aktiviteetin aikana voidaan määritellä, kuinka paljon rajuja liikehdintää on tapahtunut kyseisessä aktiviteetissä. Korkeiden kiihtyvyyspiikkien määrä aktiviteetin aikana on yksinkertaisesti yli tietyn raja-arvon ylittäneiden kiihtyvyysarvojen määrä aktiviteetin alun ja lopun välissä. Kaatumisten aikana tällaisia arvoja syntyy yleensä vähintään 1, mutta usein enemmän. Joissakin normaaleissa aktiviteeteissa näitä arvoja syntyy kuitenkin enemmän kuin useimmissa kaatumisissa. Kuvassa 13 on esimerkki tällaisesta aktiviteetista.



Kuva 13. Kiihtyvyyssignaali. Käden ravistelu käsien pesun jälkeen.

Kuten kuvasta 13 voi nähdä, niin käden ravistelu aiheutti huomattavasti enemmän korkeita kiihtyvyyspiikkejä kuin esimerkiksi kuvassa 11 esitetyssä kaatumistapauksessa, jonka perusteella tämä käden ravistelu voidaan määritellä normaaliksi aktiviteetiksi.

Kuvassa 14 esitellään, miten korkeiden kiihtyvyyspiikkien määrä jakautuu kaatumisissa ja normaaleissa aktiviteeteissa.

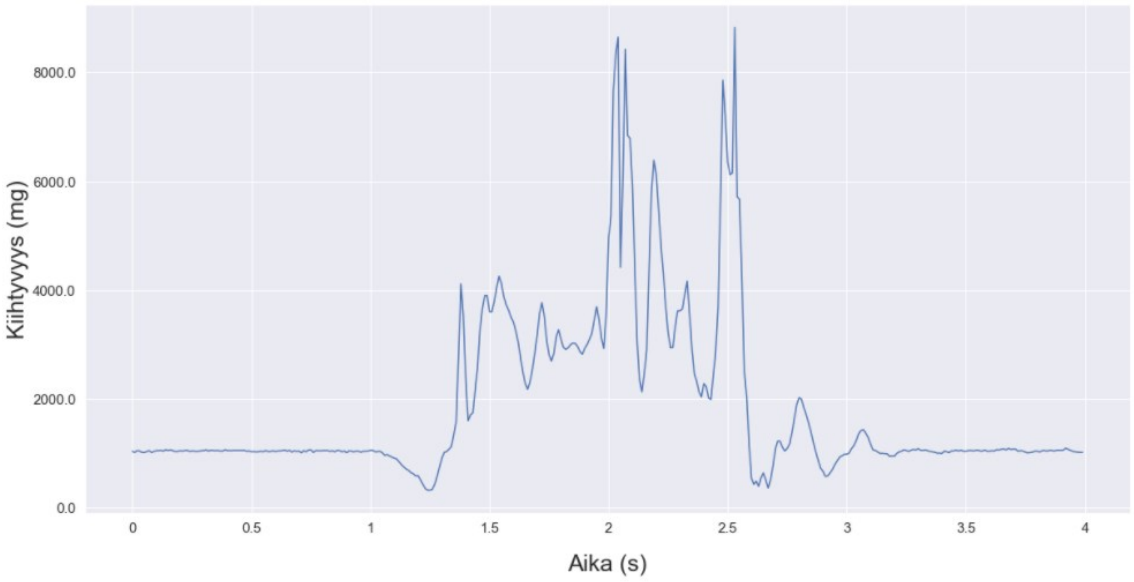


Kuva 14. Korkeiden kiihtyvyyspiikkien määrät kaatumisten ja normaalien aktiviteettien aikana.

Kuvasta 14 näkyy, että lähes kaikissa kaatumisissa korkeiden kiihtyvyyspiikkien määrät ovat maksimissaan 10, kun taas normaaleissa aktiviteeteissa on useita kymmeniä tapauksia, joissa korkeiden kiihtyvyyspiikkien määrä on yli 10. Yli 10 korkeaa kiihtyvyyspiikkiä sisältäviä kaatumisia taas on vain 7. Eli vaikka sekä kaatumisissa että normaaleissa aktiviteeteissa suurin osa tapauksista jää 1 ja 10 välille, voitaisiin algoritmissa silti sulkea monia normaaleja aktiviteetteja pois asettamalla yläraja korkeille kiihtyvyyspiikeille.

4.2.3 Matalien kiihtyvyyspiikkien määrä aktiviteetin aikana

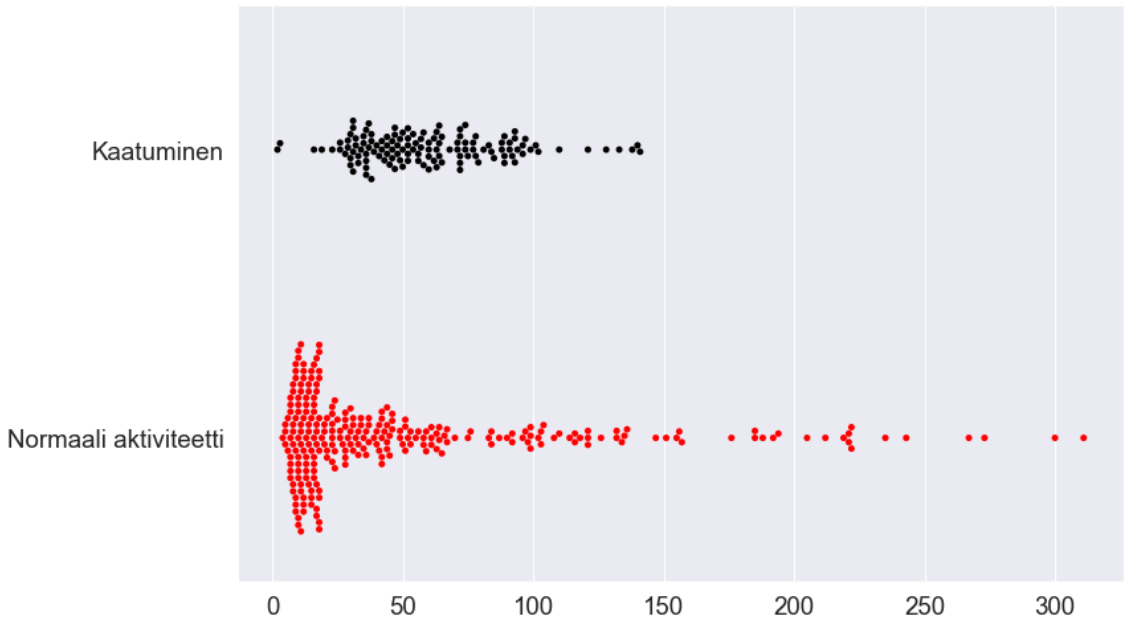
Laskemalla matalien kiihtyvyyspiikkien määrää aktiviteetin aikana voidaan määritellä, kuinka paljon liikettä on yleisesti tapahtunut kyseisen aktiviteetin aikana. Tämä määritellään laskemalla tietyn raja-arvon ylittäneiden kiihtyvyysarvojen määrä aktiviteetin alun ja lopun välillä. Kuvassa 15 näkyy matalien kiihtyvyyspiikkien määrä X-hyppelyssä.



Kuva 15. Kiihtyvyyssignaali. X-hyppely.

Kuvassa 15 esitellyssä X-hyppelyssä lähes kaikki arvot ovat suhteellisen korkeita, jonka perusteella se voitaisiin määritellä normaaliksi aktiviteetiksi, vaikka aktiviteetin pituus onkin samankaltainen kuin kaatumisissa, eikä korkeita kiihtyvyyspiikkejäkään ole montaa. Tämä on myös ihan silmin nähtävissä, jos kuvaa 15 vertaa esimerkiksi kuvan 11 kaatumiseen. Kuvassa 15 lähes jokainen arvo on yli 2000 mg korkea, kun taas kuvan 11 kaatumisen aikana kiihtyvyysarvot laskevat usein alle 2000 mg.

