Liikennevälineen automaattinen tunnistaminen

Arttu Ekholm

Tampereen yliopisto
Informaatiotieteiden yksikkö
Vuorovaikutteinen. teknologia
Pro gradu -tutkielma
Ohjaaja: Markku Turunen
Toukokuu 2015

Tampereen yliopisto
Informaatiotieteiden yksikkö
Vuorovaikutteinen teknologia
Arttu Ekholm: Liikennevälineen automaattinen tunnistaminen
Pro gradu -tutkielma, 47 sivua
Toukokuu 2015

Liikkumismuodon automaattinen tunnistaminen mahdollistaa yksilöidyn, tilanteeseen sopivan informaation tarjoamisen käyttäjälle. Erilaiset liikkumismuodot, kuten joukkoliikenne, eroavat toisistaan piirteiltään. Kerättyyn harjoitteludataan vertaamalla liikkumismuoto on mahdollista tunnistaa suurella todennäköisyydellä oikein, mutta riippuvuus aikaisemmin kerätystä datasta haittaa sovelluksien välitöntä käyttöönottoa.

Tutkielmassa selvitettiin kuinka hyvin liikkumismuoto on mahdollista tunnistaa automaattisesti vertaamalla matkapuhelimen GPS-paikkatietoja raitiovaunujen reaaliaikaisiin GPS-päivityksiin. Selvitys tapahtui keräämällä aineistoa käyttäjän liikkumisesta ja vertaamalla sitä kerättyyn raitiovaunujen paikkatietodataan, ja ajamalla data itse kehitetyn, ennalta määriteltyjen piirteiden vertailuun perustuvan algoritmin läpi. Algoritmi suoriutui tunnistamisesta hyvin, vaikka siitä paljastui selkeitä puutteita. Tulosten perusteella on perusteltua väittää, että on mahdollista kehittää alustariippumaton ohjelmistokomponentti, joka tunnistaa käyttäjän liikkumismuodon yksinkertaisilla säännöillä ja ilman aikaisemmin kerättyä harjoitteludataa.

Avainsanat ja -sanonnat: kontekstitietoinen tietojenkäsittely, toiminnan tunnistaminen, GIS

Sisällys

1. Johdanto

Matkapuhelimien käyttö on yleistynyt 2000-luvun aikana kaikkialla maailmassa. Puhelimet ovat kehittyneet ja nykyään useissa niin kutsutuissa älypuhelimissa on erilaisia ympäristön tilaa seuraavia mittalaitteita, kuten esimerkiksi GPS-paikannin, kompassi ja kiihtyvyyssensori. Näitä käyttäjän tilaa seuraavia ja tilan muutoksiin reagoivia sovelluksia kutsutaan kontekstitietoisiksi sovelluksiksi.

Joukkoliikenteen reittioppaat tarjoavat hyviä käyttökohteita puhelimien mittalaitteille ja kontekstitietoisuudelle. Navigointia voi helpottaa asettamalla käyttäjän lähtöpisteen kartalle käyttäjän koordinaattien perusteella ja opas voi muistuttaa matkan aikana seuraavista pysäkeistä ja kehottaa matkustajaa jäämään pois oikealla pysäkillä. Käyttäjän tilaa on mahdollista tulkita monipuolisemmin kuin pelkän sijainnin perusteella. Mitä enemmän tietoa kontekstista on, sitä enemmän käyttäjää voidaan auttaa ennakoimalla ja tulkitsemalla tilannetta. Lähialueen liikennevälineiden reaaliaikainen sijainti ja tiedot pysäkeistä auttavat tekemään syvempää tulkintaa matkustajan kontekstista, esimerkiksi selvittämään minkälaisella ajoneuvolla hän matkustaa.

Vallitsevan tilan päättely ei ole koskaan täysin luotettavaa, ja tilasta voi tehdä vain arvioita. Navigointitilanteessa ihmisen liikkumisessa kävellen tai esimerkiksi bussilla tai metrolla on tietynlaisia tunnistettavia piirteitä. Bussilla liikkuessa käyttäjän matkapuhelin mittaa suurempia positiivisia ja negatiivisia kiihtyvyyksiä kuin kävellessä ja keskinopeus on suurempi kuin kävellessä. Yksittäinen piirre ei riitä sellaisenaan tunnistamaan liikkumismuotoa, vaan tulkinnan pitää perustua useampaan piirteeseen. Aikaisemmat käyttäjän liikkumistapaa arvioivat menetelmät ovat perustuneet kattavan harjoitteludatan keräämiseen. Tulkinta liikkumismuodosta on tehty käyttämällä tilastollisia malleja, esimerkiksi Markovin mallia. Harjoitteludataa käyttämällä saadaan tarkkoja arvoita nykyhetken liikennevälineestä, mutta sovelluskäyttöön sellainen malli ei sovellu hyvin, koska sovelluksen toiminnan edellytyksenä olisi tarpeeksi suuren harjoitteludatan omaaminen. Ilman riittävää määrää harjoitteludataa ei sovelluksella olisi mahdollisuutta tunnistaa oikeita piirteitä eri liikkumismuodoista sovelluksen käyttöönoton yhteydessä.

Tässä tutkielmassa testataan hypoteesia, jonka mukaan matkustajan liikennevälineestä voi tehdä onnistuneen tulkinnan ilman aikaisempaa harjoitteludataa. Tämä edellyttää, että tarjolla on reaaliaikainen joukkoliikenteen sijaintia välittävä palvelu. Tulkinnan toimintaperiaate on käyttäjän ja lähellä olevien liikennevälineiden eri ominaisuuksia kuvaavien muuttujien määrittely ja painotusten antaminen muuttujille. Muuttujien yhteenlasketuista summista jokaiselle liikennevälineelle voidaan antaa arvo, joka kuvaa todennäköisyyttä olla matkustajan käyttämä liikenneväline. Hypoteesin taustalla on kirjoittajan kiinnostus kehittää automaattista liikennevälineen tulkintaa hyödyntävä komponentti, joka olisi välittömästi käyttöönotettavissa osana mitä tahansa

mobiilisovellusta. Komponentin yksi merkittävä ominaisuus olisi lähes välitön tulkintojen tekeminen. Menetelmä ei kuitenkaan sulkisi pois harjoitteludatan käyttöä; menetelmän muuttujia ja vakioita olisi mahdollista hienosäätää aikaisemmin kerätyn datan perusteella.

Hypoteesia testataan suunnittelemalla yksinkertainen algoritmi, joka jäljittelee reaaliaikaista liikennevälineen tunnistamista. Jäljitteleminen tapahtuu syöttämällä algoritmille koehenkilöiden nauhoittamaa testidataa. Algoritmi tekee koehenkilöiden nauhoittamasta testidatasta tulkintoja, joita verrataan koehenkilöiden oikeasti käyttämiin liikkumismuotoihin. Koehenkilöiden nauhoittama testidata sisältää sekä koehenkilön koordinaatit että samaan aikaan liikenteessä olevien raitiovaunujen paikkatiedot.

Tutkielman toisessa luvussa avataan toiminnan tunnistamisen menetelmiä ja siihen liittyvää käsitteistöä. Toiminnan tunnistamisen menetelmät kuvataan mobiililaitteiden kontekstissa, kuten myös menetelmissä hyödynnettävät mittalaitteet. Kolmas luku esittelee olemassa olevia paikkatietojärjestelmiä ja erilaisia paikkatietostandardeja. Luvuissa esitellään tarkemmin tilannetta Helsingissä, jossa testidatan kerääminen suoritettiin.

Hypoteesia varten kehitetty algoritmi ja siihen liittyvä suunnitteluprosessi kuvataan iteraatioineen neljännessä luvussa. Ennen suunnittelua kuvataan käytettävissä olevat resurssit, sitten menetelmän vaatimukset. Algoritmin vaatimuksina ovat alustariippumattomuus ja komponentti on voitava olla mahdollista lisätä toisen sovelluksen osaksi. Algoritmin suunnittelu tapahtuu kolmessa iteraatiossa. Iteraatioiden tarkoitukset ja niistä seuraavat muutokset kuvaillaan jokaisessa iteraatiossa.

Viides luku esittelee koeasetelman, testilaitteet, testiaineiston ja sen mittaustavan ja siihen käytetyt välineet. Tulokset käsitellään kuudennessa luvussa. Luvussa jäljitellään myös puutteellisen internet-yhteyden vaikutusta tuloksiin poistamalla osa päivityksistä ja raportoimalla tulokset. Kuudennen luvun lopussa arvioidaan hypoteesin paikkaansa pitävyyttä, menetelmän onnistumista ja soveltuvuutta käyttötarkoitukseensa. Luvussa myös pohditaan mahdollisia aiheita jatkokehitykseen.

2. Toiminnan tunnistaminen

Tämä luku esittelee liikennevälineen automaattista tunnistamista yhtenä toiminnan tunnistamisen muotona, esitellen aluksi aiheen tutkimuksen historiaa ja menetelmiä. Aluksi avataan kontekstin ja kontekstitietoisen tietojenkäsittelyn käsitteiden määrittelyä ja sitten luetellaan syitä automaattiselle liikennevälineen tunnistamiselle sekä käyttäjän että palveluiden tarjoajien näkökulmista. Kolmannessa alaluvussa esitellään aikaisempaa tutkimusta, vertaillen erilaisten sensoreiden käyttöä lopputulosten kannalta. Seuraavaksi esitellään yleisimmät menetelmät hankkia mittausdataa kontekstin tunnistamista varten, kuten GPS-paikannus ja kiihtyvyyssensori. Seuraava alaluku esittelee lyhyesti valmiita sovelluksia toiminnan ja liikennevälineen tunnistamiselle.

Toiminnan tunnistaminen on yksi kontekstitietoisen tietojenkäsittelyn osa-alue. Sen tavoitteena on saavuttaa arvio tilanteeseen liittyvästä toiminnasta tulkitsemalla mittalaitteiden arvoja. Pohjimmiltaan toiminnan tunnistaminen on kaikissa kontekstitietoisissa järjestelmissä samanlainen, noudattaen seuraavia neljää vaiheita (Choudhury ja muut, 2008):

1. Datan kerääminen. Dataa kerätään jatkuvasti tai pyydettäessä. Dataa kerätään ohjelmallisesti, esimerkiksi kalenterin tai sääpalvelun tai reittioppaan kautta, tai fyysisten mittalaitteiden, kuten kiihtyvyyssensorin tai kompassin avulla.
2. Datan suodatus. Mittausdata käsitellään tarvittaessa suotimen läpi. Suodatuksen tarkoitus on poistaa kohina ja yksittäiset poikkeamat.
3. Toiminnan tulkitseminen yhdistetyn datan pohjalta.
4. Tulkinnasta kertovan tiedon lähettäminen eteenpäin, ja muun järjestelmän reagointi

2.1. Kontekstista ja kontekstitiedosta

Ympäristön vallitsevaa tilaa kutsutaan kontekstiksi ja järjestelmää joka kykenee selvittämään ympäristön vallitsevan tilan, kontekstitietoiseksi. Paikkatietoisuus on yksi kontekstitiedon muoto, jossa käyttäjän sijainti on kontekstina. Kontekstitietoisuus yhdistetään yleensä mobiililaitteisiin, sillä mobiililaitteissa on useita mittalaitteita, kuten GPS-paikannin ja kiihtyvyyssensori. Termiä “kontekstitietoinen” käytti ensimmäisen kerran Schilit (1994), jonka määritelmän mukaan kontekstitietoiset laitteet yrittävät tehdä oletuksia käyttäjän tilanteesta. Hänen määritelmänsä kontekstista oli lähinnä paikkaa kuvaava. Dey (2001) määritteli kontekstin laajemmaksi kokonaisuudeksi, tarkoittamaan mitä tahansa informaatiota, joka jollain tavalla kuvaa käyttäjän ja sovelluksen vuorovaikutuksen kannalta relevantin asian tilaa. Vaikka kontekstia on yritetty määritellä useasta eri näkökulmasta ja siitä on muun muassa ISO-standardi, ISO-13407, on määritelmien käyttökelpoisuutta kyseenalaistettu sovellusten suunnittelussa (Tamminen ja muut, 2004).

Mobiililaitteisiin liittyen ihmisten käyttötilanteen tai toiminnan selvittämistä erilaisilla fyysisillä mittalaitteilla ja ohjelmallisilla menetelmillä on tutkittu paljon. Tavoitteena on ollut tarjota käyttäjille yksilöidympää tietoa tai varoittaa käyttäjää ympäristössä olevista asioista. Yksi suosittu tutkimussuunta on ollut käyttäjän kontekstin tunnistaminen pelkästään mobiililaitteiden sensoreiden ja palvelujen avulla. Tutkimuksien tuloksia on sovellettu laajasti mobiilisovelluksissa, esimerkiksi käyttäjän aktiivisuutta seuraavina ja mittaavina mobiilisovelluksina.

Sen lisäksi että kontekstitietoinen sovellus tai järjestelmä tunnistaa käyttäjänsä tilan, se myös reagoi tilaan ja tilassa tapahtuviin muutoksiin. Schilit ja muut (1994) määrittelevät kontekstitietoisen tietojenkäsittelyn (engl. context-aware computing) ryhmittelemällä kontekstitietoiset ohjelmat neljään ryhmään niiden toiminnan perusteella.

1. Proksimaalinen valinta. Järjestelmä järjestää vaihtoehdot niin, että läheisimmät vaihtoehdot ovat järjestyksessä ensimmäisiä. Yksinkertaisimmillaan proksimaalinen valinta on järjestää hakutulokset käyttäjän etäisyyden perusteella. Järjestely säästää käyttäjältä vaivan vertailla itse etäisyyksiä.
2. Automaattinen kontekstuaalinen uudelleenjärjestely. Komponenttien järjestely muuttuvan tilanteen mukaan. Sovellus voi esimerkiksi järjestää hakutuloksia samalla kuin sovelluksen käyttäjä liikkuu uuteen sijaintiin.
3. Kontekstuaalinen informaatio ja komennot. Käskyt tuottavat erilaisia tuloksia riippuen siitä minkälaisessa kontekstissa niitä tehdään. Sijaintia hyödyntävät palvelut voivat esimerkiksi rajata haettavaa tietoa käyttäjän lähialueelle.
4. Kontekstin laukaisemat toiminnot. Sovelluksessa on sääntöjä jotka määrittelevät kuinka kontekstitietoisen ohjelman pitäisi sopeutua. Sovellus voi ilmoittaa käyttäjälle, jos hänen lähellään tapahtuu jotain kiinnostavaa.

Tiivistettynä kontekstitietoisen tietojenkäsittelyn edellytyksenä on vallitsevan tilan, kontekstin, mittaaminen ja mitattavien asioiden perusteella tapahtuva päättely. Osa kontekstityypeistä on sovellusten kannalta tärkeämpiä kuin toiset tyypit, riippuen sovellusten tiedontarpeista ja toimintatavoista. Dey ja Abowd (1999) ovat ehdottaneet kontekstien taksonomiaksi rakennetta, jossa kontekstit ryhmitellään niiden atomisuuden mukaan primaarisiksi ja sekundaarisiksi. Sekundaariset kontekstityypit koostuvat primaarisista konteksteista, ja niiden järjestelyn perusteena voi käyttää primaaristen kontekstityyppien arvoja. Esimerkiksi konteksti “ulkona oleva sää” vaatii vähintään kaksi primaarista kontekstityyppiä - lämpötilan ja paikkatiedon, ollakseen hyödyllinen. Primaarisiksi konteksteiksi Dey ja Abowd määrittelivät sijainnin, identiteetin, ajan ja toiminnan.

Kontekstitietoisessa tietojenkäsittelyssä sijainti on usein käytetty primaarinen konteksti, erityisesti mobiilisovelluksissa. Sijaintia hyödynnetään monissa mobiilisovelluksissa, kuten esimerkiksi reittioppaissa, hakukoneissa, sosiaalisissa

käyttäjän sijaintia jakavissa palveluissa ja valokuvia kartalle tallentavissa sovelluksissa. Sijaintiin perustuvien palvelujen ja sovellusten suurta määrää mobiilisovelluksissa voi selittää sovellusten tekninen yksinkertaisuus – käyttäjän kontekstin, sijainnin, tulkitseminen onnistuu helposti esimerkiksi GPS-paikantimella kerättyjen koordinaattien perusteella.

2.2. Aikaisempi tutkimus

2000-luvun vaihteessa ja vuosituhannen alkupuolella tehty toiminnan automaattiseen tunnistamiseen liittyvä tutkimus keskittyi pitkälti käyttäjän liikkumismuotoihin ja fyysisen toiminnan tunnistamiseen puettavien mittalaitteiden avulla. Tunnistettavat liikkumismuodot olivat yksinkertaisia kuten käveleminen, juokseminen ja polkupyöräily, mutta monimutkaisempia toiminnan muotoja, kuten esimerkiksi ruoanlaittoa, pukeutumista ja lukemista on yritetty tunnistaa eri paikkoihin kehoa puettavilla sensoreilla (Atallah ja muut, 2010). 2000-luvun puolivälissä, samalla kun GPS-paikantimen ja kiihtyvyyssensorin sisältävät matkapuhelimet yleistyivät, alkoivat yleistyä tutkimukset joissa käytettiin vain sellaisia mittalaitteita joita löytyy matkapuhelimista, esimerkiksi vain yhtä kolmiakselista kiihtyvyyssensoria useamman kiihtyvyyssensorin sijaan.

Pelkästään GPS-paikannuksen käyttäminen on aikaisemmissa tutkimuksissa antanut huonompia tuloksia kuin GPS:n ja kiihtyvyyssensorien käyttäminen syötteen keräämiseen (Taulukko 1).

Tutkimus	Luokittelut	Mittalaitteet	Testidatan keräysaika	Käyttäjiä	Tarkkuus
Zheng ja muut, 2008a	Auto, linja-auto, polkupyörä, kävely	GPS	10 kuukautta	65	76.2%
Reddy ja muut, 2010	liikkumatta, kävely, juokseminen, polkupyörä, moottoriajoneuvo	GPS, kiihtyvyyssensori	50 päivää	16	93.6%
Patterson ja muut, 2003; Liao, Fox ja Kautz, 2004	kävely, bussi, auto	GPS, GIS	60 päivää	1	84%
Zheng ja muut, 2008b	auto, bussi, polkupyörä, kävely	GPS	6 kuukautta	45	74%

Taulukko 1. Aikaisempia tutkimuksia liikennevälineen automaattisesta tunnistamisesta tuloksineen (Stenneth ja muut, 2011).

2.3. Toiminnan tunnistamisessa käytettyjä sensoreita

Taulukossa 1 kuvatuissa tutkimuksissa mittausdataa on kerätty fyysisillä sensoreilla. Osassa tutkimuksista on käytetty matkapuhelimia, ja osassa mittausdataa on kerätty erillisillä, kehoon kiinnitettävillä mittalaitteilla. Käyttäjän käyttämän liikennevälineen tunnistamista ilman fyysisiä sensoreita ei nykytiedon mukaan ole mahdollista tehdä luotettavasti, vaan tunnistamiseen vaaditaan tietoja vähintään mittalaitteen sijainnista.

2.3.1. Kiihtyvyyssensori

Kiihtyvyyssensori mittaa kiihtyvyyttä. Ollessaan paikallaan kolmiakselisen kiihtyvyyssensorin akseleiden vektori on voimakkuudeltaan maapallon putoamiskiihtyvyys, eli 1 g (9,81 m/s^2). Putoamiskiihtyvyyden vaikutus voidaan eliminoida kiihtyvyydestä käyttämällä ylipäästösuodinta (engl. High-pass filter).

Kiihtyvyyssensoreiden käyttö mittalaitteena on ollut yksi eniten käytetty tapa tunnistaa ihmisen toimintaa ohjelmallisesti. Kiihtyvyyssensori kykenee mittaamaan

muutoksia tiheällä ja kiinteällä intervallilla (useat laitteet mittaavat lukemat yli 60 kertaa sekunnissa) ja suurella tarkkuudella. Aikaisemmissa tutkimuksissa on saatu hyviä tuloksia erilaisten toimintojen tunnistamisesta kehoon kiinnitetyillä kiihtyvyyssensoreilla. On selvitetty, että vain yksi kolmiakselinen kiihtyvyyssensori riittää tavallisimpien toimintatyyppien, kuten mm. paikallaan seisomisen, kävelyn, juoksemisen, portaissa kävelemisen ja vatsalihasliikkeiden tekemisen tunnistamiseen (Ravi ja muut, 2005). Mittaustarkkuus paranee jos sensoreita on useampia kuin kolme, mutta matkapuhelinten kolmiakselinen kiihtyvyyssensori riittää hyvin luotettavaan mittaukseen. Kiihtyvyyssensorilla varustetun laitteen ollessa kiinnitettynä tai lähellä kehoa, esimerkiksi vyöllä tai taskussa, on mahdollista mitata laitteen käyttäjän kävelemät askeleet ja tekemään arvioita käyttäjän kävelemästä matkasta keskimääräisen askelen pituuden perusteella. Askeleen pituuden keskiarvo määräytyy sukupuolen, iän, pituuden ja painon perusteella (Alvarez ja muut, 2006). Kiihtyvyyssensorin mittaamasta datasta on mahdollista päätellä käyttäjän käyttämät liikennevälineet (Stenham, 2011).

Kiihtyvyyssensorin käyttöä toiminnan tunnistamisessa rajoittaa ja hankaloittaa mittaustulosten muuttuminen laitteen ollessa kiinni eri kohtaa kehoa. Esimerkiksi kädessä pidetyn matkapuhelimen mittausdata on erilaista taskussa pidetyn laitteen keräämään mittausdataan. Laitteen siirtäminen paikasta toiseen aiheuttaa merkittävän muutoksen mittaustuloksessa. Siirtämistä tapahtuu mm. silloin kun käyttäjä ottaa matkapuhelimen esiin taskustaan lukeakseen näytössä näkyvää tekstiä. Siirtämisestä tapahtuvaa mittausvirhettä eliminoidaan usein laskemalla mittausdatan keskiarvo ennalta määritellyn intervallin välein (esim. Stenhamin menetelmässä käytetty intervalli).

Ensimmäiset toiminnan tunnistamiseen kiihtyvyyssensorin avulla tapahtuvat tutkimukset perustuivat koeasetelmiin joissa koehenkilöille puettiin useita, valittuihin paikkoihin kiinnitettäviä kiihtyvyyssensoreita Kwapisz ja muut (2010) käyttivät puhelimen kiihtyvyyssensoreita tunnistamaan kuutta erilaista liikkumistapaa, mm. kävelemistä ja juoksemista. Heidän käyttämänsä menetelmä ei edellyttänyt puhelimen pitämistä samassa asennossa. Heidän tutkimuksensa osoitti, että toiminnan tunnistamiseen riittää yleensä yksi kolmiakselinen kiihtyvyyssensori joka löytyy useista matkapuhelimista. Yleisesti ihmisen toimintaa tunnistavat menetelmät onnistuvat tarkasti (yleensä 90-95% tulkinnoista on oikein), mutta tunnistettavat toimintatyypit ovat rajoitettuja, ja ovat usein monotonisia liikuntamuotoja kuten käveleminen, juokseminen tai voimistelu. Toinen tunnistettava toiminnan tyyppi on ollut käyttäjän käyttämä liikenneväline joukosta, joka koostuu kävelemisestä, pyöräilystä ja autosta tai määrittelemättömästä ajoneuvosta.

Kiihtyvyyssensoreita hyödyntäviä, terveyttä mittaavia sovelluksia on useita. Useat sovellukset mittaavat sovelluksen käyttäjän kävelemien askelien määrät. iPhonelle saatavilla oleva Fjuul-sovellus (Fjuul, 2014) laskee askelee ja arvioi käyttäjän

liikkumismuodon kävelemisen ja juoksemisen välillä sekä arvioi käyttäjän kuluttaman energian.

2.3.2. GPS-paikannus ja matkapuhelinmastojen kolmiomittaus

GPS-paikannus (Global Positioning System) perustuu maata kiertävien satelliittien maapallon pinnalle lähettämien signaalien aikaerojen vertailuun. GPS-paikannus on yleinen matkapuhelimen ominaisuus. GPS-paikantimen tarkkuus vaihtelee navigointiympäristöstä – mitä parempi näkyvyys taivaalle on, sitä parempi signaali paikannussatelliitteihin on. Suurissa kaupungeissa korkeat rakennukset voivat haitata GPS-paikannusta merkittävästi (engl. Canyon effect). Matkapuhelimia varten on kehitetty A-GPS-paikannusmenetelmä (engl. Assisted GPS) joka nopeuttaa paikannusta hyödyntämällä matkapuhelinverkkoja. A-GPS-paikannusmenetelmä vaatii internet-yhteyden toimiakseen.

Toinen, GPS-paikannuksen kaltainen matkapuhelimien hyödyntämä paikannusmenetelmä on puhelinmastojen kolmiomittauksen perusteella tapahtuva. Se on hitaampi ja epätarkempi menetelmä kuin GPS-paikannus, mutta kaikki matkapuhelimet pystyvät siihen, ja menetelmä on toteutettu useiden puhelimien navigointiohjelmissa varavaihtona GPS-paikannukselle. Jotkut älypuhelimien paikannuskirjastot, kuten iOS-mobiililaitteiden CoreLocation-kirjasto osaavat yhdistää puhelinmastojen, GPS-sateliittien ja WLAN-tukiasemien lähettämät sijainnit yhdeksi kokonaisuudeksi, nopeuttaen paikannusta ja vähentämällä laitteen virrankulutusta.

GPS-paikannuksen käytön suosio liikennevälineen tunnistamiseen liittyvissä tutkimuksissa ja sovelluksissa perustuu paikannuksen datan yksinkertaisuuteen ja siihen liittyvän tulkitsemisen tarpeen vähyyteen. GPS-paikantimen tarjoama tieto on yksinkertaisimmillaan koordinaattiparin ja aikaleiman sisältävä tietorakenne. Useasta aikaleima-koordinaattiparista voi johtaa liikutun matkan pituuden, nopeuden ($v = d/t$), kiihtyvyyden (a = deltav/deltat) ja liikkeen suunnan. GPS-paikannuksen tarkkuuteen vaikuttavat mm. korkeat rakennukset, jotka aiheuttavat etenkin suurissa kaupungeissa epätarkkuutta paikannuksiin (engl. Canyon effect). Poikkeamia paikannuksessa voi vähentää erilaisilla ohjelmallisilla suotimilla, kuten esimerkiksi Kalman-suotimella (Malleswari ja muut, 2009). GPS-paikannus ei edellytä matkapuhelimen tai paikantimen pitämistä esillä, eikä laitteen asento tai sen sijainnin vaihtuminen esimerkiksi taskusta käteen vaikuta mittaustuloksiin.

GPS-paikannuksen käyttö toiminnan tunnistamisen ainoana kontekstin lähteenä on ongelmallista paikantimen riittämättömän tarkkuuden takia. Koordinaateista johdettu nopeus ja kiihtyvyys ei yksin riitä määrittämään ihmisen liikkumismuotoa esim. kävelemisen ja juoksemisen välillä.

2.3.3. Muita sensoreita

GPS-paikannus ja kiihtyvyyssensori eivät ole ainoita sensoreita mitata käyttäjän toimintaa. Puhelimen kompassia käytetään pääasiassa karttasovelluksissa. Matkapuhelimen yhdellä tai useammalla kameralla voidaan mitata ympäristön valoisuutta ja mikrofonilla on mahdollista mitata ympäristön meluisuutta. Kumpikin sensori edellyttää laitteen pitämistä esillä. Taustalla sensoridataa keräävät ja toimintaa automaattisesti tunnistavat sovellukset edellyttävät toimimista myös silloin kun matkapuhelin on käyttäjän taskussa. Tästä syystä kamera ja mikrofoni eivät ole usein käytettyjä sensoreita toiminnan tunnistamiseen. Kameran kuvaa hyödyntävää hahmontunnistusta on tutkittu paljon ja pitkään, ja myös puheentunnistus on paljon tutkittu tieteenala, mutta niiden käyttö käyttäjän kontekstin tunnistamisessa on ollut hyvin vähän tutkittua. Syynä tähän voi olla kummankin kontekstin monimutkaisuus verrattuna sijainnin ja liikkumisen yksinkertaisuuteen kontekstina. Usein kontekstitietoiset kamerapohjaiset järjestelmät on sijoitettu kiinteästi, esimerkiksi potilaan kotiin tai neuvotteluhuoneisiin ja toimistoon. (Choujaa ja Dulay, 2009)

Fyysisten sensoreiden lisäksi matkapuhelimen sovelluksien sisältämiä tietoja voidaan käyttää eräänlaisina ohjelmallisina sensoreina. Kello ja kalenterisovellus sisältävät tietoa, joista voidaan tulkita vuorokaudenaika ja tapahtuma tai paikka jossa käyttäjä on tai jonne hän on menossa.

2.4. Käyttökohteita ja käyttäjien tarpeita liikennevälineen automaattiselle tunnistamiselle

Liikennevälineen automaattinen tunnistaminen tarkoittaa sitä, että järjestelmä tietää minkälaisella ja millä liikennevälineellä käyttäjä matkustaa. Tietoa. Käyttäjän käyttämät liikkumismuodot voidaan tunnistaa ja tallentaa, ja sen perusteella voidaan suositella sellaisia reittejä, joissa liikutaan käyttäjän aikaisemmin suosimilla matkustusmuodoilla.

Mobiilisovelluksien käyttäjille voidaan tarjota yksilöityä mainontaa. Facebook osti käyttäjän liikkumista ja toimintaa tulkitsevan ja seuraavan Moves –sovelluksen toukokuussa 2014. Sovelluksen algoritmia voidaan hyödyntää yksilöidyn ja tilanteenmukaisen mainonnan tarjoamiseen.

Joukkoliikennettä käyttäville ihmisille voidaan tarjota yksityiskohtaisia navigointiohjeita, jos käyttäjän sijainti ja aktiivinen liikenneväline ovat tiedossa. Esimerkiksi seuraavien pysäkkien nimet, lähestyvät pysäkit ja asemat sekä poikkeustiedot ja ruuhkatiedotukset voidaan tarjota tarpeen mukaan käyttäjälle, ilman, että hänen tarvitsee hakea niitä itse käsin. Huonosti näkevät ja täysin sokeat ihmiset hyötyvät puhekäyttöliittymistä. Puhutut ohjeet seuraavien pysäkkien nimistä, sijainneista ja matkustusohjeista helpottavat matkustamista. Normaalisti näkevät ihmiset voivat myös hyödyntää puhuttuja ohjeita ja ilmoituksia.

Puhutun tiedonvälityksen lisäksi visuaaliset ohjeet voidaan tarjota niin, että vain relevantti tieto on käyttäjälle näkyvillä. Se voi tarkoittaa bussi- tai raitiolinjan sijaintia kartalla, seuraavien pysäkkien sijaintia ja matka-aikojen ilmoittamista. Jos käyttäjän käyttämän joukkoliikennevälineen yksilöivä tunniste kyetään tunnistamaan nopeasti, voidaan käyttäjälle tarjota seuraavaa informaatiota matkan varrella olevista asioista, kuten reittiin liittyvää poikkeusinfoa. Esimerkiksi HSL:n reittiopasrajapinta tarjoaa tutkielman kirjoitushetkellä, maaliskuussa 2015, tietoa mahdollisista poikkeuksista, seuraavat pysäkit ja ajan koska liikenneväline on pysäkillä, reitin kartan pysäkkeineen ja ajastetun ohjeen jäädä pois oikealla pysäkillä.

2.5. Olemassa olevia menetelmiä liikennevälineen tunnistamiseen

Matkapuhelimien käyttöjärjestelmät sisältävät menetelmiä, joilla käyttäjän liikkumismuoto voidaan tunnistaa karkealla tasolla. On olemassa kaupallisia sovelluksia, jotka pystyvät tunnistamaan liikkumismuodon tarkemmin.

2.5.1. Apple Core Motion ja M7-prosessori

Applen syyskuussa 2013 julkaisema iPhone 5S –puhelin sisältää käyttäjän liikedataa keräävän ja prosessoivan prosessorin, M7:n. Prosessori toimii jatkuvasti taustalla. Puhelimen käyttöjärjestelmä kerää jatkuvasti tietoa käyttäjän liikkeistä kiihtyvyyssensorin ja GPS-paikantimen avulla. Kerätystä datasta lasketaan puhelimen käyttäjän liikkumat askeleet ja arvioidaan liikkumistapa liikkumattomuuden, kävelemisen, autolla matkustamisen ja juoksemisen väliltä. Kaikki sovellukset voivat kysyä tietoa käyttäjän liikkumiskäyttäytymisestä – joko takautuvasti tai reaaliaikaisesti. Reaaliaikaisissa päivityksissä kirjasto lähettää applikaatiolle takaisinkutsuja liikkumismuodon vaihtuessa tai halutun askelmäärän kertyessä.

2.5.2. Google Location

Google Location API on pääpiirteittäin samanlainen kuin Core Motion –rajapinta, ja tarjoaa samat asiat kuin Core Motion. Rajapinnan kautta voi tiedustella todennäköisintä käyttäjän nykyhetkistä liikkumismuotoa. Rajapinnan tunnistamat liikkumismuodot ja tilat ovat {IN_VEHICLE, ON_BICYCLE, ON_FOOT, STILL, TILTING, UNKNOWN}, eli ajoneuvossa matkustaminen, polkupyöräily, liikkuminen kävellen ja juosten, paikallaan oleminen, laitteen kallistuminen ja tuntematon liikkumismuoto. Kuten Core Motion –rajapinta, arviot liikkumismuodosta esitetään oliona joka sisältää arvioidun liikkumismuodon ja luottamusvälin arviolle. Google Location –rajapinta eroaa Core Location –rajapinnasta siten, että yhden todennäköisimmän arvion sijaan se toimittaa listan useasta todennäköisestä liikkumismuodosta. Toisin kuin Core Motion, Google Location ei ole riippuvainen erillisestä liikeprosessorista, vaan toimii osana Android-käyttöjärjestelmää.