Kuvassa 16 esitellään, miten matalien kiihtyvyyspiikkien määrä jakautuu kaatumisissa ja normaaleissa aktiviteeteissa.



Kuva 16. Matalien kiihtyvyyspiikkien määrät kaatumisten ja normaalien aktiviteettien aikana.

Kuvasta 16 näkyy, kuinka normaaleissa aktiviteeteissa matalien kiihtyvyyspiikkien määrät jäävät usein todella alhaiseksi. Normaaleissa aktiviteeteissa on myös yli 20 tapausta, joissa matalia kiihtyvyyspiikkejä on enemmän kuin yhdessäkään kaatumisessa. Algoritmiin voitaisiin siis asettaa matalien kiihtyvyyspiikkien määrälle sekä ala- että ylärajat.

Koska aktiviteetin kesto oli muutamassa kaatumisessa määritelty virheellisesti todellista lyhyemmäksi, niin tästä johtuen myös muutamassa kaatumisessa mitattujen matalien kiihtyvyyspiikkien määrät ovat todellista alhaisempia.

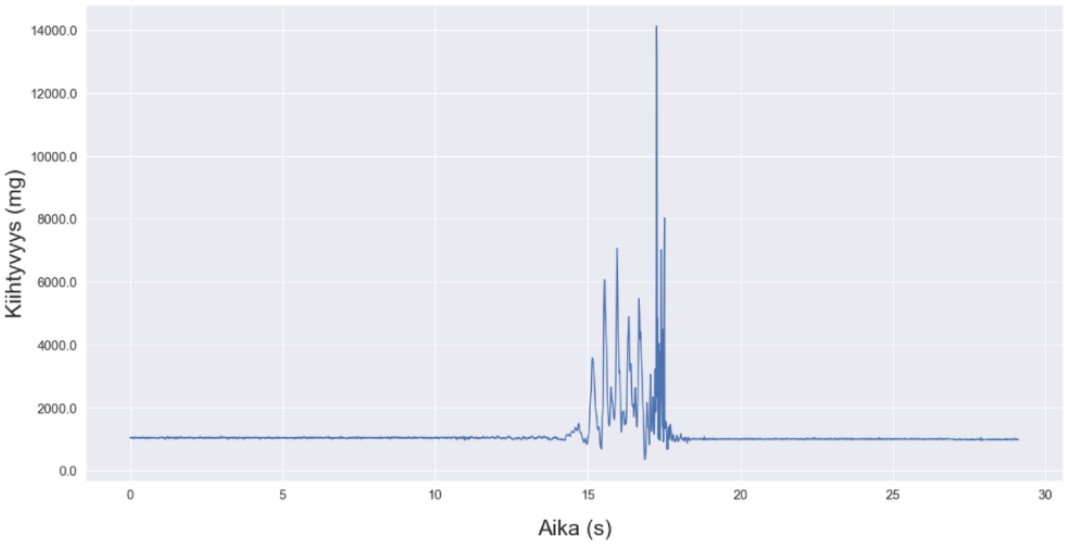
4.2.4 Korkeiden kiihtyvyyspiikkien määrä ennen ja jälkeen aktiviteetin

Viimeisenä kaatumisten ja normaalien aktiviteettien erottavana piirteenä kiihtyvyyssignaalissa on korkeiden kiihtyvyyspiikkien esiintyminen ennen ja jälkeen aktiviteetin. Nämä eivät ole niinkään itse aktiviteettiin liittyviä piirteitä vaan varmistuksia, että aktiviteetti on yksittäinen tapahtuma, eikä osa jotakin liikesarjaa. Algoritmissa voitaisiin siis määrittää, että jos juuri ennen tai jälkeen aktiviteetin

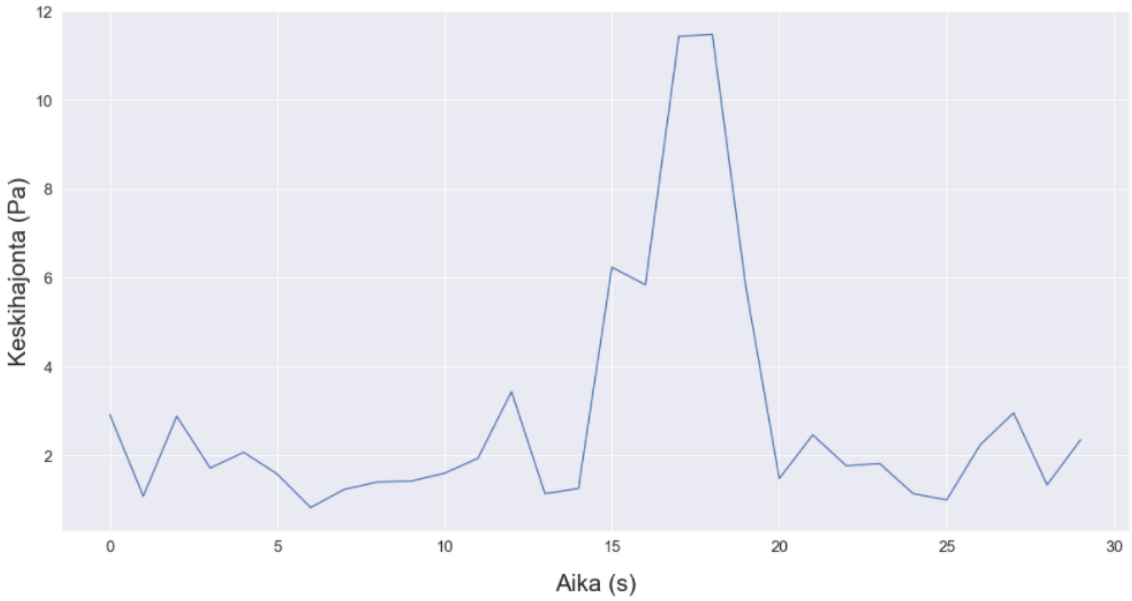
on esiintynyt muita korkeita kiihtyvyyspiikkejä, niin silloin aktiviteetti voidaan määritellä normaaliksi aktiviteetiksi.

4.2.5 Ilmanpaineen kasvu

Koska kaatuessa henkilö, mukaan lukien ranne, jossa kello sijaitsee, siirtyy aina alemmalle tasolle, niin tällöin myös kellon mittaama ilmanpaineen pitäisi kasvaa. Ilmanpaineen kasvua mitattiin ottamalla mediaani 2 s ennen aktiviteetin alkua sisältävistä arvoista, sekä 2–4 s aktiviteetin lopun jälkeen. Koska ilmanpaineen täydellinen mittaaminen näin pienessä skaalassa on mahdotonta, on datassa paljon vaihtelua mittauspisteiden välillä. Tämän takia mediaani on keskiarvoa parempi mittari, koska datassa sijaitsevat keskiarvosta huomattavasti poikkeavat arvot eivät vaikuta mediaaniin yhtä paljon kuin keskiarvoon. Syy siihen, että aktiviteetin lopun jälkeen odotetaan vielä 2 s ennen mediaanin mittaamista, on se, että kaikenlaiset ranteen liikkeet aiheuttavat vielä lisää vaihtelua mitatussa datassa. Mitä enemmän vaihtelua mitatussa datassa on, niin sitä vähemmän voidaan olla varmoja mediaanin tarkkuudesta. Kuvista 17 ja 18 näkyy, kuinka liike vaikuttaa mitatun ilmanpaineen keskihajontaan. Kuvassa 17 näkyy kaatumisen kiihtyvyysdataa, ja kuvassa 18 näkyy ilmanpainedatan keskihajontaa saman kaatumisen aikana.