3. Reittioppaat

Tässä luvussa esitellään olemassa olevia reittioppaita ja liikennevälineiden paikkatietopalveluja sekä tapoja, joilla reittioppaat palvelevat joukkoliikennettä käyttävien ihmisten tiedontarpeita. Sekä tutkielman koeasetelmassa käytettävä kaupunkialue että koeasetelmassa hyödynnettävä HSL Live –paikkatietorajapinta kuvaillaan tässä luvussa.

Service Interval for Real Time Information (SIRI) on XML-pohjainen protokolla, jonka tarkoituksena on toimittaa reaaliaikaista tietoa julkisen liikenteen liikennevälineiden ja palvelujen tilasta (SIRI, 2014). SIRI-protokollan mukaista reaaliaikaista tietoa tarjotaan useassa maassa, ja Suomessa sitä tarjotaan Tampereella ja Helsingissä. Iso-Britanniassa SIRI-protokollaa käyttäviä reittioppaita on kautta maan ja Yhdysvalloissa mm. New Yorkissa. Pohjoismaissa Suomen ulkopuolella SIRI-protokollaa hyödynnetään kattavasti Ruotsissa ja Norjassa (VDV, 2013). Aasiassa SIRI-protokollaa hyödyntäviä palveluita on käytössä Hongkongissa.

TransLink Open API on Kanadassa toimivien liikennevälineiden reaaliaikaisten tietojen toimittamiseen tarkoitettu rajapinta. HSL Live on suomalainen, pääkaupunkiseudulla toimiva reaaliaikaista paikkatietoa lähettävä rajapinta. Google GTFS-realtime (General Feed Transit Specification) on Googlen kehittämä standardi reaaliaikaisten paikkatietojen lähettämiseen. Yhdysvalloissa on useita kymmeniä GTFS-realtime-standardin mukaista paikkatietodataa lähettävää joukkoliikennepalvelua.

Taulukko 2:ssa esiteltävä tieto osoittaa sen, että palveluilla on joukko yhteisiä tietoja, ja sovellusta suunnitellessa voi olettaa kaikkien reaaliaikaisten paikkatietodataa toimittavien palveluiden toimittavan kyseisiä tietoja.

	HSL Live	**Translink (Translink, 2015)**	**GTFS-realtime (GTFS, 2015)**
Aikaleima	X	X	X
Pituus- ja leveyskoordinaatit	X	X	X
Ajoneuvon yksilöllinen tunniste (id)	X	X	X
Reitin tunniste	X	X	X
Reitin suunta	X	X	X
Reitin pisteet		X	

Taulukko 2. Reaaliaikaisten liikennevälinedataa lähettävien palveluiden ja standardin tietojen vertailua.

3.1. Olemassa olevia navigointijärjestelmiä Suomessa

Suomessa pääkaupunkiseudulla käytössä olevat, matkapuhelimille kehitetyt reittioppaat käyttävät pääasiassa Helsingin Seudun Liikenteen (HSL) tarjoamaa reittiopas-rajapintaa. Suurimmissa suomalaisissa kaupungeissa on vastaavanlaisia reittiopaspalveluita, kuten Tampereen Repa Reittiopas ja Turun alueella palveleva Brahe-reittiopas. Reittiopassovellukset palvelevat käyttäjiään visualisoimalla navigointiohjeet. Kontekstitiedon hyödyntäminen rajoittuu käyttäjän sijainnin automaattiseen paikantamiseen ja käyttämiseen reitin lähtöpaikkana ja nykyhetken käyttämisen lähtöaikana. Suomessa ei ole vielä navigointiapplikaatiota, joka olisi sillä tavalla kontekstitietoinen, että tunnistaisi käyttäjän liikkumismuodon ja mukauttaisi tilan perusteella matkaohjeita.

3.2. Liikennevälineiden erityispiirteet Helsingin seudulla

Helsingin julkisen liikenne koostuu linja-autoista, metrolinjasta, raitiovaunusta, lähijunasta ja Suomenlinnan lautasta. Kaupungissa on 17 metroasemaa, ja rakenteilla on kahdeksan asemaa lisää. Asemien väleistä viisi kulkee kokonaan maan alla, ja loput ovat kokonaan tai osittain maan pinnalla. Metrojen aikataulut ilmoitetaan, pois lukien vuorokauden ensimmäinen ja viimeinen vuoro, tarkkojen kellonaikojen sijaan vuoroväleinä. Vuorovälit vaihtelevat neljästä minuutista kymmeneen minuuttiin, ja tulevat puolittumaan länsimetron käyttöönoton jälkeen, vuonna 2016.

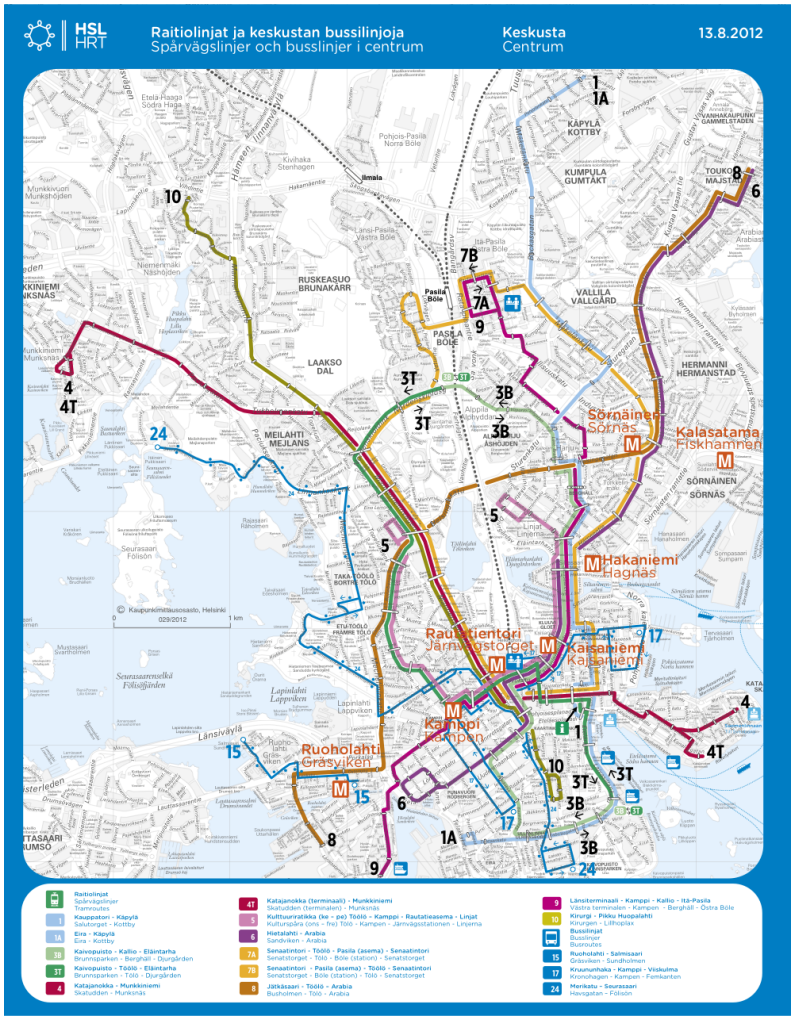
Helsingin kaupungin alueella on kolmetoista raitiolinjaa joiden varrella sijaitsee 266 pysäkkiä. Päivittäin raitiovaunuja käyttää 200 000 matkustajaa, ja vuonna 2009 raitiovaunuilla tehtiin yhteensä 54,9 miljoonaa matkaa. HSL:llä on käytössä 122 raitiovaunua ja linjojen vuoroväli on normaalisti arkiruuhkassa 5-10 minuuttia ja hiljaisena aikana 10-20 minuuttia.

3.3. Liikennevälineen tunnistamiseen käytettävissä olevat resurssit pääkaupunkiseudulla

Pääkaupunkiseudulla on käytössä kaksi reaaliaikaista liikennevälineiden koordinaatteja lähettävää järjestelmää: HSL Live ja VR:n Junat kartalla -palvelut. HSL Live on verkkopalvelu, joka toimittaa tietoja liikennevälineen sijainnista, liikkumissuunasta ja –nopeudesta ja muista liikennetiedoista reaaliaikaisesti. Viestit lähetetään push-viesteinä socket-yhteyttä pitkin. Palvelusta on mahdollista pyytää joko kaikki liikenteessä olevat välineet tai rajata kyselyä linjojen ja koordinaattien perusteella. Palvelu kattaa kaikki raitiovaunut ja metrot, mutta vain osan linja-autoista. Suomenlinnan lauttojen tietoja ei lähetetä palvelun kautta.

Reaaliaikaista tietoa tarjoavan rajapinnan lisäksi HSL tarjoaa myös staattisen HTTP GET –rajapinnan, jota kautta on mahdollista kysellä käytössä olevat raitiolinjat koordinaattijoukkoina ja listana pysäkkien koordinaatteja. Reittiopas-rajapinta laatii ehdotuksen kuljettavasta reitistä käyttäjän sijainnin, kellonajan ja liikkumismieltymysten perusteella. Poikkeustiedoille on oma rajapintansa, kuten myös pysäkkikohtaiselle seuraavia lähtöjä ilmoittavalle palvelulle. HSL:n uutisista ja palveluista kertovalle kanavalle on myös rajapintansa. (HSL, 2014)

Junat kartalla API on VR:n ylläpitämä, junien reaaliaikaisia paikkatietoja toimittava rajapinta. Paikkatiedot toimitetaan RSS-syötteenä. Syöte ei ole muodoltaan minkään taulukossa 2 kuvatun standardin mukainen, vaan VR:n itse määrittelemä malli. Syötteen päivitysväli on viisi sekuntia. Rajapinta tarjoaa tiedon junan sijainnista, nopeudesta, suunnasta ja junan seuraavasta pysäkistä. Juna-asemien sijainnit ja liikenteessä olevien junien tunnisteet on saatavilla rajapinnan kautta. (ITS, 2014)



Kuva 1. Helsingin seudun liikenteen raitiovaunulinjat pysäkkeineen 13.8.2012.

4. Liikennevälineen automaattinen tunnistaminen GPS-paikkatiedon avulla

Tässä luvussa kuvaillaan tutkielman osana suunniteltu ja toteutettu oma liikennevälineen tunnistamismenetelmä, joka pohjautuu käyttäjän laitteen GPS-paikantimen keräämien paikkatietojen ja HSL Live –rajapinnan lähettämien paikkatietojen vertailuun. Menetelmän tavoitteet ja vaatimukset kuvaillaan perustellen kyseisen menetelmän sopivuutta valitussa kontekstissa. Menetelmään liittyvät haasteet ja ongelmat kuvataan luvun alussa, ennen menetelmän toteuttamisen kuvaamista. Menetelmän kehittäminen esitellään luvun lopussa iteraatioihin jaettuina. Kehittäminen on jaettu iteraatioihin, joiden sisällöt kuvataan omissa alaluvuissaan.

Kahdella - uusimmalla ja ajallisesti edeltävällä - aikaleimalla ja koordinaateilla saa laskettua liikennevälineen karkean suunnan, nopeuden kahden päivityksen välillä ja kiihtyvyyden. Johdettujen suureiden tarkkuus riippuu silloin täysin koordinaattien tarkkuudesta. HSL Live –rajapinta tarjoaa nämä tarvittavat tiedot, joten rajapinnan tarjoaman data sopii koeasetelmaa varten. Lisäksi HSL Live –rajapinta on tarjoamaltaan tiedoilta yhteneväinen SIRI-rajapinnan kenttien kanssa, joten koeasetelma olisi toistettavissa toisessa kaupungissa, jossa olisi SIRI-yhteensopiva paikkatietopalvelu.

4.1. Kehitettävän menetelmän tavoitteet

Tässä tutkielmassa kehitettävälle menetelmälle tunnistaa liikkumismuoto asetettiin kaksi tavoitetta. Ensimmäinen tavoite on tunnistaa käyttäjän nykyhetkellä käyttämä liikenneväline lähes reaaliaikaisesti, mukaan lukien liikennevälineen yksilöivä tunniste. Tämä tarkoittaa käytännössä sitä, että menetelmää käyttämällä voisi muutaman sekunnin sisällä vastata kysymykseen “Onko käyttäjä raitiovaunussa, ja jos on, niin missä raitiovaunussa?”. Menetelmän tarkoitus ei ole tunnistaa muita liikkumismuotoja kuin käveleminen ja raitiovaunut. Menetelmän pääasiallinen käyttökohde olisi joukkoliikenteen käyttäminen, ja siksi juokseminen, polkupyöräily ja autolla matkustaminen jätetään pois. Liikkumistyyppien määrää vähentämällä voidaan liikkumismuodon tunnistamisesta tehdä teknisesti yksinkertaisempi. Menetelmä ei sulje pois muiden liikennevälineiden tunnistamisen lisäämistä algoritmiin myöhemmin. Muut liikennevälineet jätettiin pois tutkimuksesta tutkielman laajuuden ja monimutkaisuuden kasvamiseen liittyvien huolien takia. Helsingissä ei myöskään ollut mahdollista saada reaaliaikaisia paikkatietopäivityksiä kaikista busseista, joten menetelmän testaaminen busseilla olisi ollut erittäin aikaa vievää ja vaatinut paljon suunnittelua etukäteen. Rajaamalla liikennevälineet vain raitiovaunuihin, saatiin testiaineiston ja mittausten laajuus pysymään hallittavan kokoisena. Merkittävin ero luvussa kolme esiteltyihin

tunnistamismenetelmiin on tieto käyttäjän käyttämän liikennevälineen yksilöivästä tiedosta, eli tieto millä yksittäisellä raitiovaunulla käyttäjä matkustaa.

Menetelmän toisena tavoitteena on sen mahdollisimman helppo integroitavuus muihin järjestelmiin. Ehtona on se, että menetelmä ei vaadi aikaisemmin kerätyn harjoitteludatan liittämistä komponentin mukaan. Komponentin tulisi tehdä tulkintoja käyttäjän tilanteesta lähes välittömästi käyttöönoton jälkeen. Helpon integroitavuuden yksi vaatimus on alustariippumattomuus, eli komponentti pitäisi olla mahdollista toteuttaa yleisimmille mobiilikäyttöjärjestelmille, ja mahdollisuus toteuttaa komponentti verkkopalveluna.

Liikennevälineen yksilöivän tiedon käyttäminen mahdollistaa lisäinformaation näyttämisen käyttäjälle matkan aikana. Esimerkiksi seuraavan pysäkin nimi ja jäljellä oleva matka voidaan ilmoittaa käyttäjälle, tai poikkeustiedot bussi- tai raitiolinjasta voidaan tuoda esiin, ja käyttäjälle voidaan kertoa koska ja missä hänen tulee jäädä pois kyydistä.

4.2. Vaatimukset ohjelmistoarkkitehtuurin ja tietotyyppien suhteen.

Luvussa 2 läpikäydyissä aikaisemmissa tutkimuksissa tavoitteena oli pystyä hyödyntämään paikkatietoja oppivissa järjestelmissä. Tässä tutkielmassa tarkastellaan aihepiiriä sovelluskehityksen näkökulmasta, eli kehitettävän menetelmän kehittäminen tuotteeksi huomioidaan. Tutkimuksen on tarkoitus selvittää, voiko paikkatietoja hyödyntämällä tunnistaa käyttäjän liikennevälineen. Menetelmän on tarkoitus olla mahdollisimman alustariippumaton ja siten tarvittaessa eri alustoille siirrettävä ratkaisu. Menetelmälle annetut vaatimukset asettavat rajoitteita mittalaitteiden ja liikennetietojärjestelmien hyödynnettävien tietojen suhteen. Alustariippumattoman menetelmän tulee käyttää vain niitä paikkatietoja, mitä yleisimpien mobiilikäyttöjärjestelmien (tässä tutkielmassa iOS, Android ja Windows Phone) paikannuskirjastot tarjoavat. Liikennevälineistä hyödynnettävien tietojen tulisi olla vain SIRI-standardin mukaista tietoa.

4.3. Menetelmän toimintatapojen vertailua ja suunnittelua

Algoritmia suunnitellessa tavoitteena oli kehittää mahdollisimman yksinkertainen ja ylöspäin skaalautuva menetelmä tunnistaa erilaisia liikennevälineitä. Suunnitteluprosessi alkoi kuvailemalla kävelemisen ja raitiovaunulla matkustamisen yleisimmät piirteet niissä konteksteissa, joista GPS-paikannus ja HSL Live tarjoaa tietoa. Menetelmän suunnittelun alkuvaiheessa pohdittiin kuutta erilaista teknistä ratkaisua käyttäjän liikkumismuodon selvittämiseen.

1. **Käyttäjän liikkeeseen perustuva päättely**. Käyttäjien GPS-päivityksien nopeutta ja kiihtyvyyttä verrataan raja-arvoihin, ja tulosten perusteella määritellään käyttäjän tilanne joko raitiovaunussa matkustamiseksi tai

kävelyksi. Menetelmä ei kykene erittelemään käyttäjän nykyhetkellä käyttämää liikennevälinettä. Myös liikennevälineen linja olisi useissa tilanteissa mahdotonta eritellä, sillä esimerkiksi useat bussit ja raitiovaunut kulkevat samoja reittejä pitkin. Menetelmän hyöty olisi varsin rajallinen, eikä se tarjoaisi merkittävää lisäarvoa käyttötapauksiin.

2. **Kiihtyvyyssensorin käyttö liikkumisen tunnistamisessa.** Liikkeen suunnan ja etäisyyden voi määritellä derivoimalla kiihtyvyysvektorin kahdesti. Useat lähteet toteavat menetelmän käytännössä niin epätarkaksi, ettei menetelmää voi käyttää sijainnin selvittämiseen matkustukseen liittyvissä tapauksissa. Kiihtyvyyssensorin lukemiin vaikuttaa myös käyttäjän aiheuttama liike, jota syntyy kävellessä, istuessa, ajoneuvon tärinästä ja matkapuhelimen käytöstä. Kiihtyvyyssensoria käyttämällä voisi tunnistaa kävelyn, mutta käveleminen ei ole täysin poissuljettua, sillä käyttäjät voivat kävellä liikennevälineissä. On kuitenkin epätodennäköistä, että käyttäjä pitäisi matkapuhelintaan jatkuvasti samassa asennossa, joten pelkästään matkapuhelimen kiihtyvyyssensorin käyttöä tunnistamisessa ei voida pitää järkevänä.
3. **Käyttäjän GPS-päivityshistorian vertaaminen liikennevälineiden reittijanoihin.** Käyttäjän GPS-päivityksiä kerätään ylös ja sijaintipäivityksiä verrataan liikennevälineiden reitteihin. Menetelmän tarkkuus paranee kerättyjen GPS-päivitysten myötä. Tilanteissa, joissa kaksi liikennevälineen reittilinjaa kulkee samaa katua pitkin, ei tämä menetelmällä voida erottaa kumpaa reittiä pitkin kulkevassa liikennevälineessä käyttäjä on.
4. **Käyttäjän GPS-päivitysten vertaaminen reaaliaikaiseen liikennevälineiden paikkatietoon.** Menetelmällä jokaista käyttäjän uutta GPS-päivitystä verrattaisiin jokaiseen alueella olevan liikennevälineen viimeisimpään paikkatietoon, ja sopivin liikenneväline tai kävely valittaisiin käyttäjän liikennemuodoksi. Menetelmä mahdollistaisi sekä liikennevälineen linjan, että liikennevälineen yksilöivän tunnisteen löytämisen. Esimerkiksi HSL:n kaikissa liikennevälineissä on yksilöivä tunniste, joka lähetetään HSL Live –rajapinnan paikkatietodatan mukana.
5. **Käyttäjän GPS-päivitysten vertaaminen pysäkkien sijaintiin.** Menetelmä vertaa käyttäjän sijaintia ja liikkumisnopeutta pysäkkien sijaintiin. Saadulla tiedolla ei voida tehdä päätelmiä käytetystä liikennevälineestä, mutta pysäkin ja käyttäjän läheinen sijainti ja paikallaan pysyminen voidaan tulkita liikennevälineen odottamiseksi.
6. **Aikataulutietojen vertaaminen käyttäjän GPS-päivityksiin.** Käyttäjän GPS-päivityksiä verrataan liikennevälineiden aikatauluihin, ja vertailun perusteella

päätellään käyttäjän käyttämä liikenneväline. Menetelmä on todennäköisesti erittäin virhealtis aikataulujen poikkeamille ja raitiovaunujen myöhästelyille.

Yllä kuvatuista valittiin kehitettäväksi menetelmäksi vaihtoehto neljä, eli GPS-paikannuksen ja reaaliaikaisen paikkatiedon lähettämisen vertaaminen, sillä se vaikutti luotettavalta, yksinkertaiselta ja alustariippumattomalta ratkaisulta.

4.4. Mobiililaitteiden käyttöjärjestelmien paikkatietokirjastot

Paikkatietodatasta, esimerkiksi HSL Live –datasta, saa paljon erilaisia tietoja liikennevälineen käyttäytymisestä, kuten kiihtyvyyden ja tiedon onko väline pysäkillä vai ei. Muissa kaupungeissa ja maissa olevat liikennevälineiden tietoja lähettävät järjestelmät eivät välttämättä tarjoa vastaavia tietoja, joten kaikkia HSL Liven tarjoamia tietoja ei kannata hyödyntää algoritmissa. Järkevintä on hyödyntää vain niitä tietoja, joita muissakin palveluissa todennäköisesti on. Käytettävien tietojen joukko tulee olemaan {aikaleima, linja tai liikennevälineen yksilöivä tunniste, pituuskoordinaatti, leveyskoordinaatti}. Nämä tiedot sisältyvät SIRI-protokollan määrittelemään päivitykseen, joten menetelmä tulee olemaan SIRI-järjestelmän kanssa yhteensopiva. Tietoa esimerkiksi raitiovaunun olemisesta pysähtyneenä pysäkillä ei ole järkevä käyttää, vaikka siitä olisikin hyötyä tunnistamisessa.

Paikkatietokirjastot tarjoavat pääpiirteittäin samat palvelut – käyttäjän sijainnin ja liikkeen selvittämisen, mutta kehitettävän komponentin alustariippumattomuuden takia voidaan hyödyntää vain niitä tietoja, joita yleisimmät mobiilialustat tarjoavat. Tätä tutkielmaa varten yleisimmiksi alustoiksi valitaan iOS-, Android- ja Windows Phone –alustojen uusimmat versiot.

Ominaisuus	**iOS, CLLocation**	**Android, Location (Android Developers, 2013)**	**WP8 (Microsoft, 2013)**
Aikaleima	X	X	X
Kompassisuunta	X	X	X
Etäisyyksien laskeminen	X	X	-
Pituuden ja leveyden tarkkuus	X	X	X
Korkeuden tarkkuus	X	-	X
Korkeus merenpinnasta	X	X	X
Pituus- ja leveysasteet	X	X	X
Nopeus	X	X	X

Taulukko 3. iOS-, Android- ja Windows Phone –käyttöjärjestelmän sijaintikirjastojen paikkatieto-olioiden sisältämät tietotyypit.

Taulukko 3 osoittaa, että paikkatietokirjastojen tarjoamien päivitysten tiedot ovat suurimmaksi osin samanlaisia kaikilla kolmella alustalla. Korkeus merenpinnasta ei ole navigointiin liittyviä etäisyyksiä laskiessa merkittävä tieto. Android-käyttöjärjestelmän Location-olion *getAccuracy()*-metodi palauttaa säteen, jonka sisällä Location-olion kuvaama sijainti on 68% todennäköisyydellä, eli yhden keskipoikkeaman verran. Windows Phone 8 eikä iOS –dokumentaatioissa ole määritelty tarkkuuksien määritelmää, mutta sijaintipalveluiden käyttämien tietorakenteiden samankaltaisuus viittaa vahvasti siihen, että muissakin alustoissa pituus-, leveys- ja korkeuskoordinaattien tarkkuus on määritelty saman, yhden keskihajonnan määritelmän mukaisesti. Nopeustiedon käyttämistä sellaisenaan ei suositella iOS SDK:n CLLocation-luokan dokumentaatiossa (Apple, 2014a). Windows Phonen paikkatietokirjasto ei tarjoa suoraan kahden sijaintiolion välisen etäisyyden laskemista, mutta se ei ole ongelma alustariippumatonta menetelmää kehittäessä, sillä kahden koordinaatin välisen etäisyyden mittaamista varten on olemassa valmiita kaavoja.

Sekä Windows Phone että Android -alustat tarjoavat yksityiskohtaisempaa tietoa toimitetusta paikkatiedosta kuin iOS, kuten tietoa paikannukseen osallistuneista GPS-satelliiteista. iOS-käyttöjärjestelmän paikkatietopäivityksissä ei erikseen saa tietoa paikannukseen osallistuneista välineistä. Kehitettävä menetelmä ei siis saa huomioida sitä,

onko paikannukseen käytetty jotain muuta paikannusmenetelmää kuin GPS-paikannusta, esim. puhelinmastojen kolmiomittausta. Kehitettävä menetelmä voi hyödyntää tärkeimpiä tietoja, eli koordinaatteja, pituus- ja leveyspiirin koordinaattien tarkkuutta ja nopeutta.

4.5. Menetelmän tekniset haasteet

Menetelmää hyödyntävää sovellusta tullaan ajamaan älypuhelimessa, joten algoritmin on tehtävä arvio liikennevälineestä tarpeeksi nopeasti, ilman että aktiivinen sovellus pysähtyy tai jumittuu paikalleen. Laitteen käytössä oleva laskentateho on otettava huomioon algoritmin suunnittelemisessa. Algoritmi voidaan suunnitella niin, että sitä on mahdollista ajaa monisäikeisessä käyttöjärjestelmässä taustasäikeessä, jolloin sovelluksen käyttöliittymä ei pysähdy tai hidastele algoritmin prosessoinnin aikana.

Applikaation on kyettävä suorittamaan päätelmät käyttämättä oppimisdataa. Suurin tutkimusongelman selvittämisen motivaatio on ollut tulosten soveltaminen reittiopaskäytössä. Reittioppaan tulisi toimia riittävän hyvin heti ensimmäisestä käyttökerrasta lähtien, joten oppimisdataan perustuvat ratkaisut eivät ole sopivia. Algoritmin tavoitteena oli aloittaa tulkinta heti saadessaan tietoa sekä liikennevälineistä että päivityksen käyttäjän sijainnista. Tällöin algoritmia pystyisi hyödyntämään sekä kertakäyttöisiin, että jatkuvaa seurantaa edellyttäviin käyttötapauksiin.

Algoritmin voi sijoittaa sekä ohuen että paksun (engl. thin ja thick client) ohjelmistoarkkitehtuurin mukaiseen sovellukseen. Osa tai kaikki paikkatietodatan prosessoinnista voidaan siirtää palvelimelle, mutta tarvittaessa kaiken prosessoinnin voi tehdä pelkästään mobiilisovelluksessa.

Algoritmin käyttämä tallennustila laitteen muistissa on oltava kohtuullinen. Kaikki algoritmin keräämä ja tallentama tieto on oltava mahdollista poistaa, jos käyttöjärjestelmän käytössä oleva muisti on loppumassa ja algoritmin on pystyttävä jatkamaan toimintaansa tämän jälkeen. Mobiililaitteissa on usein hyvin rajattu määrä muistia käytössä ja sovellusten odotetaan poistavan tarpeettomia resursseja muististaan vapaan muistin ollessa vähissä. Esimerkiksi iOS-sovelluksissa käyttöjärjestelmä voi vaatia sovellusta poistamaan ylimääräiset resurssit muistista koska tahansa. Jos sovellus ei kykene tähän, voi käyttöjärjestelmä sulkea sovelluksen koska tahansa.

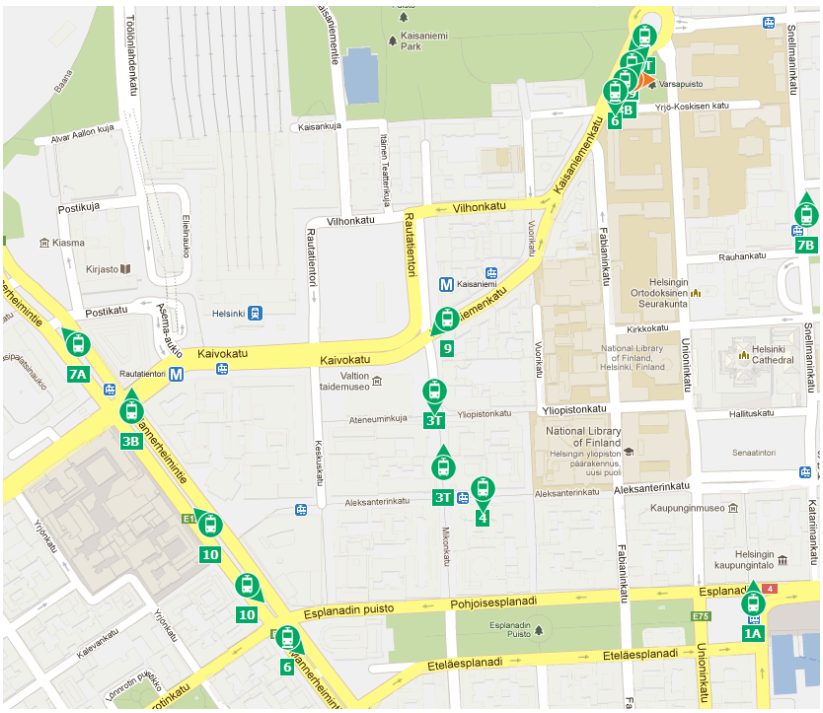
Tutkimusongelmaa varten hyödynnettiin Helsingin Seudun Liikenteen (HSL) tarjoamaa reaaliaikaista kulkuneuvodataa. Yhtäaikaisten ajossa olevien liikennevälineiden määrä Helsingissä on melko pieni, ja suurin osa raitiovaunuista kulkee omia, suurimmaksi osaksi toisistaan erillään olevia kiskoja pitkin. Muutamassa paikassa, esimerkiksi Hakaniemessä ja Kaivokadulla, useampi raitiovaunulinja kulkee toistensa läheltä ja samoja kiskoja. Tällaisissa solmukohdissa algoritmilla on mahdollisuus tehdä virheellisiä tulkintoja, koska niin monta raitiovaunua on lähellä toisiaan. Menetelmän toimintatavan

on oltava sellainen, että se toimisi sekä suurissa ja ruuhkaisissa kaupungeissa, että pienemmillä ja liikenteeltään vähäisemmillä alueilla.

Kuten Reddy et al. (2010) osoittivat, voisi sekä kiihtyvyyssensorin että GPS-datan yhdistäminen yhdeksi syötteeksi parantaa oikean liikennevälineen tunnistamisen todennäköisyyttä. Kiihtyvyyssensorin lukemien hyödyntäminen on kuitenkin hankalaa, jos käyttäjä pitää laitetta kädessään, kuten matkustaessa ja navigointiapua hakiessaan on tapana tehdä. Siksi kiihtyvyyssensorin lukemia ei hyödynnetä tässä tutkimuksessa.

4.6. Liikennevälineen toimintaan liittyvät haasteet

Testialueena käytetty Helsingin ydinkeskusta sisältää useita paikkoja, joissa on monta vierekkäistä pysäkkiä ja paljon raitiovaunuliikennettä, kuten rautatieaseman edustan, Hakaniemen torin ja Helsingin Yliopiston metroaseman sisäänkäynnin. Pysäkeillä on voi olla usein paljon lähekkäin pysähtyneitä liikennevälineitä, kuten Kuva 2 oikeassa yläkulmassa, Kaisaniemen pysäkeillä (kuvassa Varsapuiston vasemmalla puolella). Perehtymällä testialueen raitiovaunuliikenteen tiheyteen ja raitiovaunukiskojen sijaintiin, arvioitiin testialueella olevan kolme tilannetta joissa on suuri todennäköisyys virhetulkintaan. Ensinnäkin, on mahdollista, että algoritmi tekee paljon vääriä tunnistuksia tilanteissa, joissa lähekkäin on monta pysähtynyttä raitiovaunua, tai tilanteissa, joissa monta raitiovaunua liikkuu peräkkäin saman suuntaisesti.



Kuva 2. Helsingin ydinkeskustan raitiovaunuliikenne 12.3.2013 kello 18:30. Kuva on kuvankaappaus HSL Live –rajapintaa käyttävästä sivustosta. http://transport.wspgroup.fi/hklkartta/

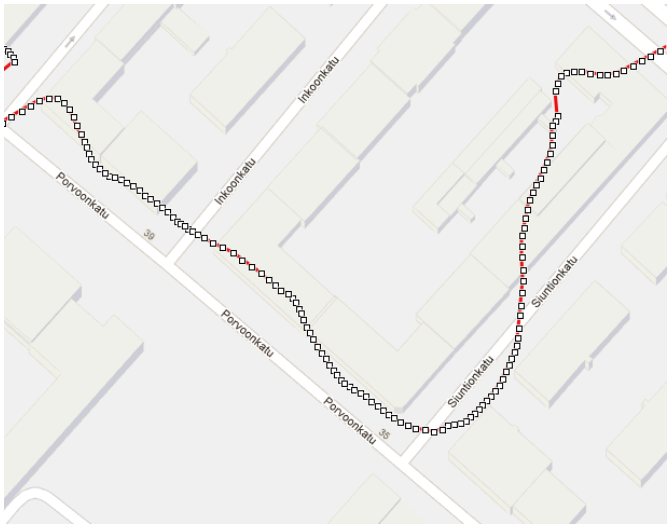
Toinen etukäteen oletettu ongelmatilanne tapahtuu heti käyttäjän noustessa raitiovaunuun tai poistuessa raitiovaunusta ja käyttäjän ollessa liikkumatta pysäköidyn raitiovaunun vieressä. Näissä tilanteissa on todennäköistä, että algoritmin tekemät tulkinnat ovat virheellisiä ennen liikennevälineen vaihtumista, vaihtumishetkellä ja jonkin aikaa vaihtumisen jälkeen.