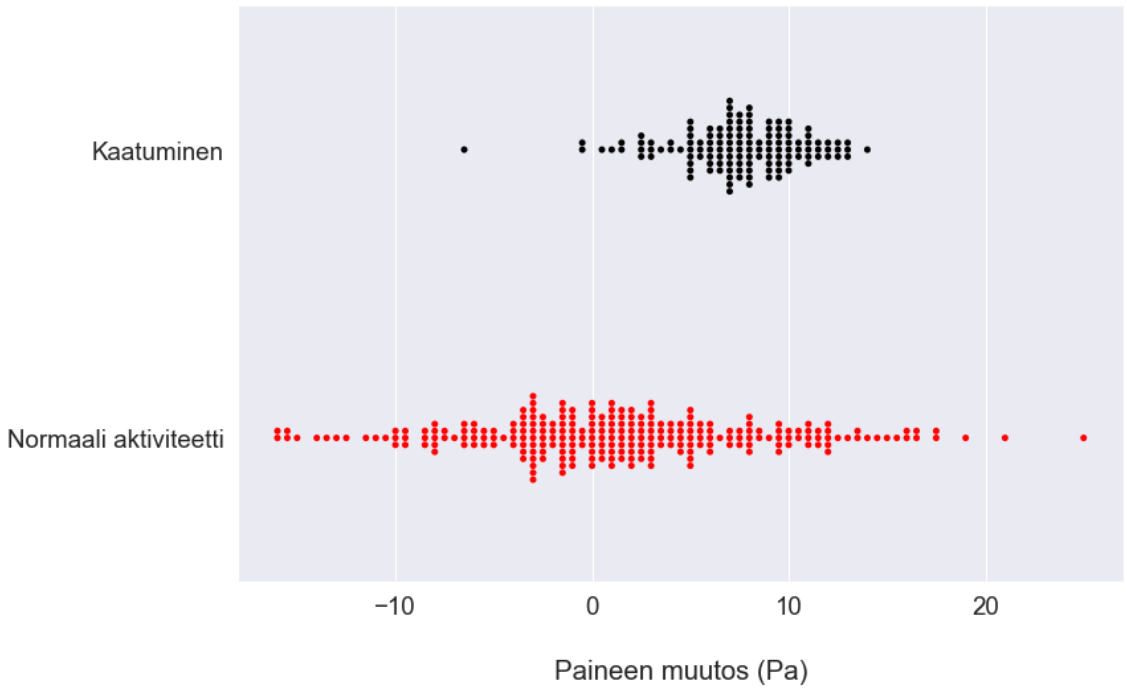
Kuva 17. Kiihtyvyyssignaali. Kaatuminen eteenpäin, lähtöasentona hölkkäys, kaatumisen syynä kompastuminen.



Kuva 18. Ilmanpainesignaalin keskihajonta. Kaatuminen eteenpäin, lähtöasentona hölkkäys, kaatumisen syynä kompastuminen.

Kuvassa 18 on mallinnettu keskihajonta jokaisen mitatun sekunnin ilmanpainearvoista. Vertaamalla kuvaa 18 kuvaan 17 voidaan huomata, että kun hölkkäys alkaa noin 15 sekunnin kohdalla, niin myös ilmanpaineen arvoissa alkaa näkyä suurta vaihtelua, joka on suurimmillaan kaatumisen aikana. Tämä näkyy keskihajonnan kasvuna, joka kuitenkin laskee taas, kun henkilö on hetken aikaa maannut maassa.

Kun mediaanit ovat määritelty, niin aktiviteetin jälkeisestä mediaanista erotetaan aktiviteettia edeltävä mediaani, josta saadaan ilmanpaineen muutos. Kuvassa 19 esitellään, miten ilmanpaineen muutos jakautuu kaatumisissa ja normaaleissa aktiviteeteissa.



Kuva 19. Ilmanpaineen muutos normaalien aktiviteettien jälkeen.

Kuvassa 19 näkyy, kuinka ilmanpaine kasvoi keskimäärin useammin kaatumisissa kuin normaaleissa aktiviteeteissa. Huomattavaa kuitenkin on, että ilmanpaineen mittauksen epätarkkuudesta johtuen, kaikissa kaatumisissa ilmanpaineen kasvua ei tunnistettu, tai tunnistettu kasvu jäi hyvin pieneksi. Ilmanpaine kasvoi merkittävästi myös osissa normaaleissa aktiviteeteissa, vaikka läheskään kaikissa näissä koehenkilön käsi ei olisi edes siirtynyt alemmalle tasolle.

4.3 Algoritmin kirjoittaminen

Vaikka lopullinen Navigil 580 -rannekellossa toimiva algoritmi tullaan kirjoittamaan C-ohjelmointikielellä, käytettiin datan tutkimisen lisäksi myös algoritmin kehityksessä Pythonia, koska se on minulle huomattavasti tutumpi kieli, mikä nopeuttaa kehitysvaihetta. Algoritmin kehitysvaiheessa ohjelmoitiin kaatumisentunnistukseen tarkoitetun algoritmin lisäksi myös funktioita, joiden avulla saatiin tietoon kehitetyn algoritmin luokittelutarkkuus sekä muita hyödyllisiä tietoja, esimerkiksi jos algoritmi ei tunnistanut kaatumista, niin näytölle tulostettiin, mitä ehtoa algoritmista kyseinen kaatuminen ei läpäissyt, ja tämän parametrin arvo. Al-

goritmiin ajettiin myös kaatumisia sisältävien tiedostojen lisäksi normaaleja aktiviteetteja sisältäviä tiedostoja ja algoritmin luokittelutarkkuutta näistä voitiin mitata kuvassa 20 esitetyllä virhematriisilla.

		Actual Class	
		Positive (**P**)	Negative (**N**)
Predicted Class	Positive (**P**)	True Positive (**TP**)	False Positive (**FP**)
	Negative (**N**)	False Negative (**FN**)	True Negative (**TN**)

Kuva 20. Virhematriisi [18].

Virhematriisi sisältää 4 arvoa, jotka ovat oikea positiivinen (TP), väärä positiivinen (FP), väärä negatiivinen (FN) ja oikea negatiivinen (TN). Tässä kontekstissa termit tarkoittavat seuraavaa:

1. TP = Kaatuminen luokiteltu oikein, eli kaatumiseksi
2. FP = Normaali aktiviteetti luokiteltu väärin, eli kaatumiseksi
3. FN = Kaatuminen luokiteltu väärin, eli normaaliksi aktiviteetiksi
4. TN = Normaali aktiviteetti luokiteltu oikein, eli normaaliksi aktiviteetiksi

Algoritmin kehitys oli iteroiva prosessi. Alkuun tiedettiin alustavia piirteitä kaatumisista sekä normaaleista aktiviteeteistä ja näitä pyrittiin ohjelmoimaan algoritmiin konkreettisiksi säännöiksi. Tämän jälkeen näitä sääntöjä hienosäädettiin luokittelutarkkuuden mukaisesti. Tavoitteena oli pitää väärät positiiviset nollassa ja saada mahdollisimman monta oikeaa positiivista. Ohjelmoitujen sääntöjen piti kuitenkin olla aina perusteltavissa, koska väärien oletuksien perusteella ohjelmoidut säännöt saattavat mitatun datan perusteella antaa hyvän lopputuloksen, mutta oikeassa elämässä huonon.

Kun päästiin siihen pisteeseen, että algoritmin luokittelutarkkuutta ei saatu enää merkittävästi kasvatettua, alettiin kehitetyn algoritmin pohjalta ohjelmoimaan lopullista algoritmia, jonka rakenne muistuttaa enemmän Navigil 580 -rannekellon

sulautetussa järjestelmässä pyörivän C-koodin rakennetta. Tätä algoritmia ei enää käytetty luokittelutarkkuuden parantamiseen, vaan sen tarkoituksena oli muokata jo toimivaksi todettu algoritmi vastaamaan sulautetun järjestelmän vaatimuksia. Näin algoritmi on huomattavasti helpompi kääntää lopulta kellossa käytettäväksi C-koodiksi. Esimerkkejä Navigil 580 -rannekellon sulautetun järjestelmän vaatimuksista on rajallinen muisti ja algoritmiin tulevan datan rakenne.