Kolmas etukäteen havaittu ongelmatilanne ilmenee käyttäjän matkustaessa raitiovaunulla, kun vastakkaista raitiovaunukiskoa ajaa raitiovaunu suoraan kohti. Silloin algoritmi voi hetkellisesti tulkita käyttäjän raitiovaunun vastaantulevaksi raitiovaunuksi, sillä ne ovat hetken aikaa lähes vierekkäin.

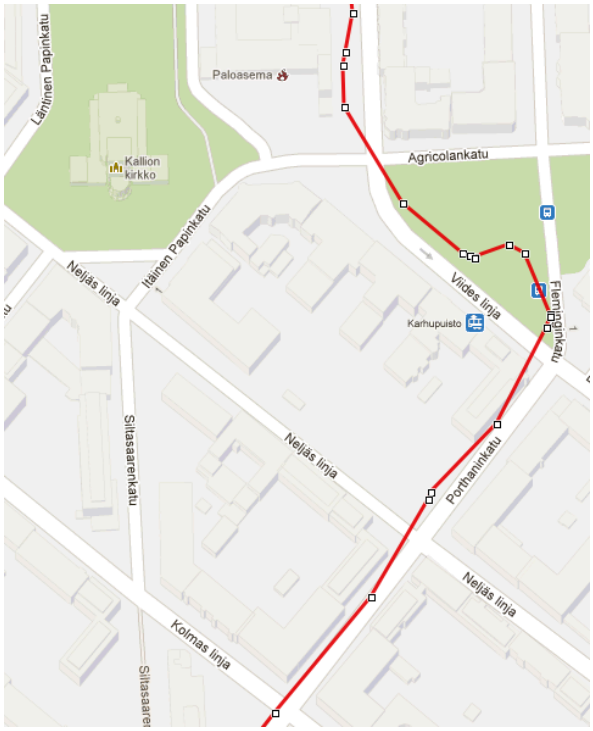
Mikään kolmesta ongelmatilanteesta ei ollut luonteeltaan sellainen, että sen ilmeneminen olisi mahdollista vain Helsingissä eikä muualla maailmassa, vaan ongelmatilanteet esiintyvät oletettavasti muissakin raitioliikennettä sisältävissä kaupungeissa. Yhtään sellaista ongelmatilannetta, jonka aiheuttaisi Helsingin ainutlaatuinen arkkitehtuuri tai liikennejärjestelyt, ei osattu määritellä etukäteen.

4.7. Esitutkimus

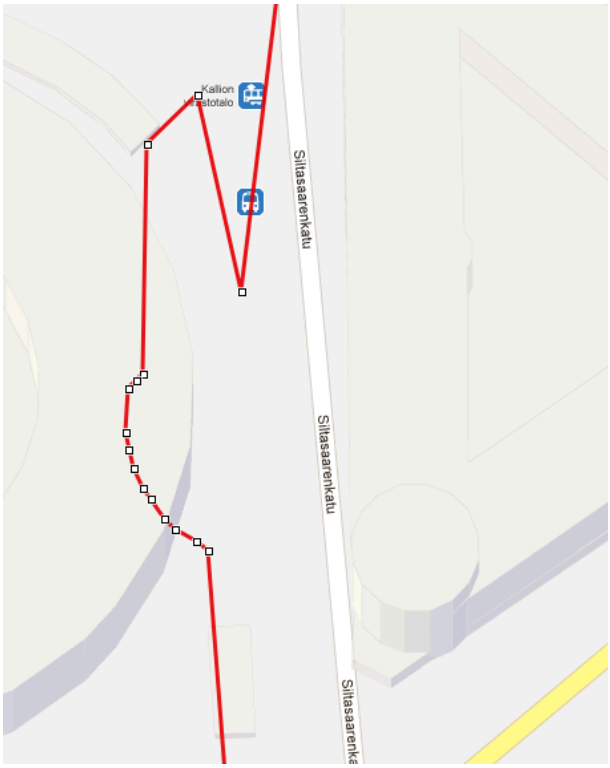
Ennen varsinaista mitta-aineiston keräämistä aineiston keräämisessä käytettävän testilaitteen GPS-paikantimen ja raitiovaunujen GPS-paikantimien tarkkuutta selvitettiin silmämääräisesti. Tutkielman kirjoittaja nauhoitti puhelimen GPS-paikantimella sekä kävelyä että raitiovaunulla matkustamista, visualisoiden kerätyt koordinaatit pisteinä kartalle. Raitiovaunujen paikantimien tarkkuus arvioitiin piirtämällä yksittäisten raitiovaunujen lähettämät sijaintipäivitykset kartan päälle tekemällä GPX-tiedostot kummastakin syötteestä. Sama tehtiin muutamalle kävely- ja raitiovaunulla matkustamista sisältäneelle nauhoitukselle. Matkapuhelimen paikantimella tallennettiin matka, jossa koehenkilö käveli suoraan suoraa tietä pitkin. Ennen tulosten analysointia testilaitteen GPS-paikantimen tarkkuus selvitettiin piirtämällä osan nauhoitusten sisältämät GPS-päivitykset kartalle GPS Visualizer (GPS Visualizer, 2013) –palvelun avulla. Karttakuvista kävi ilmi kävelyä sisältävien reittien epätarkkuus (Kuva 1). Raitiovaunulla matkustetut reitit vastasivat pääpiirteittäin niitä reittejä, joissa raitiovaunukiskot kulkevat (Kuva 4).



Kuva 3. Kävelyn aikana iPhone 4S-puhelimella nauhoitetut GPS-päivitykset kartalla. Kävelyreitti kulki Siuntionkatua ja Porvoonkatua pitkin. Päivityksen intervalli on noin sekunti.



Kuva 4. iPhone 4S –puhelimen GPS-päivityksiä matkustaessa raitiolinja 9:ää pitkin kulkevalla raitiovaunulla etelästä pohjoiseen Porthaninkatua ja Viidettä linjaa pitkin. Päivityksen intervalli on noin yksi sekunti.



Kuva 5. Poikkeamia raitiovaunun GPS-päivityksissä matkustaessa raitolinja 9:n raitiovaunulla kuvassa näkyvää Siltasaarenkatua pitkin. GPS-päivitysten väli on noin yksi sekunti. Kartan vasemmassa reunassa näkyy kahdeksankerroksinen Ympyrätalo.

Raitiovaunujen GPS-päivitykset vastasivat silmämääräisesti paremmin todellisuutta kuin matkapuhelimen GPS-paikantimet. Syynä voi olla raitiovaunujen parempi GPS-paikannin, joka on sijoitettu katolle raitiovaunun etuosaan. Esteetön yhteys taivaalle mahdollistaa tarkemman paikannuksen kuin talojen ja raitiovaunun katon katveeseen jäävän matkapuhelimen GPS-paikannin. Vaikka raitiovaunun GPS-paikkatiedot olivat tarkempia kuin iPhone 4S:n, näkyi raitiovaunujen liikkeissä virheitä. Useasti sekä raitiovaunun että matkapuhelimen GPS-päivityksissä esiintyi samanlaisia poikkeamia samoissa paikoissa, mikä voi johtua lähellä olevien rakennusten aiheuttamasta häiriöstä GPS-signaaliin.

Esitutkimuksen aikana CoreLocation-paikkatietokirjaston lähettämissä paikkatietopäivityksissä sijainnin tarkkuutta ilmaiseva accuracy-attribuutti vaihteli 5…500 metrin välillä. Kävellessä ulkona accuracy-attribuutin arvo oli 5…10 metriä, ja raitiovaunulla matkustaessa aina vähintään 10 metriä. Käytetyn liikennevälineen vaikutus GPS-paikantimen tarkkuuden huonontumiseen näytti olevan säännönmukainen ilmiö. Sen hyödyntämistä raitiovaunussa matkustamisen tunnistamiseen harkittiin, mutta ajatuksesta luovuttiin sillä ilmiö saattoi tapahtua vain Helsingin alueen raitiovaunuissa,

tai muiden kaupunkien raitiovaunujen rungolla saattoi olla enemmän tai vähemmän tarkkuutta huonontava vaikutus.

4.8. Menetelmän suunnittelu iteraatioineen

Tässä alaluvussa esitellään menetelmän kehittämisen eri vaiheet, ja niissä ilmenneet ongelmat. Menetelmää lähdettiin kehittämään iteratiivisesti, käyttötarkoitukset ja tavoitteet jatkuvasti huomioiden. Iteraatioiden tavoitteena oli kehittää menetelmää ja siitä sovellus, joka ottaa syötteenä testidataa, tulkitsee testidatan sisältöä ja tulostaa oikeiden ja väärien tulkintojen määrät. Sovelluksen sisältämä algoritmi tulisi olemaan sellainen, jonka voisi siirtää oikeaan käyttökohteeseen testien tekemisen jälkeen.

Esitutkimuksessa kerätty pilottitestin aineisto toimi ensimmäisen iteraation testiaineistona. Viimeisen iteraation jälkeen kaikki kerätty testiaineisto ajettiin sovelluksen läpi. Jokaisessa iteraatiossa testiaineisto ajettiin sovelluksen läpi ja algoritmin tuloksia tarkasteltiin sekä käsin - jokainen virhe erikseen, että visualisoimalla reitit kartan päälle. Menetelmän ongelmat ja puutteet huomioitiin iteraation päätteeksi, ja niitä yritettiin parantaa seuraavassa iteraatiossa. Iteraatioissa ei viimeistä iteraatiota lukuun ottamatta dokumentoitu onnistumisprosenttia eikä virheiden tyyppiä.

4.8.1. Ensimmäinen iteraatio

Menetelmän suunnittelu alkoi selvittämällä kävelemiseen ja raitiovaunulla matkustamiseen liittyviä piirteitä ja sitä, kuinka nämä piirteet ilmenisivät mittalaitteiden arvoina. Jalankulkijan liikkumat reitit voivat olla missä tahansa kartalla. Raitiovaunut kulkevat pitkin kiskoja, joten niiden kulkemat reitit ovat pääosin samoja aina.

Usea aikaisempi liikennevälineen tunnistamismenetelmä perustuu käyttäjän nopeuden ja kiihtyvyyden seurantaan. Nopea kiihtyminen tai hidastuminen ja keskinopeus on ominaista ajoneuvoille. Liikennevälineen päättely GPS-päivityksissä olevien nopeuksien ja kiihtyvyyksien perusteella päätettiin hylätä, koska raitiovaunuliikenne on luonteeltaan pysähtelevää ja nopeudeltaan vaihtelevaa.

Koordinaatteja vertailemalla voidaan selvittää raitiovaunujen etäisyydet käyttäjään halutulla ajanhetkellä. On todennäköistä, että käyttäjän käyttämä raitiovaunu on lähellä käyttäjää, ja ideaalitilanteessa koordinaatit olisivat lähes samat. Pelkkään etäisyyteen perustuva tunnistaminen ei tulisi olemaan tarkka, sillä on mahdollista, että usea raitiovaunu on yhtä lähellä käyttäjää, tai käyttäjä kävelee raitiovaunun ohi. Sijainteja verratessa on huomioitava, että raitiovaunun lähettämät koordinaatit vastaavat sitä pistettä, jossa raitiovaunun GPS-paikannin sijaitsee. Käyttäjän ja paikantimen ero voi olla useita metrejä, jos käyttäjä matkustaa raitiovaunun sisällä, mahdollisimman kaukana paikantimesta.

Käyttäjän ja raitiovaunujen nopeuksia vertailemalla voi havaita käyttäjän ja raitiovaunujen nopeuksien samanlaisuuden. Suunnan ja nopeuden perusteella tuotetaan

liikennevälineen ja käyttäjän liikkeen voimavektori ja näiden kahden erotuksen perusteella voidaan laskea liikkeen samankaltaisuus. Täsmälleen samanlaisten liikkeiden voimavektoreiden erotuksen pituus on 0. Virheettömien paikkatietojen kanssa liikennevälineen tunnistaminen onnistuu silloin, kun käyttäjä on liikkeessä. Paikallaan ollessa käyttäjän tunnistaminen ei ole mahdollista, sillä paikallaan olevalla käyttäjällä olisi sama liikkeen voimavektori kuin kaikilla samalla hetkellä pysähtyneenä olevilla raitiovaunuilla.

Ensimmäisen iteraation algoritmi esitettynä pseudokoodina:

```
FUNCTION get_potential_tram_from_update(latest_tram_update_list):
  VAR tram_candidate // todennäköisin raitiovaunu
  FOR (tram IN latest_tram_update_list) DO:
       // käydään läpi kaikki viimeisimmän raitiovaunupäivityksen
       // raitiovaunut. Etsitään se raitiovaunu, jonka liikkumisen
       // voimavektorin ja käyttäjän voimavektorin ero on pienin, eli joka
       // liikkuu mahdollisimman samalla tavalla kuin käyttäjä.
       IF ((tram.speed_vector – user_speed_vector) <
       (tram_candidate.speed_vector user.speed_vector)):
            tram_candidate ← tram
       ENDIF

  // lopuksi: jos raitiovaunu liikkuu tarpeeksi samalla tavalla kuin
  // käyttäjä, palauta raitiovaunun tiedot, muuten tulkitse liikkumismuoto
  // kävelyksi.
  IF (tram_candidate.speed_vector – user_speed_vector < RAJA_ARVO):
       RETURN tram_candidate
  ELSE:
       RETURN None
  ENDIF
END
```

4.8.2. Toinen iteraatio

Ensimmäisessä iteraatiossa esitelty menetelmä soveltui osittain liikennevälineen tunnistamiseen. Hitaasti liikkuvan tai paikallaan olevan käyttäjän tunnistamiseen on suunniteltava toinen tapa tunnistaa. Pysähtyneen käyttäjän tunnistamiseen tarvitaan tieto käyttäjän ja raitiovaunukandidaattien välisestä etäisyydestä.

Toinen havaittu puute menetelmässä liittyi vanhentuneiden päivitysten luotettavuuteen. Osa liikennevälineiden paikkatiedoista saattaa olla vanhentuneita, koska yhteys paikkatietopalveluun on huono tai palvelussa on vikaa. Tällöin on syytä olettaa, että liikkuva raitiovaunu ei ole enää samassa paikassa kuin se oli päivityksen lähetyshetkellä. Päivitysten luotettavuus huononee niiden vanhentuessa, ja se on huomioitava algoritmissa.

Kolmas havaittu puute liittyi tilanteisiin, joissa yksittäinen liikennevälinepäivitys tulkittiin todennäköisimmäksi kandidaatiksi, vaikka moni aikaisempi tulkinta oli antanut eri lopputuloksen. Tilanne esiintyi esimerkiksi käyttäjän seistessä paikallaan ja raitiovaunun ajaessa ohi, tai käyttäjän matkustaessa raitiovaunulla, jonka perässä tai edellä kulki toinen raitiovaunu. Ensimmäisen iteraation menetelmä ei sisältänyt suodatusta yksittäisiä poikkeamia varten. Yksinkertaisin ratkaisu tähän oli antaa enemmän painoarvoa sellaisille päivitykselle, jotka oli tulkittu aikaisemmissa päivityksissä todennäköisimmiksi kandidaateiksi.

Yllä mainitut puutteet ratkaisuineen määriteltiin liikennevälinepäivityksen piirteiksi. Liikennevälineen neljä määriteltyä piirrettä ovat seuraavat:

1. Liikennevälineen ja käyttäjän etäisyys. Lyhyempi etäisyys tulkitaan todennäköisemmäksi liikennevälineeksi
2. Käyttäjän ja liikennevälineen liikkeen samanlaisuus, eli liikkeen voimavektorien erotus. Mitä pienempi erotus, sitä yhtenevämpi kummankin liike on, ja sitä todennäköisempi liikenneväline on.
3. Käyttäjän ja liikennevälineen GPS-päivitysten aikaero. Aikaeron pieni määrä ei tee liikennevälinettä todennäköisemmäksi, mutta vanha päivitys voidaan tulkita epäluotettavaksi.
4. Liikennevälineen esiintyminen aikaisemmissa päivityksissä. On todennäköistä, että käyttäjän liikkumismuoto ei muutu usein, ja siksi aikaisempaa liikkumismuotoa voidaan pitää todennäköisenä.

Jokaiselle piirteelle määriteltiin painokerroin ja laskennallinen arvo välillä [0,1]. Painotettujen piirteiden yhteenlaskettu summa kertoo, kuinka paljon piirteet vastaavat käyttäjän tilaa. Liikennevälinepäivityksen piirteet vastaavat täysin käyttäjän tilaa arvon ollessa 1. Arvon ollessa 0 ei päivityksellä ole mitään yhteistä käyttäjän tilan kanssa.

Laskemalla jokaiselle uusimmalle raitiovaunun GPS-päivitykselle todennäköisyyden yllä mainittujen piirteiden perusteella, voidaan raitiovaunupäivityksistä päätellä todennäköisin liikenneväline, jossa käyttäjä sillä hetkellä on. Jokaiselle piirteelle annettiin aluksi mielivaltaiset painokertoimet, mutta painokertoimia muutettiin myöhemmin Pythonin Scipy-kirjaston optimize-metodin palauttamien arvojen perusteella sellaisiksi, että ne palauttivat paremmat todennäköisyydet. Kävelylle ei onnistuttu kehittämään selkeitä piirteitä, joiden perusteella kävelemisen todennäköisyyttä oltaisiin voitu arvioida

kaikissa tilanteissa. Kävelemiseksi päätettiin tulkita kaikki tilanteet, joissa yksikään liikennevälineen päivitys ei ylittänyt asetettua liikennevälineen päivityksen summan raja-arvoa.

Toisen iteraation sisältämä algoritmi pseudokoodina kuvattuna:

```
FUNCTION get_potential_tram_from_update(latest_tram_update_list):
  VAR tram_candidate_keyvalues /* lista raitiovaunukandidaateista ja niiden
  painotetuista kertoimista. */

  FOR (tram IN latest_tram_update_list) DO:
       // Lasketaan jokaisen raitiovaunun painotettu kerroin.
       VAR probability = calculate_probability(tram)
       tram.probability ← probability
  VAR highest_probablity_tram ← latest_tram_update_list.max(probability)

  // Palautetaan todennäköisin raitiovaunu, jos painotettu arvo on suurempi
  // kuin raja-arvo. Muuten ei palauteta mitään, ja päivitys tulkitaan
  // kävelynä.
  IF (highest_probability_tram > RAJA_ARVO) :
       RETURN highest_probability_tram
  ELSE:
       RETURN None
  ENDIF
END

FUNCTION calculate_probability(tram):
  /* Algoritmissa on neljä todennäköisyyden muodostavaa tekijää: liikkeen
  voimavektoreiden ero, etäisyyksien ero, aikaleimojen ero ja esiintyminen
  aikaisemmissa tulkinnoissa. Lasketaan jokaisen muuttujan painotettu arvo
  alla olevalla kaavalla. Vakio MAX_DIFFERENCE on algoritmiin kovakoodattu
  yläraja arvojen erolle. Muuttuja difference_between_user_and_tram kuvaa
  mitattavan arvon eroa raitiovaunun ja käyttäjän välillä.*/

  VAR probability = MAX_DIFFERENCE / (MAX_DIFFERENCE +
  difference_between_user_and_tram) * weight

  // Lopuksi lasketaan yhteen jokaisen painotetun todennäköisyyden
```

```
    // muodostavan osan painotettu arvo. Probability-muuttujan arvo on väliltä
    // 0…1.
    VAR probability = speed_difference + distance_diffence +
    timestamp_difference + appears_on_history_difference.
    RETURN probability
END
```

Yllä esiteltyä funktiota kutsutaan aina, kun algoritmi saa uuden GPS-päivityksen. Funktiolle annetaan parametriksi viimeisin saatu lista raitiovaunujen GPS-päivityksistä, ja funktio palauttaa parhaimman raitiovaunukandidaatin tai ei mitään. Tyhjän arvon palautuminen on tulkittava kävelynä.

4.8.3. Kolmas iteraatio

Voimavektoreiden käyttämisestä seurasi epätarkkoja tuloksia, sillä niissä ei huomioitu päivitysten aikaeroja. Voimavektorit laskettiin aluksi siten, ettei niissä huomioitu liikennevälineiden GPS-päivitysten aikaeroja suhteessa käyttäjän GPS-päivitysten aikaeroihin. Voimavektorin laskeminen muutettiin muotoon, jossa kummastakin voimavektorista vähennettiin kahden päivityksen aikaero. Näin voimavektorien pituudet olivat suhteessa samat silloin, kun päivitysten välinen aikaero oli erilainen.

Käyttäjän GPS-päivitykset ja raitiovaunujen GPS-päivitykset poikkesivat toisistaan merkittävästi, koska päivitysten koordinaattien vertailussa ei huomioitu päivitysten välistä aikaeroa. Käyttäjän GPS-päivitysten koordinaatteja muutettiin siirtämällä niitä taaksepäin vastaamaan raitiovaunun GPS-päivityksen ajanhetkeä.

Menetelmään lisättiin myös raja-arvot etäisyyksille ja voimavektorien erojen pituudelle. Raja-arvon ylittäessä piirteen todennäköisyys muutettiin nollaksi. Tämä esti selvästi liian kaukana sijaitsevien, mutta muilta piirteiltään samanlaisten raitiovaunujen esiintymisen todennäköisimpänä liikennevälineenä. Algoritmi nopeutui hieman, sillä se sai hylätä suoraan liian kaukana olevat päivitykset.

Erittäin epätarkat käyttäjän GPS-päivitykset ohitettiin ja niiden perusteella ei yritetty tehdä tulkintaa liikkumisvälineestä, sillä ne aiheuttivat lähes poikkeuksetta väärän tulkinnan. Tarkkuuden ylärajaksi asetettiin 100 metriä. Ylärajan asettaminen tarkoitti sitä, että kaikki ne päivitykset, joiden tarkkuus oli yli 100 metriä, hylättiin.

Kolmannen iteraation jälkeen tiedossa olevat puutteet olivat joko niin vähäisiä tai luonteeltaan sellaisia ettei niitä voitu ratkaista, joten testiaineisto voitiin ajaa lopullisen sovelluksen läpi.

Kolmannen iteraation algoritmi on kuvattu pseudokoodina alla:

```
FUNCTION get_potential_tram_from_update(latest_tram_update_list):
```

```
 VAR tram_candidate_keyvalues /* lista raitiovaunukandidaateista ja niiden
 painotetuista kertoimista. */

IF (user_location.accuracy > TARKKUUDEN_RAJA-ARVO):
 return None
ENDIF

 FOR (tram IN latest_tram_update_list) DO:
      // Lasketaan jokaisen raitiovaunun painotettu kerroin.
      VAR probability = calculate_probability(tram,user_location)
      tram.probability ← probability
 VAR highest_probablity_tram ← latest_tram_update_list.max(probability)

 // Palautetaan todennäköisin raitiovaunu, jos painotettu arvo on suurempi
 // kuin raja-arvo. Muuten ei palauteta mitään, ja päivitys tulkitaan
 // kävelynä.
 IF (highest_probability_tram > RAJA_ARVO) :
      RETURN highest_probability_tram
 ELSE:
      RETURN None
 ENDIF
END

FUNCTION calculate_probability(tram,user_location):
 /* Algoritmissa on neljä todennäköisyyden muodostavaa tekijää: liikkeen
 voimavektoreiden ero, etäisyyksien ero, aikaleimojen ero ja esiintyminen
 aikaisemmissa tulkinnoissa. Lasketaan jokaisen muuttujan painotettu arvo
 alla olevalla kaavalla. Vakio MAX_DIFFERENCE on algoritmiin kovakoodattu
 yläraja arvojen erolle. Muuttuja difference_between_user_and_tram kuvaa
 mitattavan arvon eroa raitiovaunun ja käyttäjän välillä.*/

 VAR converted_user_location ←
 convert_user_location_by_approximating_position(user_location.time −
 tram.time) /* palautetaan käyttäjän sijaintia "ajassa taaksepäin"
 approksimoimalla sijainti käyttäjän uusimman päivityksen ja edellisen
 päivityksen välillä piirtämällä kahden päivityksen väliin jana ja
 palautetaan janalta ne koordinaatit jotka vastaavat ajallisesti
```

```
  raitiovaunun päivitystä. Näin saadaan arvio käyttäjän sijainnista sillä
  ajanhetkellä missä hän oli raitiovaunun sijaintipäivityksen aikana. */

  VAR probability = MAX_DIFFERENCE / (MAX_DIFFERENCE +
  difference_between_user_and_tram) * weight

  IF (difference_between_user_and_tram > RAJA_ARVO):
       probability = 0  /* Jos mitattava arvo on suurempi kuin raja-arvo,
  sitä ei huomioida todennäköisyyttä laskiessa. */
  ENDIF

  /* Lopuksi lasketaan yhteen jokaisen painotetun todennäköisyyden
  muodostavan osan painotettu arvo. Probability-muuttujan arvo on väliltä
  0…1. */
  VAR probability = speed_difference + distance_diffence +
  timestamp_difference + appears_on_history_difference.
  RETURN probability
END
```

5. Koeasetelma

Tässä luvussa esitellään hypoteesi ja koeasetelma. Koeasetelmaan liittyvät testilaitteet, tiedonkeruumenetelmät ja kerättävä data muotoineen ja sisältöineen kuvataan alaluvuissa. Testidatan keräämiseen suunniteltu ja toteutettu testisovellus esitellään luvun lopussa.

5.1. Hypoteesi

Tutkimuksen tarkoituksena oli selvittää kuinka hyvin mobiililaitteen käyttäjän käyttämä liikenneväline voidaan tunnistaa reaaliaikaisesti. Tunnistamiseen käytettävän datan lähteinä toimivat vain puhelimen GPS-paikannin ja reaaliaikaista paikkatietodataa tarjoava HSL Live -palvelu. Vertailemalla näiden kahden tiedonlähteen kautta saatuja tietoja käytetty liikenneväline on mahdollista selvittää. Tutkielman hypoteesina on oletus, että on mahdollista selvittää käyttäjän liikkumismuoto reaaliaikaisesti ilman aikaisemmin kerättyä testidataa.

Hypoteesia testataan hankkimalla koehenkilöiden keräämää testidataa ja ajamalla se luvussa 4 kehitetyn algoritmin läpi ja analysoimalla algoritmin tekemät tulkinnat. Tulkintoja verrataan käyttäjän tekemiin matkakertomuksiin, joista ilmenee käytetyt liikennevälineet ja niillä matkustetut ajankohdat.

5.2. Testilaitteet

Testilaitteena käytettiin Apple iPhone 4S -matkapuhelinta. Laitteessa oli Soneran 3G-mobiilidataliittymä, jonka nopeudeksi oli ilmoitettu 10 Mb/s. IPhone 4S on vuonna 2011 julkaistu puhelin. Laitteen asetuksia oli muutettu testejä varten niin, että puheluiden soittaminen ja vastaanottaminen oli estetty. Testilaitteesta oli poistettu kaikki kolmannen osapuolen sovellukset. Valmiiksi asennetut sovellukset suljettiin ja niiden dataliikenteen ja paikkatietojen käyttäminen oli estetty. Toimenpiteiden tarkoituksena oli muiden sovellusten vaikutuksen minimointi kerättäviin mittaustuloksiin.

Testilaitteena toiminut iPhone 4S –puhelin sisältää Broadcomin valmistaman MDM6610 –GPS-paikantimen. Paikannin käyttää sekä GPS-paikannusta että venäläistä GLONASS (ven. GLObalnaya NAvigatsionnaya Sputnikovaya Sistema) –satelliittipaikannusta. IPhone 4S:n paikannuksen toimintaperiaatetta ei ole dokumentoitu Applen toimesta, mutta MacWorld-lehden arvion (MacWorld, 2013) mukaan paikannus tapahtuu aluksi puhelinmastojen ja WLAN-tukiasematietojen perusteella, ja sen jälkeen GPS-satelliittipaikannuksen perusteella.

5.3. Testisovellus

Testejä varten toteutettiin testihenkilöiden sijainnin nauhoittanut mobiilisovellus. Sovellus kirjoitettiin Objective-C-kielellä ja käyttämällä iOS 6 SDK:ta. Sovellus koostuu neljästä erillisestä osasta:

1. Käyttäjän sijaintia seuraava luokka. Luokka kuuntelee CoreLocation-paikkatietokirjaston CLLocationManagerDelegate-kuuntelijan lähettämiä päivityksiä sijainnin muutoksista, ja lähettää ne eteenpäin tiedostoja kirjoittavalle luokalle.
2. Raitiovaunujen sijaintia seuraava luokka. Luokka avaa yhteyden HSL Live –rajapintaan ja kuuntelee sieltä saapuvia päivityksiä raitiovaunujen sijainneista. Luokka kerää nauhoittamisen aikana kaikki HSL Live –rajapinnan lähettämät raitiovaunupäivitykset, lisää niihin aikaleimat ja tallentaa ne tiedostoon. Tuloksia analysoidessa algoritmi voi jäljitellä reaaliaikaista tilannetta lukemalla tiedoston päivityksiä.
3. Kahden edellisen luokan päivityksiä tiedostoon tallentava luokka. Luokan tehtävänä oli luoda ja täyttää kolme tiedostoa: käyttäjän GPS-päivityksiä sisältävä tiedosto, raitiovaunujen paikkatietojen päivityksiä sisältävä tiedosto ja käyttäjän itse merkitsemiä päivityksiä sisältävä tiedosto. Tiedostojen sisältö tallennetaan CSV-muodossa.
4. Käyttöliittymäkerros, jolla datan kerääminen aloitetaan ja lopetetaan (Kuva 6). Käyttöliittymä sisälsi painikkeet nauhoittamisen aloittamiseen ja lopettamiseen. Nauhoittamisen aikana näytöllä oli näkyvissä “Enter / exit tram” –painike, jota painamalla testisovellus tallensi aikaleimalla varustetun tapahtuman laitteen lokitiedostoon.

Kuva 6. Aineiston keräämiseen käytetty sovellus. Vasemmanpuoleinen näyttökuva näyttää sovelluksen tilan ennen nauhoittamista ja oikeanpuoleinen nauhoituksen aikana.

Sovelluksen käyttöliittymä sisältää kaksi erillistä näkymää (Kuva 6). Ensimmäinen näkymä näkyy koehenkilölle ennen aineiston keräämistä. Siinä on yksi painike, jossa lukee "Start logging". Painikkeen painallus käynnistää testin ja rupeaa nauhoittamaan GPS-paikantimen keräämiä sijaintipäivityksiä. Testin aikana näkymä vaihtuu toiseen näkymään jossa on näkyvillä kaksi painiketta, nimiltään "End logging" ja "Enter / exit tram". Ensimmäisenä mainittu painike päättää nauhoituksen ja jälkimmäistä painamalla käyttäjä voi ilmoittaa nousevansa raitiovaunuun tai poistuvansa raitiovaunusta. Testisovellus kirjoittaa silloin tiedostoon aikaleimalla varustetun merkin käyttäjän liikkumistyypin muuttumisesta.

Nauhoituksen ollessa päällä sovellus tallentaa puhelimen GPS-päivityksen lisäksi HSL Live –rajapinnan lähettämät päivitykset raitiovaunujen paikkatiedoista. Raitiovaunujen paikkatietojen päivitykset olisi voitu tallentaa erilliselle palvelimelle, joka olisi kerännyt jatkuvasti päivityksiä. Palvelimen hyödyntäminen päivitysten keräämisessä ei ollut mahdollista testien suorittamisen aikana käytettävien resurssien vähyyden takia, ja siksi raitiovaunujen sijaintipäivitykset päädyttiin tallentamaan laitteelle nauhoituskohtaisesti.

Kaikki tieto tallennettiin CSV-muodossa (engl. Comma-separated values, pilkulla erotettuja arvoja) helpon automaattisen analysoinnin varmistamiseksi. HSL Live –

rajapinnan lähettämä data oli valmiiksi CSV-muotoista, mutta puhelimen GPS-päivitykset piti muuttaa oikean muotoisiksi. Taulukossa Taulukko 4. GPS-paikkatietopäivitysten sisältämät kentätkuvataan yksittäisen GPS-päivityksen sisältämät tiedot.

Kenttä	Kuvaus	Esimerkkiarvo
1	Aikaleima UNIX-formaatissa, sekunteja päivämäärästä 1.1.1970	1353261431.241914
2	Leveys asteina	60.168718
3	Pituus asteina	24.931681
4	Suunta asteina	-1.000000
5	Tarkkuus leveyssuunnassa (m)	65.000000
6	Tarkkuus pystysuunnassa (m)	10.000000
7	Nopeus (m/s)	-1.000000
8	Korkeus merenpinnasta (m)	30.000000

Taulukko 4. GPS-paikkatietopäivitysten sisältämät kentät

HSL Live –palvelun dokumentaatio (Mattersoft, 2015) määrittelee raitiovaunun päivityksen sisältämät 17 erilaista kenttää, joita päivitysten merkkijonossa voi esiintyä. Testidatan keräämisen aikana päivitykset sisälsivät vain 15 kenttää (Taulukko 5), ajoneuvon nimen ja IP-osoitteen puuttuessa kaikista päivityksistä. Kenttä nimeltä "gpsTimeDifference" sai aina arvon 0.0.