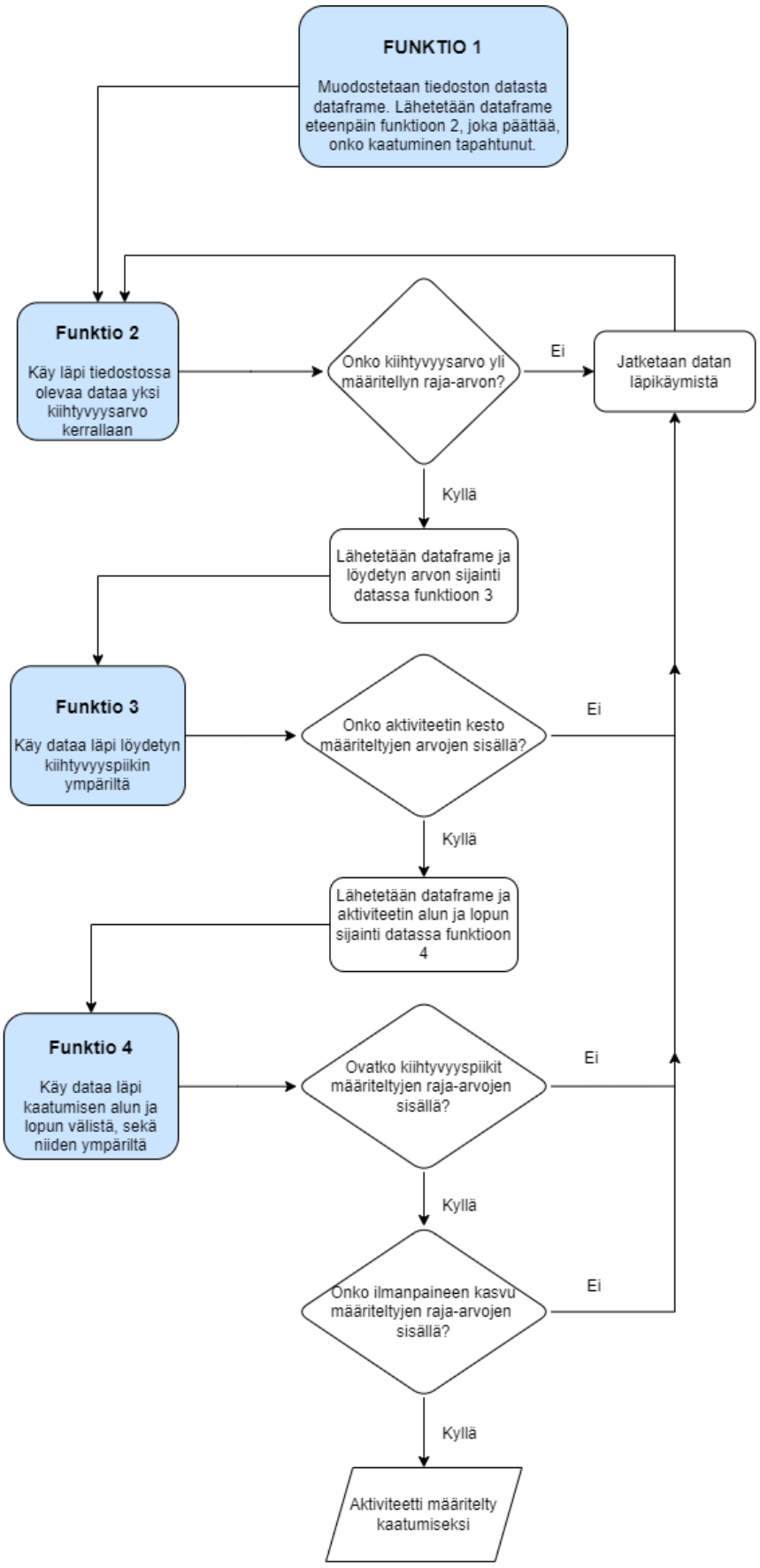
Edellä mainittujen rakenteellisten muutoksien lisäksi muita muutoksia lopulliseen algoritmiin oli vain Pythonille ominaisten koodinpätkien muuttaminen sellaiseen muotoon, että ne voidaan koodata myös C-kielellä. Nämä muutokset koskivat esimerkiksi tapoja, miten arvoja käydään läpi jostakin listasta. Esimerkiksi nämä kaikki arvojen läpikäymiset muutettiin yksinkertaisiksi for-loopeiksi, joiden toimintaperiaate on samankaltainen lähes kaikissa ohjelmointikielissä. Lopullisen algoritmin koodauksessa pyrittiin siis käyttämään yleisiä kaikissa ohjelmointikielissä käytettäviä metodeja, jotta algoritmin kääntäminen C-kielelle olisi mahdollisimman yksinkertaista.

4.4 Kehitetyn algoritmin rakenne

Koska tässä insinöörityössä kehitetyn algoritmin koodia ei julkaista, esitellään kehitetyn algoritmin rakennetta sanallisesti vuokaavioita apuna käyttäen. Alaluvussa 4.4.1 kuvaillaan alustavan algoritmin rakennetta ja alaluvussa 4.4.2 kuvaillaan tästä alustavasta algoritmista jalostetun lopullisen algoritmin rakennetta.

4.4.1 Alustava algoritmi

Algoritmin kehityksen alkuvaiheessa algoritmin rakenne oli hyvin yksinkertainen, ja sen tarkoituksena oli vain testata erilaisten ehtojen ja parametrien vaikutusta tunnistustarkkuuteen. Kuvassa 21 esitellään tämän alustavan algoritmin toimintaperiaatetta vuokaaviossa.



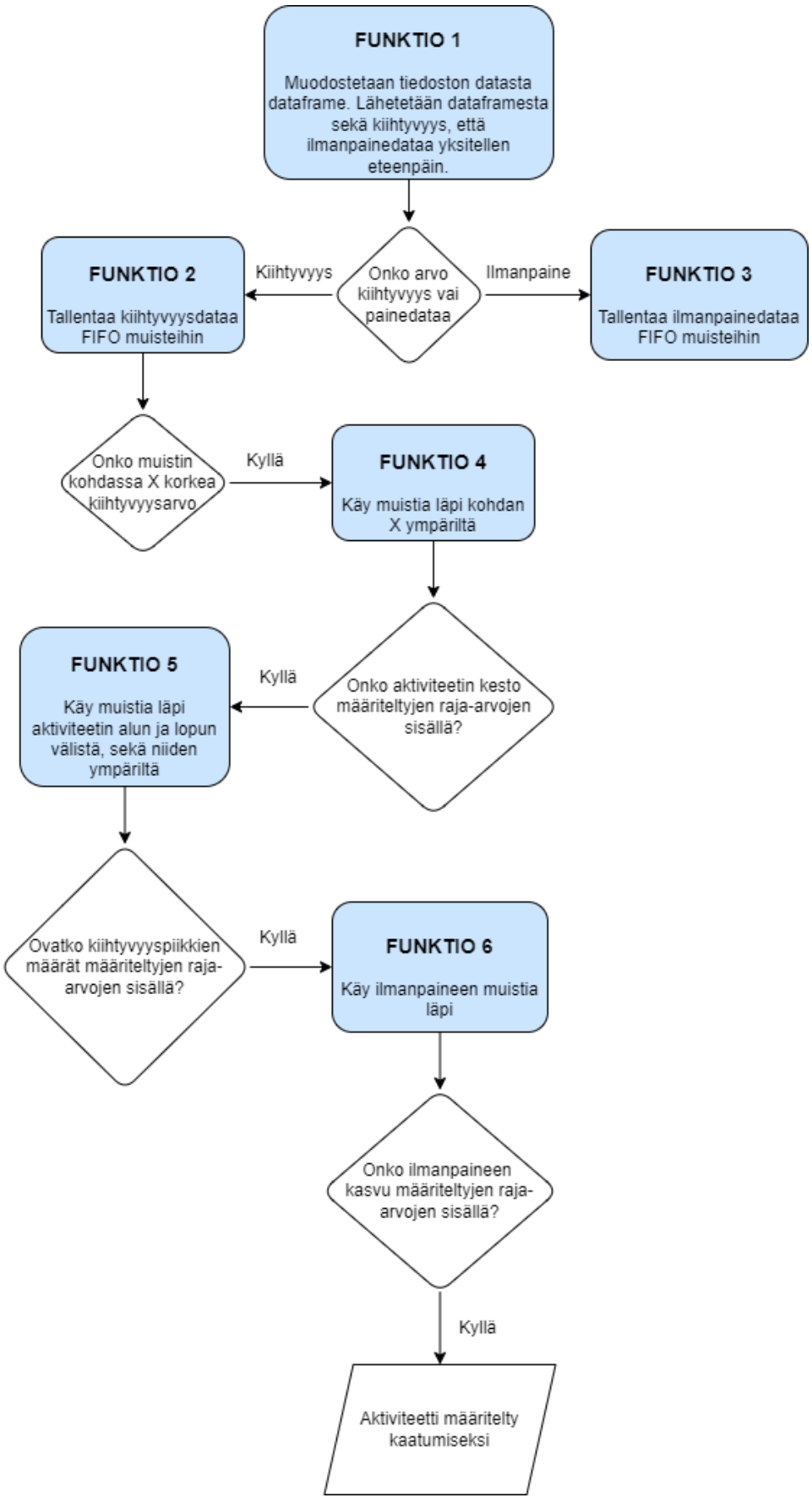
Kuva 21. Vuokaavio alustavan algoritmin rakenteesta.

Kuten kuvasta 21 voi huomata, on algoritmin rakenne suhteellisen yksinkertainen. Algoritmi tarkastelee luvussa 4.2 kuvailtuja piirteitä sekä kaatumisia että normaaleja aktiviteetteja sisältävistä tiedostoista. Näiden tiedostojen datasta muodostetaan dataframe, joka on Pandas-kirjaston sisältämä datatyyppi. Dataframe koostuu kolumneista ja riveistä, 1 rivi sisältää jokaista kolumnia vastaavan arvon. Kolumni taas määrittää, onko kyse kiihtyvyys- vai painearvosta. Funktio 2 käy näitä rivejä läpi keskittyen vain kiihtyvyysarvoihin. Tästä eteenpäin algoritmi tarkastaa yksitellen, onko tietyssä kohtaa kiihtyvyysdataa sellaisia piirteitä, jotka ovat määritelty tapahtuvan vain kaatumisissa. Jos kaikki kiihtyvyyden piirteet vastaavat kaatumista, niin lopuksi tarkastetaan, onko ilmanpaine kasvanut yli valmiiksi määritellyn raja-arvon. Jos on, niin silloin ilmoitetaan, että tästä tiedostosta on löytynyt kaatuminen.

4.4.2 Lopullinen algoritmi

Kun oltiin jo suhteellisen varmoja siitä, minkälaisilla ehdoilla ja parametreilla voitaisiin saavuttaa paras tunnistustarkkuus, niin tässä vaiheessa algoritmin rakenne katselmoitiin, jotta se voitaisiin suoraan kääntää C-kielelle ja ottaa käyttöön Navigil 580 -rannekellossa. Katselmoinnissa pohdittiin, miten dataa tulisi käsitellä, kuinka paljon dataa voidaan pitää kerralla muistissa, ja muita asioita, joita pitää ottaa huomioon lopullista algoritmia kehittäessä. Näiden katselmointien pohjalta kirjoitettiin lopullinen algoritmi, jonka rakenne hyväksytettiin työn tilaajalta.

Lopullisen algoritmin toimintaperiaate on hyvin samankaltainen alustavan algoritmin kanssa, piirteitä kiihtyvyydestä ja ilmanpaineen muutoksesta tarkastellaan samassa järjestyksessä. Lopullisen algoritmin rakenne on kuitenkin hyvin erilainen ja alkuperäistä monimutkaisempi, mikä johtuu Navigil 580 -rannekellon sulautetun järjestelmän vaatimuksista koodille. Kuvassa 22 esitellään lopullisen algoritmin toimintaperiaatetta vuokaaviossa.



Kuva 22. Vuokaavio lopullisen algoritmin rakenteesta.

Kuten kuvasta 22 voi huomata, on lopullisen algoritmin rakenne erilainen verrattuna kuvan 21 alkuperäiseen algoritmiin. Seuraavaksi kerrotaan tarkemmin, miten data liikkuu funktioiden väleillä tietyissä tapauksissa.

Ensimmäisenä erona alkuperäiseen algoritmiin on se, miten algoritmiin tulevaa dataa käsitellään. Rannekellossa dataa ei tule isoina tiedostoina, vaan jatkuvana yksittäisten arvojen virtana kiihtyvyys- ja painesensoreista. Tästä johtuen tätä tilannetta simuloitiin funktiossa 1 ja lähettämällä ne yksittäisinä arvoina eteenpäin sen sijaan, että koko mittausdata lähetetään algoritmiin, kuten kuvassa 21 on näytetty.

Sisään tulevaa kiihtyvyys- ja ilmanpainedataa tallennetaan omiin FIFO-muisteihin. FIFO on lyhenne sanoista first in, first out, ja tässä kontekstissa tällä tarkoitetaan sitä, että kun muistiin loppuun lisätään uusi arvo, niin muistin toisesta päästä, eli alusta otetaan viimeinen arvo pois. Tämä on se periaate, jolla dataa tallennetaan muistiin Navigil 580 -rannekellon sulautetun järjestelmän C-koodissa. Tästä johtuen muistin koko määrittää, kuinka monen kiihtyvyys- ja painearvon perusteella voidaan päätellä, onko aktiviteetti kaatuminen vai ei. Tämä ei kuitenkaan aiheuttanut ongelmia, koska alkuperäinen algoritmi tarvitsi näitä arvoja vain muutaman sekunnin ajalta, ja tältä ajalta niitä tullaan pystymään pitämään muistissa myös C-koodissa.

Seuraavana on esimerkki algoritmin toiminnasta: Kiihtyvyyden FIFO-muistin koko on 400 arvoa, ja muistin kohdasta 200 löytyy korkea kiihtyvyyspiikki. Tällöin tiedetään, että jos tarkastellaan kiihtyvyysarvoja muistin kohdista 0–200, ovat nämä arvot syntyneet 2 s ennen tätä korkeaa kiihtyvyyspiikkiä. Jos taas tarkastellaan arvoja 200–400, ovat nämä arvot syntyneet 2 s kiihtyvyyspiikin jälkeen. Funktio 2 tekee siis korkean kiihtyvyyspiikin tarkastuksen aina samassa kohtaa muistia, jotta tiedetään, mitkä kiihtyvyysarvot ovat syntyneet ennen ja mitkä jälkeen aktiviteetin.