Kenttä	Kuvaus	Esimerkkiarvo	Tyyppi, desimaalien määrä	Käytössä
id	Vehicle id	CEENG1074300245	String	X
name	Name of the vehicle	PL / 811	String	
type	0 = tram	0	bit	X
ip	Vehicle ip-address	127.0.0.1	String	
lat	Vehicle Latitude WGS84 coordinate with four decimals	60.2244	Double, DD.dddd/DDD.dddd	X
lng	Vehicle Longitude WGS84 coordinate with four decimals	24.9064	Double, DD.dddd/DDD.dddd	X
speed	Vehicle speed km/h	42.74	Double, 0.00	X
bearing	Vehicle bearing in degrees	225	Double, 0.00	X
acceleration	Vehicle acceleration based on the two last speed measurements m/s^2	1.65	Double, 0.00	X
gpsTimeDifference	Difference between time stamp of vehicle device and Live!-system in milliseconds when message is received by Live!-server	-28.11*1008	Double, 0.00	(aina 0.0)
route	Route id	1062	String	X

direction	Direction on route	1	Int	X
departure	Route departure identification	1045	String	X
departureTime	Time of departure	03092008111504	DateTime (ddMMyyyyHHmmss)	X
departureStartsIn	Time to departure in seconds. After departure, value is 0	20	Int	X
distanceFromStart	Distance form route starting point in meters	500	Double	X
snappedLat	Latitude coordination point snapped for route	60.2350	Double, DD.dddd/DDD.dddd	X
snappedLng	Longitude coordination point snapped for route	24.9100	Double, DD.dddd/DDD.dddd	X
snappedBearing	Vehicle bearing snapped for route.	225	Double	X
nextStopIndex	Index number of next stop	1	Int	X
onStop	Information on vehicle location(on stop/not on stop) 1=on stop, 0= not on stop	0	Boolean	X
differenceFromTimetable	Difference between timetable and the last stop passing in seconds. Positive value= before scheduled, negative value= late from schedule.	12	Int	X

Taulukko 5. HSL Live –rajapinnan lähettämät liikennevälinetiedot rajapintadokumentaation mukaan.

Esimerkkirivi HSL Live –rajapinnan lähettämästä raitiovaunudatasta näytti alla olevalta merkkijonolta:

```
1353261432.232838;RHKL00225;;1;;60.187752;24.962461;0.00;231;0.00;0.00;
1007A;2;1853;18112012185300;0;2898;60.1877;24.9624;0;16;0;-382
```

Ennen kuin aineistoa alettiin kerätä ja algoritmia suunnitella, tarjosi HSL Live –rajapinta reaaliaikaiset paikkatiedot kaikista raitiovaunuista, mutta vain osasta Helsingin sisäisistä busseista. Metrojen ja Suomenlinnaan kulkevien lauttojen sijainteja ei ollut palvelussa tutkielman kirjoittamisen alkuvaiheessa. Aineiston keräämisen aikana rajapinnan kautta ruvettiin toimittamaan paikkatietoja myös ajossa olevista metroista ja useammista busseista. Tämän havaitsemisen jälkeen kaikki liikennevälineiden päivitykset, jotka olivat muuta muotoa kuin 0 (raitiovaunu), jätettiin käsittelemättä. Metrojen mukaan ottamista harkittiin aineistoa käsitellessä, mutta metro liikennevälineenä päätettiin olla ottamatta mukaan. Syynä tähän oli se, että GPS-paikannus toimi erittäin huonosti metrotunneleissa – tarkkuus saattoi huonoimmillaan olla useita kilometrejä. Metrolla ja metrotunneleissa liikkumisessa tapahtuvat liikkumismuodon päivitykset ovat luonteeltaan erilaisia, ja GPS-paikantimen sijaintiin perustuvat tunnistamismenetelmät

toimivat huonosti. Metrossa liikkumisessa on tärkeämpää tunnistaa seuraava asema ja asemalle saapumiseen kuluva aika. Se onnistuu paremmin esimerkiksi kiihtyvyyssensorin avulla ja matkapuhelimen antennin signaalin voimakkuutta tarkkailemalla (Yu ja muut, 2013).

5.4. Aineiston kerääminen

Testiaineisto kerättiin Helsingin kantakaupungissa. Aineiston keräämisen suoritti kaksi koehenkilöä valittuina tutkielman kirjoittajan tuttavapiiristä. Koehenkilönä toimimisen edellytyksenä oli raitiovaunun käyttäminen joko työmatkoilla tai vapaa-aikana ja lisäksi normaali havainnointi- ja liikkumiskyky. Koehenkilönä toimimisen ehtona oli myös se, että koehenkilöt liikkuivat testien ajan vain jalan tai raitiovaunulla. Aineisto kerättiin aikavälillä 26.11.2012 – 16.2.2014. Aineisto jaettiin kahteen ryhmään mittauksen sisällön perusteella: ensimmäiseen ryhmään kuuluva aineisto sisälsi pelkkää kävelyä ja toiseen ryhmään kuului sekä kävelyä että raitiovaunun käyttöä. Koska testiympäristöä ei voinut eristää koetta varten, kerättiin testidataa käyttäjien normaaleilta työ- ja vapaa-ajan matkoilta. Koehenkilöitä pyydettiin piirtämään kartalle heidän kulkemansa reitit sekä paikat, joissa he nousivat raitiovaunuihin ja poistuivat raitiovaunuista. Testiaineistot, joiden sisältämiä tapahtumia testihenkilöt eivät muistaneet, poistettiin. Testien aikana osa raitiovaunuista saattoi ajaa poikkeusreittiä, mutta sillä ei katsottu olevan merkitystä mittaustuloksiin.

Yksittäiset koehenkilöiden tekemät nauhoitukset olivat kooltaan 238 kilobitin ja 11,1 megabitin väliltä. Yhtä mobiililaitteen GPS-päivitystä kohden nauhoitusdatassa oli noin 70 raitiovaunupäivitystä, eli algoritmi joutui päättelemään liikkumismuodon näistä 70 päivityksestä. Osasta sessioiden sisältämästä GPS-datasta puuttui käyttäjän käyttämän raitiovaunun lähettämät GPS-päivitykset. Näiden sessioiden aineisto hylättiin, sillä niissä ei ollut mitään mihin verrata käyttäjän GPS-päivityksiä.

CoreLocation-kirjasto lähetti päivityksiä käyttäjän sijainnista noin kerran sekunnissa. Intervallia ei ollut mahdollista muuttaa nopeammaksi tai hitaammaksi. Myös HSL Live –rajapinta lähetti socket-rajapinnan yli päivityksiä noin kerran sekunnissa, eikä palvelussa ollut mahdollista muuttaa päivitysten tiheyttä. Osassa aineiston nauhoituksissa testiapplikaatio ei ollut saanut yhteyttä HSL Live –palveluun, jolloin nauhoitukset hylättiin. HSL Live –rajapinnan käytössä ilmeni muitakin ongelmia, kuten luvussa 4 mainittu metrojen ja bussien lisääminen paikkatietodataan ja virheilmoitusten täydellinen puuttuminen rajapinnan dataliikenteestä.

5.5. Aineiston käsittely

Aineisto analysoitiin ajamalla se luvussa 4 kuvatun algoritmin läpi. Koehenkilöiden kaikki nauhoitukset ajettiin algoritmin läpi ja tulokset kerättiin yhteen. Koska jokaisessa CSV-

rivissä oli aikaleima, oli mahdollista simuloida algoritmin toimintaa niin, että se olisi saanut päivitykset reaaliaikaisesti.

Algoritmin käyttämien parametrien sopivia arvoja ei tiedetty ennen aineiston käsittelyä, ja niiden säätäminen käsin olisi ollut liian työlästä. Siksi algoritmin parametrien ja raja-arvojen arvot asetettiin aluksi mielivaltaisesti. Arvot muutettiin ajamalla Python-kielen Scipy-kirjaston minimize-optimointifunktiota (Scipy, 2013) testialgoritmille useita satoja iteraarioita. Optimointifunktiona käytettiin fmin_tnc-funktiota parametreilla *{bounds = bounds, approx_grad = 1, epsilon = 0.05}*. Minimize-funktio ajoi algoritmia haarukoiden eri parametreilla, ja palautti lopuksi parhaimpaan mittaustulokseen johtavat raja-arvot.

6. Tulokset

Tässä luvussa esitellään testimittauksissa kerätyn aineiston pohjalta tehtyjen analyysien tulokset. Alaluvuissa esitellään analyysien tulokset onnistumisprosentteineen, epäonnistuneiden mittausten esiintymiset ja tyypit, sekä mittaustulokset kun sekä liikennevälineen että käyttäjän sijaintitietopäivityksiä vähennettiin. Analyysien tulokset virheineen on esitelty tarkemmin taulukoissa ja kuvaajissa. Tuloksia analysoimalla selvisi, että suunniteltu algoritmi erotti kummatkin määritellyt liikkumismuodot hyvin toisistaan testiaineiston ja –ympäristön puitteissa.

Jokaisen tulkinnan onnistuminen selvitettiin vertaamalla tulkintaa käyttäjän tekemiin merkintöihin liikkumismuodon muuttumisesta. Mikäli tulkinnan liikkumisväline ja –muoto oli sama kuin käyttäjän tekemissä merkinnöissä, tulkinta määriteltiin onnistuneeksi, eli algoritmi onnistui tunnistamaan käyttäjän liikkumismuodon haluttuna ajanhetkenä. Muut tulkinnat määriteltiin vääriksi tulkinnoiksi ja ne jaettiin kahteen eri virheellisten tulkintojen ryhmään.

6.1. Onnistuneet tulkinnat

Pelkkää kävelyä sisältävät mittaukset tunnistettiin onnistuneesti 97,01-prosenttisesti, ja kävelemistä ja raitiovaunulla matkustamista sisältävät mittaukset tunnistettiin 92,62-prosenttisesti oikein (Taulukko 6).

Ryhmä	**Mittausten määrä**	**Onnistuneiden tulkintojen määrä / %**	**Käyttäjän GPS-päivitysten määrä**
1. Pelkkää kävelyä	16	97,01	4650
2. Kävelyä ja raitiovaunulla liikkumista sekaisin	2	92,62	1193
Yhteensä	**18**	**96,18**	**5843**

Taulukko 6. Testiaineiston määrä ja onnistuneiden tulkintojen määrä tyypeittäin.

Verrattuna Stennethin ja muiden (2011) tekemään tutkimukseen automaattisesta liikennevälineen tunnistamisesta, oikean liikkumistyypin erottamisen onnistumisen erot olivat lähes samat – Stennethin ja muiden tutkimuksessa 93,5% ja tutkimuksessa esitetyssä menetelmässä 92,6%. Tulosten vertailu ei kuitenkaan ole mielekästä, sillä Stennethin tutkimuksessa tunnistettavia liikkumismuotoja oli kuusi – käveleminen, bussi, auto, juna, polkupyörä ja paikallaan oleminen, ja tässä tutkimuksessa tavoitteena oli tunnistaa oikea yksilöity liikenneväline raitiovaunujen joukosta.

6.2. Väärät tulkinnat

Algoritmi laski kaikki väärät tulkinnat jakaen ne samalla kolmeen alla määriteltyyn virhetyyppiin:

1. **Kävely tunnistettiin raitiovaunussa matkustamiseksi.** Algoritmi tulkitsi käyttäjän tilan raitiovaunussa matkustamiseksi, vaikka käyttäjä oli jalan tulkinnan hetkellä. Pelkkää kävelyä sisältävissä mittauksissa tämä virhetyyppi oli ainoa esiintynyt virhe.
2. **Raitiovaunussa matkustaminen tunnistettiin kävelyksi.** Algoritmi tulkitsi käyttäjän liikkuvan jalan, vaikka käyttäjä matkusti raitiovaunulla mittaushetkellä.
3. **Raitiovaunu tunnistettiin vääräksi raitiovaunuksi.** Algoritmi tulkitsi väärän raitiovaunun käyttäjän käyttämäksi liikennevälineeksi. Tämä virhetyyppi oli harvinaisempi (8.14 % virheistä ryhmässä 2), ja yleensä sitä esiintyi vain yksittäisenä virhetulkintana.

Virheiden luokittelu kertoo tavan, jolla algoritmi tulkitsi tilanteen vääräksi, mutta ei kertonut syytä, miksi algoritmi teki virheellisen tulkinnan.

Virhetyyppi	Esiintyminen	Osuus kaikista virheistä
1 (kävely raitiovaununa)	29	33,72%
2 (raitiovaunu kävelynä)	50	58,14%
3 (väärä raitiovaunu)	7	8,14%

Taulukko 7. Virheiden jakauma mittauksissa, joissa oli sekä kävelyä että raitiovaunulla matkustamista.

Yleisin virhetyyppi oli tyyppi 2 (Taulukko 7), eli virhe jossa raitiovaunu tunnistettiin kävelemisenä. Vääriä raitiovaunujen tunnistamisia oli virhetyyppinä vähiten. Pelkkää kävelyä sisältävissä mittauksissa kaikki virheet olivat oletetusti tyyppiä 1, eli kävely tulkittiin raitiovaunulla matkustamisena. Esiintymiä oli 76 kappaletta. Mikäli pelkkää kävelyä sisältävissä mittauksissa olisi ollut tyypin 2 tai 3 virheitä, olisi algoritmissa tai koeasetelmassa ollut vakava virhe. Näin ei kuitenkaan tapahtunut.

Virhetyyppi	Nopeus	Etäisyys	Aikaero	Esiintyminen
1 (kävely raitiovaununa)	0,85	0,69	0,87	0,90
2 (raitiovaunu kävelynä)	0,13	0,36	0,80	0,43
3 (väärä raitiovaunu)	0,83	0,67	0,86	0,51

Taulukko 8. Väärien tulkintojen painotusten keskiarvot sekä kävelyä että raitiovaunulla matkustamista sisältävistä mittauksissa.

Virheiden jakaminen kolmeen ryhmään auttaa havaitsemaan miten algoritmi tulkitsee tilanteet väärin, mutta ei kerro algoritmien väärien tulkintojen syytä. Algoritmi tallensi

lokitiedostoon kaikkien väärien tulkintojen sisältämien painotusten keskiarvot (Taulukko 8).

Virhetyypin 1 virheet näyttävät johtuvan eniten todennäköisimmän raitiovaunun ja käyttäjän samanlaisista nopeuden voimavektoreista etäisyyden ollessa vähemmän merkittävänä tekijänä. Virhetyypin 2 virheiden syynä on käyttäjän ja todennäköisimmän liikennevälineen erilaiset nopeusvektorit ja näiden kahden suuret etäisyydet. Esiintyminen aikaisemmissa tulkinnoissa (Taulukko 8, viimeinen sarake) on huomattavan alhainen, 0,428, mikä voi tarkoittaa sitä, että tulkinta on täysin erilainen aikaisempiin, todennäköisesti oikeisiin, tulkintoihin verrattuna.

Virhetyyppi	**Nopeus**	**Etäisyys**	**Aikaero**	**Esiintyminen**
1 (kävely raitiovaununa)	0,83	0,67	0,62	0,80
2 (raitiovaunu kävelynä)	-	-	-	-
3 (väärä raitiovaunu)	-	-	-	-

Taulukko 9. Väärien tulkintojen painotusten keskiarvot pelkkää kävelyä sisältävistä mittauksissa.

Pelkkää kävelyä sisältävien mittausten virheelliset tulkinnat sijoittuivat kaikki odotetusti ryhmään 1 (Taulukko 9). Etäisyys ja aikaerot olivat suuria virheellisten tulkintojen painotuksissa. Suuri ero etäisyydessä voi johtua epätarkoista matkapuhelimen GPS-paikannuksista ja epätarkoista raitiovaunun paikkatiedoista.