Jos funktio 4 pystyi määrittelemään aktiviteetin lopun, niin siitä lähetettiin tieto funktioon 2. Tämän toiminnon tarkoituksena oli rajoittaa aktiviteetin määrittele-

minen vain yhteen kertaan per löydetty korkea kiihtyvyyspiikki. Ilman tätä toimintoa tehtäisiin samat määritelmät useaan otteeseen samasta aktiviteetista, koska yhden aktiviteetin aikana voi esiintyä useita korkeita kiihtyvyyspiikkejä. Seuraavana on esimerkki tästä toiminnallisuudesta: Jos aktiviteetin loppu on määritelty tapahtuvan kiihtyvyyden FIFO-muistin kohdassa 380, niin silloin funktio 2 ei tee uusia tarkastuksia korkeista kiihtyvyyspiikeistä, ennen kuin se on päässyt tähän kohtaan. Eli samassa esimerkissä pysyen, kun funktio 2 on löytänyt korkean kiihtyvyysarvon muistin kohdasta 200, niin se sivuuttaa uusien korkeiden kiihtyvyysarvojen etsimisen seuraavan 180 arvon kohdalla, koska etäisyys muistin kohdan 380 ja 200 välillä on 180 arvoa.

Kaatumisen pituuden määrittelemisessä otettiin myös huomioon mahdollisen vapaapudotuksen esiintyminen ennen aktiviteettia. Jos juuri ennen aktiviteettia on esiintynyt huomattavaa vapaapudotusta, niin silloin aktiviteetin pituuden minimi- ja maksimiraja-arvoa lyhennettiin. Seuraavana on esimerkki tämän toiminnallisuuden tarkoituksesta: Henkilö pyrkii tasapainon menettämisen jälkeen suojaamaan itseään liikuttamalla käsiä ja ottamalla ensimmäisen iskun vastaan polvilla. Tässä tapauksessa kiihtyvyysdatassa ei esiinny merkittävää vapaapudotusta, koska nämä liikkeet aiheuttavat ylös/alas-liikehdintää datassa. Tällöin myös kaatumisen aluksi määrittyy se hetki, kun nämä liikkeet aloitettiin, koska aktiviteetin alku on se hetki, kun kiihtyvyysdatassa on tarpeeksi monta perättäistä matalaa arvoa. Jos kaatuminen olisikin johtunut esimerkiksi pyörtymisestä, niin silloin kaatumisen aluksi määrittyisi vasta se hetki, kun henkilö on iskeytynyt ensimmäisen kerran maahan, koska mitään suojaavia liikkeitä ei ole tehty. Koska kaatumisen alku olisi tässä tapauksessa hyvin paljon lähempänä maahan iskeytymistä, niin algoritmin mittaama kaatumisen kesto olisi myös lyhyempi. Näiden suojaavien liikkeiden poissaolo johtaa kuitenkin siihen, että henkilön rannekin laskee tasaisesti kohti maata, joka on tunnistettavissa kiihtyvyysdatasta vapaapudotuksena, jolloin aktiviteetin pituuden raja-arvoa voidaan lyhentää ja tunnistaa se kaatumisena.

Ilmanpaineen muutoksen mittauksessa haluttiin ilmanpainedataa 2 s ennen aktiviteettia sekä 6 s aktiviteetin jälkeen, jotta voitaisiin tarkastaa, että se on kas-

vanut sekä lyhyellä että hieman pidemmällä aikavälillä. Lyhyen aikavälin tarkistuksen tarkoituksena oli, että ilmanpaineen kasvun syy on juuri tämä aktiviteetti, josta on löydetty korkea kiihtyvyyspiikki. Pidemmän aikavälin tarkistuksen tarkoituksena oli varmistaa, että lyhyen aikavälin kasvu ei johtunut yksittäisten mittauksien epätarkkuuksista vaan että ilmanpaine on todellakin kasvanut myös pidemmällä aikavälillä, mikä viittaisi siihen, että henkilö on siirtynyt alemmalle tasolle, ja jäänyt makaamaan maahan. Tässä ongelmana oli, että kun aktiviteetin kiihtyvyyslaskelmat oli tehty, oli ilmanpainedataa yleensä tullut vasta 1 s verran aktiviteetin lopun jälkeen. Tämä ratkaistiin koodaamalla viive, jonka tarkoituksena oli odottaa vielä 5 s ajalta uusia ilmanpaineen arvoja, jonka jälkeen ilmanpaineen muutos laskettiin.

Toinen tapa tunnistaa, että kellon käyttäjä on maannut hetken aikaa maassa kaatumisen jälkeen, oli koodata funktioon 2 viive odottamaan uusia kiihtyvyysarvoja aktiviteetin lopun jälkeen. Jos 5 s aktiviteetin lopun jälkeen oli syntynyt uusia korkeita kiihtyvyysarvoja, niin funktio 2 ilmoitti funktiolle 6, että ilmanpaineen tarkastusta ei tarvitse tehdä, ja aktiviteetti oli näin määritelty normaaliksi aktiviteetiksi. Tällä tavalla ei pelkästään saada poistettua turhia hälytyksiä tapauksista, joissa henkilö on heti kaatumisen jälkeen noussut ylös, vaan tämä oli myös hyvä tapa määritellä jatkuvat liikesarjat, kuten taputtaminen, normaaleiksi aktiviteeteiksi.

5 Tulokset

5.1 Algoritmin tulokset

Algoritmia kehitettiin sillä periaatteella, että se ei saisi määritellä yhtäkään normaalia aktiviteettia kaatumiseksi, ja tässä onnistuttiinkin. Algoritmi ei siis aiheuttanut yhtäkään väärää hälytystä, ja se tunnisti 103 kaatumista 130:stä. Täten algoritmin sensitiivisyys, eli herkkyys oli 0,79 ja spesifisyys, eli tarkkuus oli 1. Spesifisyys tarkoittaa oikeiden negatiivisten (TN) osuutta kaikista negatiivisista (TN + FN), eli tässä kontekstissa todennäköisyyttä, että normaali aktiviteetti luokitellaan oikein normaaliksi aktiviteetiksi. Sensitiivisyys taas tarkoittaa oikeiden positiivisten (TP) osuutta kaikista positiivisista (TP + FP), eli tässä kontekstissa

todennäköisyyttä, että kaatuminen luokitellaan oikein kaatumiseksi. Taulukossa 2 esitellään algoritmin tuloksista muodostettu virhematriisi, johon on myös liitettynä sensitiivisyyden ja spesifisyyden tulokset.

Taulukko 2. Virhematriisi, sekä sensitiivisyys ja spesifisyys algoritmin tuloksista.

Actual positives: n = 130 Actual negatives: n = 260		**Actual class**	
		Positive (**P**)	Negative (**N**)
Predicted class	Positive (**P**)	103	0
	Negative (**N**)	27	260
Sensitiivisyys		0.79	
Spesifisyys		1	

Algoritmin kehitystavasta johtuen taulukon 2 virhematriisin tulokset ovat hyvin yksinkertaiset. Algoritmi määritteli 27 kaatumista väärin normaaliksi aktiviteetiksi, joten FN = 27. Tämä oli hyväksyttävä virhemarginaali, koska näin saatiin määriteltyä jokainen normaali aktiviteetti oikein, eli TN = 260, mikä tarkoittaa sitä, että mitatusta datasta ei ole syntynyt yhtäkään väärää hälytystä. Väärien hälytysten välttäminen on todella tärkeää, koska mitä vähemmän niitä syntyy, sitä todennäköisemmin mahdollinen loppukäyttäjä jatkaisi kaatumisentunnistuksen käyttöä rannekellossaan.

Normaalien aktiviteettien määrä oli 260. Nämä olivat kuitenkin sellaisia, joista pystyttiin määrittelemään aktiviteetin alku ja loppu. Normaaleja aktiviteetteja sisältävissä tiedostoissa oli kuitenkin yhteensä 2817 korkeiksi määriteltyjä kiihtyvyysarvoja, eli jokaisesta normaalista aktiviteetista ei edes pystytty määrittelemään aktiviteetin alkua ja loppua, joka jo itsessään aiheutti sen, että algoritmi pystyi sulkemaan ne pois tarkemmista tarkasteluista. Esimerkiksi jos taputettiin 10 kertaa putkeen, niin algoritmi ei välttämättä tunnistanut näitä kymmeneksi erilliseksi tapahtumaksi.

Jos algoritmi tarkastelisi puhtaasti vain kiihtyvyyssignaalien piirteitä tunnistetun aktiviteetin alun ja lopun välistä, niin tällöin se tunnistaisi 115 kaatumista, jolloin sensitiivisyys olisi 0,88. Samalla tavalla 45 normaalia aktiviteettia tunnistettaisiin väärin kaatumisiksi, eli spesifisyys olisi 0,83. Taulukossa 3 vertaillaan näitä tuloksia lopullisen algoritmin tuloksiin.

Taulukko 3. Algoritmien virhematriisien sekä sensitiivisyyksien ja spesifisyyksien tulosten vertailu.

Lopullinen algoritmi	**Algoritmi ilman aktiviteetin ulkoisten kiihtyvyyspiikkien, sekä paineentarkastusta**
TP = 103	TP = 115
FP = 0	FP = 45
TN = 260	TN = 215
FN = 27	FN = 15
Sensitiivisyys = 0,79	Sensitiivisyys = 0,88
Spesifisyys = 1	Spesifisyys = 0,83

Taulukon 3 oikeanpuoleisen pystysarakkeen tuloksista voi huomata, että FP = 45, eli jos algoritmi ei tarkastelisi ilmanpaineen kasvua sekä korkeiden kiihtyvyyspiikkien esiintymistä juuri ennen ja jälkeen aktiviteetin, niin se tunnistaisi 45 normaalia aktiviteettia väärin kaatumiseksi. Tämä tarkoittaa sitä, että nämä 45 normaalia aktiviteettia näyttävät kiihtyvyysdatassa hyvin samankaltaiselta kuin kaatumiset. Mittaamalla ilmanpaineen kasvua sekä uusien korkeiden kiihtyvyyspiikkien esiintymistä juuri ennen ja jälkeen aktiviteetin, pystyi lopullinen algoritmi määrittelemään myös nämä 45 tapausta oikein normaaleiksi aktiviteeteiksi.

5.2 Tulosten arviointi

Tulosten arvioinnissa pitää ottaa huomioon, että kaikki data on kerätty vain yhdeltä henkilöltä. Tämä pyrittiin ottamaan algoritmissa huomioon antamalla mahdollisimman laajat skaalat eri piirteiden parametreille, jotta nämä piirteet voisivat

toistua myös muillakin henkilöillä. On kuitenkin hyvin vaikea arvioida, kuinka hyvin yleistettävissä kyseiset piirteet ovat henkilöille, jotka ovat esimerkiksi huomattavasti pidempiä, lyhyempiä, tai vanhempia, kuin datan kerännyt henkilö. Myöskään kaikkia mahdollisia kaatumisskenaarioita ja normaaleja aktiviteetteja oli mahdotonta mitata, varsinkaan tämän insinöörityön aikamääreissä.

Kaiken kaikkiaan, koska uusi algoritmi tunnisti kaatumisia mitatusta datasta huomattavasti paremmin kuin Navigilin alkuperäinen algoritmi, niin voidaan suurella varmuudella sanoa, että sen pitäisi suoriutua paremmin myös muillakin henkilöillä. Isompana huolenaiheena onkin, että pystyykö tässä insinöörityössä kehitetty algoritmi välttämään väärien hälytysten syntyä oikeassa elämässä yhtä hyvin kuin Navigilin alkuperäinen algoritmi. Ennen uuden algoritmin käyttöönottoa tätä tullaankin testaamaan pidemmällä aikavälillä, ja testauksen perusteella tullaan mahdollisesti hienosäätämään algoritmin parametreja.

6 Pohdinta

Erilaisia tapoja kaatua on lukemattomia, ja niin on myös erilaisia aktiviteetteja, jotka eivät ole kaatumisia, mutta jotka datassa samankaltaiselta näyttävät. Tästä johtuen voidaan sanoa, että on mahdotonta kehittää ennalta määritettyihin raja-arvoihin perustuvaa algoritmia, joka osaisi täydellisesti erottaa kaikki kaatumiset normaaleista aktiviteeteista kaikilla eri käyttäjillä. Jopa Applen kaltaiset yritykset, joilla on valtavat resurssit kerätä dataa käyttäjiltä ja kehittää näiden perusteella koneoppimisalgoritmeja Apple Watch-rannekelloihin, eivät ole tässä täydellisesti onnistuneet [19; 20]. Apple pyrkii kuitenkin säätämään algoritmejaan käyttäjäkohtaisemmiksi kysymällä jokaisen kaatumishälytyksen jälkeen, onko käyttäjä kaatunut vai ei. Tämä on varmasti tulevaisuuden suunta kaatumisentunnistuksessa, koska näin algoritmissa ei tarvitse tehdä kompromisseja, jotta se toimisi kaikilla käyttäjillä, vaan algoritmi oppii erottelemaan kaatumiset normaaleista aktiviteeteista käyttäjäkohtaisesti. Tällainen kehitystyö vaatii kuitenkin jatkuvaa datankeräystä valtavalta määrältä käyttäjiä, mihin harvalla yrityksellä on varaa, ottaen huomioon, että kaatumisentunnistus on rannekelloissa vain yksi toiminto muiden seassa. Täten toiseksi paras vaihtoehto on tehdä mitatun datan perusteella yleistyksiä, miltä kaatuminen näyttää ja miltä se ei

näytä, ja soveltaa tämän perusteella kehitettyä algoritmia kaikkien käyttäjien rannekelloissa. Tiettyjä käyttäjäkohtaisia ominaisuuksia voitaisiin kuitenkin tulevaisuudessa ottaa huomioon myös Navigilin algoritmissa. Esimerkiksi ilmanpaineen kasvun raja-arvoja voitaisiin muuttaa sen mukaan, kuinka pitkä käyttäjä on.

Koska nykyisillä algoritmeilla vääjäämättä syntyy myös vääriä hälytyksiä, haasteena onkin määritellä, minkälainen suhde tunnistettujen kaatumisien ja väärien hälytysten välille halutaan, mitä itsessään on jo hyvin vaikeaa määritellä tarkasti. Mitä tiukemmin kaatumisissa tapahtuvat piirteet algoritmissa määritellään, niin todennäköisesti sitä vähemmän syntyy vääriä hälytyksiä, mutta joitakin kaatumisia saattaa tästä johtuen jäädä tunnistamatta. Asiaa vaikeuttaa se, että kaatuminen on harvinainen tapahtuma, kun taas normaalit aktiviteetit, kuten käsien heiluttelu, ovat yleisempiä. Mitä herkemmin algoritmi aiheuttaa vääriä hälytyksiä, sitä todennäköisemmin käyttäjä ottaa toiminnon pois käytöstä, mikä tekee koko toiminnosta turhan. Jos taas algoritmin raja-arvot on asetettu hyvin tiukoiksi, voi käyttäjä käyttää tuotetta pitkäänkin ollen tyytyväinen siihen, että vääriä hälytyksiä ei synny. Mutta samassa esimerkissä pysyen, jos käyttäjä sitten vihdoin kaatuu pitkään sen jälkeen, kun toiminto on otettu käyttöön, ja algoritmi ei tätä tunnista, voi hän silloinkin todeta, että toiminto on huono ja ottaa sen pois käytöstä.

On myös hyvä ottaa huomioon, että ranne on yksi vaikeimmista mittauskohdista, josta mitatun datan perusteella pyritään tunnistamaan kaatumisia. Tästä johtuen on hyvä muistaa, että kaatumisentunnistus on rannekelloihin sovellettuna vain lisätoiminto, eikä täydellinen tapa tunnistaa kaatumisia. Esimerkiksi jos omainen tiedostaa, että hänen isovanhemmallansa on suuri riski kaatua, ja sen johdosta loukkaantua vakavasti, niin silloin varmin tapa saada tieto näistä kaatumisista olisi hankkia esimerkiksi jokin vyötäröön asetettava laite, jonka päätarkoituksena on vain tunnistaa kaatumiset. Rannekelloissa on kuitenkin se hyvä puoli, että niitä on luonnollista pitää kädessä lähes koko päivän ajan, joka lisää todennäköisyyttä, että laite on käyttäjällä käytössä silloin, kun kaatuminen tapahtuu.

7 Yhteenveto

7.1 Työn yhteenveto

Insinöörityön tavoitteena oli kerätä kattava datapaketti kaatumisista ja normaaleista aktiviteeteista, ja näiden perusteella kehittää Navigilin kaatumisentunnistusalgoritmia. Dataa saatiinkin kerättyä kattavasti, ja kehitetyn algoritmin todettiin suoriutuvan hyvin kerätystä datasta.

Alla on listattuna algoritmia kehittäessä syntyneitä tärkeitä havaintoja:

- Istuma- ja makuuasennosta tapahtuvia kaatumisia on hyvin vaikea tunnistaa, koska niissä ilmanpaineen kasvu on hyvin vähäistä.
- Koska ilmanpaineen kasvu seisaaltaan kaatuessakin on suhteellisen pieni ja ilmanpaineen mittaus ei ole täysin tarkkaa, ei ilmanpaineen kasvua pystytä aina tunnistamaan edes näistä kaatumisista.
- Normaali aktiviteetti voi näyttää kiihtyvyysdatassa hyvin samankaltaiselta, kuin kaatuminen.
- Mitä useampia erilaisia kaatumistyyppejä halutaan tunnistaa, sitä vaikeampi vääriä hälytyksiä on välttää.
- Navigil 580 rannekellon sulautettu järjestelmä aiheuttaa rajoituksia sille, kuinka paljon dataa voidaan kerralla pitää muistissa, mikä vaikuttaa menetelmiin, joilla kiihtyvyyden ja ilmanpaineen piirteitä määritellään algoritmissa.

Kaiken kaikkiaan dataa tutkiessa saatiin hyvä yleiskuva siitä, miten kiihtyvyys- ja painesignaalit käyttäytyvät eri tilanteissa. Algoritmia kehittäessä taas opittiin erilaisia tapoja mitata piirteitä näistä signaaleista. Työn aikana opittiin myös hyväksymään rajoitukset, joita datankeruun menetelmät, käytössä olevat resurssit, kaatumisentunnistuksessa käytettävä laite, sekä sen sijainti kehossa aiheuttavat kaatumisentunnistuksen tarkkuuteen. Näitä tietoja voidaan tulevaisuudessa hyödyntää kaatumisentunnistustoiminnon jatkokehityksessä.

7.2 Jatkokehitys

Seuraavana jatkotoimenpiteenä on aloittaa Pythonilla kirjoitetun algoritmin kääntäminen C-kielelle. Kun tämä käännös on saatu tehtyä, tehdään testaussuunnitelma, jonka perusteella algoritmin suoriutumista testataan todellisissa tilanteissa. Testaus tulee pitämään sisällään väärien hälytysten seurantaa pidemmällä aikavälillä sekä kaatumisentunnistustarkkuuden mittaamista eri kokoisilla koehenkilöillä. Näiden testien perusteella algoritmin parametreja tullaan mahdollisesti hienosäätämään, ennen kuin algoritmi otetaan käyttöön myös loppukäyttäjillä.

Tätä insinöörityötä voidaan myös käyttää pohjana mahdollisissa tulevissa kaatumisentunnistukseen liittyvissä insinööritöissä Navigililla. Tulevissa insinööritöissä voitaisiin esimerkiksi suorittaa laajempi datankeräysprojekti useilla erityyppisillä koehenkilöillä ja tämän pohjalta validoida tässä insinöörityössä kehitettyä algoritmia.

Lähteet

1 Falls. Verkkoaineisto. 2021. WHO. <https://www.who.int/news-room/fact-sheets/detail/falls> 26.4.2021. Luettu 17.8.2021.

2 Road traffic injuries. 2021. Verkkoaineisto. WHO. <https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries> 21.6.2021. Luettu 18.1.2022.

3 Emily, Kwan; Sharon, Straus. 2014. Assessment and management of falls in older people. Canadian medical association journal.

4 Havulinna, Satu; Häkkinen, Hanna; Karinkanta, Saija; Kettunen, Jyrki; Piirtola, Maarit; Pitkänen, Tiina; Punakallio, Anne; Sihvonen, Sanna. 2017. Kaatumisten ja kaatumisvammojen ehkäisyn fysioterapiasuositus. Verkkoaineisto. Terveysportti. <https://www.terveysportti.fi/dtk/sfs/avaa?p_artikkeli=sfs00003> 26.10.2017. Luettu 20.8.2021.

5 Hoey, Jesse; Shehroz, Khan. 2017. Review of fall detection techniques: A data availability perspective. Medical engineering & physics. Vol. 39, s. 12-22.

6 Azzopardi, George; Ellul, Joshua; Wang, Xueyi. 2020. Elderly fall detection: A literature survey. Frontiers in robotics and AI. Vol. 7.

7 Li, Ruijiang; Murphy, Martin; Naqa, Issam. 2015. Machine learning in radiation oncology. E-kirja. Springer.

8 Toivonen, Hannu. 2002. Klusterointimenetelmät-seminaari, johdanto. Luentomoniste. Helsingin yliopisto.

9 How does an Accelerometer Work - Physics of Probeware. 2018. Verkkoaineisto. PocketLab. <https://archive.thepocketlab.com/educators/lesson/how-does-accelerometer-work-physics-probeware> 16.4.2018. Luettu 19.1.2022.

10 What is MEMS technology? Verkkoaineisto. MEMS and Nanotechnology Exchange. <https://www.mems-exchange.org/MEMS/what-is.html> Luettu 19.1.2022.

11 Jost, Danny. 2019. What is an accelerometer? Verkkoaineisto. Fierce Electronics. <https://www.fierceelectronics.com/sensors/what-accelerometer> 11.7.2019. Luettu 15.9.2021.

12 Dadafshar, Majid. 2015. Accelerometer and gyroscopes sensors: Operation, sensing, and applications. Verkkoaineisto. Maxim integrated.

<https://pdfserv.maximintegrated.com/en/an/AN5830.pdf> 17.3.2015. Luettu 25.11.2021.

13 Mok, Frederick; Ngo, Low; Seng, Ho; Singh, Ranjit. 2002. A silicon piezoresistive pressure sensor. IEEE.

14 Civit-Balcells, Anton; Dominguez-Morales, Manuel; Gutierrez-Galan, Daniel; Luna-Perejon, Francisco. 2020. Low-power embedded system for gait classification using neural networks. Journal of low power electronics and applications. Vol. 10, s. 93 -108.

15 Gymstick. Verkkoaineisto. <https://www.gymstick.fi/foldable-gym-mat-200-x-100-x-5cm.html> Luettu 23.1.2022.

16 Abbas, Manuel; Jeannés, Régine; Saleh, Majd. 2020. FallAllD: An Open Dataset of Human Falls and Activities of Daily Living for Classical and Deep Learning Applications. IEEE sensors journal. Vol. 7, s. 1849 -1858.

17 Anaconda Individual Edition. Verkkoaineisto. < https://www.anaconda.com/products/individual> Luettu 14.2.2022.

18 The science of machine learning. Verkkoaineisto. <https://www.ml-science.com/confusion-matrix> Luettu 20.11.2021.

19 Fall detection false trigger. Verkkoaineisto. Apple <https://discussions.apple.com/thread/8554856> 14.10.2018. Luettu 11.1.2022.

20 Fall detection false alarms. Verkkoaineisto. Apple <https://discussions.apple.com/thread/858434> 26.9.2018. Luettu 11.1.2022.