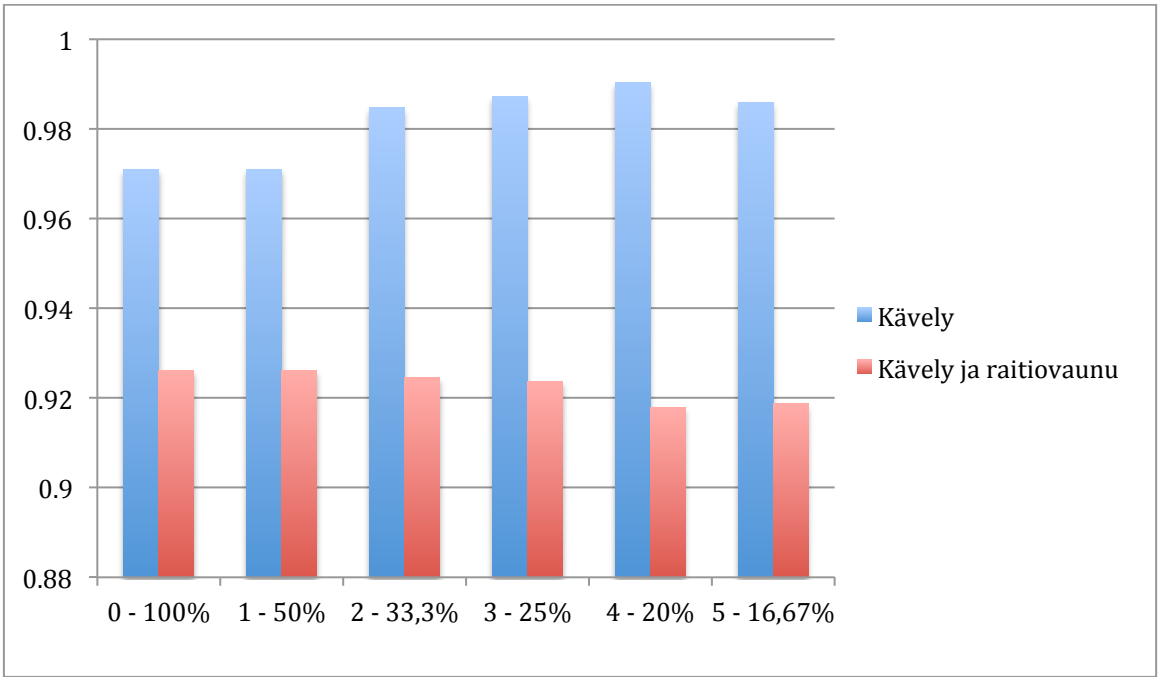
Virheiden aikaero raitiovaunuun nousemisessa ja raitiovaunusta poistumisessa oli keskiarvoltaan 179,1 sekuntia ja keskihajonta oli 116,9 sekuntia. Sen perusteella virheet eivät keskity liikkumistyypin vaihtamiseen, kuten aikaisemmin oletettiin, vaan virheet sijoittuivat pitkin matkaa.

Sekä kävelyä että raitiovaunulla matkustamista sisältävissä mittauksissa virheellisissä mittauksissa käyttäjää lähin liikenneväline olisi ollut 88,40% tilanteista oikea liikenneväline. Tämä viittaa siihen, että tulkinnoissa mitattavien piirteiden painokertoimia ja raitiovaunuksi tunnistamiseen vaadittavaa raja-arvoa pitäisi muuttaa.

6.3. Paikkatietopäivitysten tiheyden vaikutus tuloksiin

Testimittauksissa sekä yhteys HSL Live –palveluun, että testilaitteen GPS-paikannin toimivat ongelmitta. Matkapuhelimen paikkatietopalvelu ei palauttanut virheitä testiaineiston keräämisen aikana. Matkapuhelimen GPS-päivitysten koordinaatit eivät myöskään sisältäneet arvoja, jotka olisivat vaikuttaneet virheellisiltä. On mahdollista, että todellisessa käyttötilanteessa joko matkapuhelimen GPS-paikannin tai liikennevälineiden paikkatietoja tarjoava palvelu ei saa kaikissa tilanteissa päivityksiä niin usein kuin testitilanteessa. Yhteys HSL Live –palveluun voi katketa, tai palvelu voi jättää osan päivityksistä lähettämättä. Matkapuhelimen GPS-paikannin ei aina tee uusia sijaintipäivityksiä esimerkiksi huonon sijainnin takia. Näiden kahden tiedon osittaisen

puuttumisen vaikutus selvitettiin analysoimalla mittausaineisto niin, että toisesta lähteestä oli poistettu osa päivityksistä.



Kuvaaja 1. Raitiovaunujen paikkatietojen vähentämisen vaikutukset tuloksiin.

Sekä raitiovaunua että kävelyä sisältävien mittausten tunnistamistulosteet huononivat sitä mukaa kun raitiovaunupäivityksiä suodatettiin pois. Pienimmillään tulokset olivat silloin kun 80% raitiovaunupäivityksistä suodatettiin pois, tuloksen ollessa silloin 92%. Tulos on 0,91% pienempi kuin ilman raitiovaunupäivitysten suodattamista. Kävelyn tunnistaminen parani raitiovaunupäivityksiä suodattamalla, parhaimmillaan 2,0% suodattamattomaan verrattuna. (Kuvaaja 1)

GPS-päivitysten suodattamisella ei ollut vaikutusta algoritmin toiminnalle. Syynä tähän on se, että algoritmi tulkitsee käyttäjän tilaa aina uuden GPS-päivityksen saapuessa. Jos päivitysten intervalli harvenee, algoritmi tekee tulkintoja harvemmin. Käyttäjälle se näkyy liikkumismuodon tilan päivittymisen hidastumisena. Todennäköisin tilanne, jossa harvemmalla intervallilla GPS-päivityksissä on merkitystä, on ajoneuvoon nouseminen ja ajoneuvosta poistuminen. Tuloksien perusteella algoritmin päivitysintervallia voidaan harventaa rajattomasti matkapuhelimen akun ja prosessorin kuormituksen säästämiseksi. Sovelluksen käyttäjälle harvempi intervalli näkyy harvemmin päivittyvän käyttöliittymän muodossa.

6.4. Pohdintaa

Tutkimuksen rajallisten resurssien takia testiaineiston sisältö oli nauhoitettu pääosin samoilta raitiovaunu- ja kävelyreiteiltä Helsingin kantakaupungista. Yksipuolisten

mittausten perusteella kerätty data ja johtopäätökset eivät vastaa todellisia käyttötapauksia niin hyvin kuin monipuolisemmat ja vaihtelevammat mittaukset. Koeasetelma on kuitenkin toistettavissa kaupungissa, jonka liikennevälineistä on tarjolla SIRI-protokollan mukaista reaaliaikaista paikkatietodataa. Testiaineisto oli kooltaan huomattavasti suppeampi kuin muissa luvussa 2 viitatuissa liikennevälineen tunnistamista koskevissa tutkimuksissa.

Algoritmin erääksi puutteeksi osoittautui kyvyttömyys tunnistaa tilanteet, joissa käyttäjä nousee raitiovaunuun tai poistuu raitiovaunusta. Niiden virheiden määrää, jotka aiheutuvat liikkumismuodon vaihtumisesta ei voitu selvittää. Virheiden piirteitä selvittämällä ei löytynyt ratkaisua ongelmaan. Algoritmia tuskin on mahdollista saada tunnistamaan liikennevälineen muutoksen nopeasti ja tarkasti raja-arvoja ja painotuksia muuttamalla. Liikkumismuodon muuttumisen tunnistaminen olisi tapahduttava muulla menetelmällä. Kiihtyvyyssensorin käyttäminen tähän tarkoitukseen voisi olla mahdollinen ratkaisu.

Automaattisen liikennevälineen tunnistamismenetelmän soveltuvuutta muiden liikennevälineiden kuin raitiovaunun tunnistamiseen ei selvitetty. On mahdollista, että menetelmä toimii junien, bussien ja lauttojen kanssa, mutta metrojen tunnistaminen tulee olemaan vaikeaa. Maan alla GPS-paikannus on epätarkkaa, tehden sijaintiin perustuvan tunnistamisen epäluotettavaksi. Tutkielmassa esiteltyä menetelmää voisi käyttää rinnakkain yhdessä jonkin toisen automaattisen liikkumisen tunnistavan menetelmän, esimerkiksi Google Locationin tai Applen Core Motionin, kanssa siten, että toinen menetelmä tunnistaa liikkumismuodon ja tutkielmassa esitelty liikennevälineen tunnistamismenetelmä yksilöi käytetyn liikennevälineen.

Menetelmän yhdeksi ilmeiseksi heikkoudeksi osoittautui sen perustuminen liikaa paikkatietopäivitysten koordinaattien välisten etäisyyksien laskemiseen. Koordinaatit olivat testien aikana usein epätarkkoja. Epätarkkojen koordinaattien käyttäminen aiheuttaa virheellisiä tulkintoja, sillä menetelmässä on annettu suuri painoarvo käyttäjän ja liikennevälineen sijainnille. Menetelmä ei myöskään huomioinut raitiovaunun keskinopeutta tai –kiihtyvyyttä, josta olisi apua eri liikennevälinetyyppien erottamisessa toisistaan. GPS-päivitysten koordinaattien suodattaminen esim. Kalman-suotimella, olisi voinut auttaa epätarkkojen sijaintien aiheuttamia virhetulkintoja.

Kehitetty menetelmä ei ottanut huomioon raitiovaunujen GPS-lähettimen sijaintia raitiovaunuissa, eikä huomioinut käyttäjän sijaintia raitiovaunun sisällä. Helsingissä käytössä olevat raitiovaunut ovat pituudeltaan 20,1 metrin ja 27,6 metrin väliltä, joten käyttäjän ja raitiovaunun välinen etäisyys voi tulla arvioiduksi yli 20 metriksi käyttäjän ollessa sisällä raitiovaunun takaosassa, aiheuttaen vääriä tulkintoja. Ongelmaan tuskin on helppoa ratkaisua, sillä GPS-lähettimien sijainti voi vaihdella raitiovaunuissa, ja käyttäjän

sijaintia raitiovaunun sisällä ei voida päätellä. Tämä epätarkkuus puoltaa osaltaan koordinaattien merkityksen vähentämistä algoritmin jatkokehityksessä.

Tutkielmassa esitelty menetelmä hyödynsi mahdollisimman vähän erilaisia tietoja, joita HSL Live –palvelu tarjosi. Valinta oli tietoinen, koska tavoitteena oli suunnitella menetelmä, jota voi hyödyntää mahdollisimman hyvin muissakin reaaliaikaisia paikkatietoja lähettävässä järjestelmässä. Yksittäistä paikkatietopalvelua käyttävää sovellusta suunnitellessa kannattaa hyödyntää kaikki tarjottava tieto. HSL Live –rajapinnan tarjoamista *bearing*, *acceleration*, *snappedLat*, *snappedLng* ja *onStop* –kentistä, eli raitiovaunun kompassisuunnasta, kiihtyvyydestä, sijainnista kiskoilla ja tiedosta onko raitiovaunu pysäkillä, olisi saanut hyödyllistä tietoa liikennevälineen tunnistamista tehdessä.

Sen sijaan että menetelmän tekemä tulkinta olisi absoluuttinen joko/tai -jaottelu kävelemisen ja raitiovaunun välillä, voisi se sisältää tiedon, joka ilmaisisi tulkinnan varmuutta. Menetelmää käyttävät sovellukset voisivat määrittää itse sopivat raja-arvot ja jättää epävarmat tulkinnat hyödyntämättä, tulkiten ne kohinana.

Tuloksien perusteella algoritmin paketoiminen kirjastoksi mahdollistaisi nopean ja helpon tavan saada reaaliaikainen tieto käyttäjän matkustustavasta. Kirjasto tarvitsee kaksi ulkopuolista syötettä: tarvittavan määrän uusia GPS-päivityksiä sovelluksen laitteen toimittamana ja verkko-osoitteen, josta löytyy halutun alueen liikennevälineiden sijainnit.

Algoritmin sisältämä kirjasto ei tulisi olemaan täysin universaali siinä mielessä, että sitä voisi käyttää kaikissa kaupungeissa konfiguroimatta, vaan se pitäisi konfiguroida jokaista kaupunkia tai maata varten erikseen. Tämä tuskin olisi ongelma, sillä reittioppaat on usein räätälöity toimimaan yhden kaupungin tai metropolialueen sisällä.

7. Yhteenveto

Tutkielmassa suunniteltiin ja kehitettiin iteroiden alustariippumaton menetelmä automaattiseen liikennevälineen tunnistamiseen. Menetelmän tarkkuutta arvioitiin koehenkilöiden liikkumisista kerätyllä aineistolla. Menetelmä onnistui testiaineiston puitteissa tunnistamaan käyttäjän liikkumismuodon oikein 96,18% tilanteissa. Kaikkien virheellisten tulkintojen syitä ei onnistuttu tulkitsemaan.

Kehitetty algoritmi ei sellaisenaan vielä sovi aiottuihin käyttökohteisiin, kuten reittiopassovelluksen osaksi kertomaan, koska käyttäjä matkustaa liikennevälineellä ja koska hän kävelee, sillä liikkumismuotojen muuttumisen tunnistaminen ei tapahdu välittömästi eikä tarkasti. Algoritmi kuitenkin täytti muut sille asetetut vaatimukset, eli osoittautui teknisesti mahdolliseksi toteuttaa algoritmi alustariippumattomana.

Liikennevälineen tunnistaminen, ja kontekstin päättely yleisesti, osoittautui alun perin arvioitua monimutkaisemmaksi ongelmaksi. Aiheen tutkiminen paljasti jatkuvasti uusia muuttujia ympäristössä ja mahdollisia poikkeustilanteita, jotka on huomioitava käyttäjän tilaa tarkkaillessa. Ympäristön täydellinen mallintaminen ja tilan absoluuttinen tulkitseminen ei ole mielekästä, mutta luvussa 4 kuvailtujen iteraatioiden tavalla menetelmää olisi mahdollista parantaa jatkuvasti.

Algoritmin kehittämisen iteraatioittain ja jokaisen iteraation testaaminen aikaisemmin kerätyllä harjoitteludatalla osoittautui hyödylliseksi suunnittelumenetelmäksi. Menetelmän kehittämisessä sekä tulosten kvantitatiivinen että kvalitatiivinen arviointi osoittautui välttämättömäksi. Tutkimusaineiston ajaminen algoritmin läpi antoi onnistuneiden tulkintojen määrän, mutta ei selvittänyt väärien tulkintojen syitä.

Algoritmi onnistuttiin suunnittelemaan ja kehittämään arkkitehtuuriltaan sellaiseksi, että se voidaan asentaa olemassa olevaan sovellukseen kirjastona tai se voidaan ottaa käyttöön palvelinrajapintana. Silloin algoritmi olisi täysin alustariippumaton.

Lähdeluettelo

(Alvarez ja muut, 2006) Diego Alvarez, Rafael C. González, Antonio López ja Juan C. Alvarez. Comparison of Step Length Estimators from Weareable Accelerometer Devices. Proceedings of the 28th IEEE EMBS Annual International Conference New York City, USA, Aug 30-Sept 3, 2006

(AnandTech, 2013) Apple iPhone 4S: Thoroughly Revieved. Tarkastettu 17.4.2013.

(Android Developers, 2013) http://developer.android.com/reference/android/location/Location. Tarkastettu 2.12.2013.

(Apple, 2013) Core Motion Framework Reference. https://developer.apple.com/library/ios/documentation/CoreMotion/Reference/CoreMotion_Reference/_index.html Tarkastettu 23.9.2013.

(Apple, 2014a) CLLocation Class Reference. https://developer.apple.com/library/ios/documentation/CoreLocation/Reference/CLLocation_Class/index.html#//apple_ref/occ/instp/CLLocation/speed. Tarkastettu 30.9.2014

(Atallah ja muut, 2010) Sensor Placement for Activity Detection using Wearable Accelerometers. Body Sensor Networks (BSN), 2010 International Conference on.

(Choudhury ja muut, 2008) Choudhury, T., Borriello, G., Consolvo, S., Haehnel, D., Harrison, B., Hemingway, B., Hightower, J., Klasnja, P., Koscher, K., LaMarca, A., Landay, J. A., LeGrand, L., Lester, J., Rahimi, A., Rea, A., ja Wyatt, D. 2008. The mobile sensing platform: An embedded activity recognition system. Pervasive Computing 7, 2, 32–41.

(Choujaa ja Dulay, 2009) Activity Recognition from Mobile Phone Data: State of the Art, Prospects and Open Problems.

(Google, 2014) Location APIs | Android Developers. https://developer.android.com/google/play-services/location.html tarkastettu 24.9.2014.

(GPS Visualizer, 2013) GPS Visualizer. http://www.gpsvisualizer.com Tarkastettu 22.3.2015.

(GTFS, 2015) GTFS-realtime reference. https://developers.google.com/transit/gtfs-realtime/reference#Position Tarkastettu 22.2.2015.

(HKL, 2012) HSL. http://www.hel.fi/hki/hkl/fi/HKL-Raitioliikenne HKL-Raitioliikenne. Tarkastettu 20.12.2012.

(HSL, 2014) HSL Developer Community. http://dev.hsl.fi Tarkastettu 30.9.2014

(ITS, 2014) Junat kartalla API. ITS Factory Wiki. http://wiki.itsfactory.fi/index.php/Junat_kartalla_API tarkastettu 29.9.2014.

(Kwapisz ja muut, 2010) Activity Recognition using Cell Phone Accelerometers

(Liao, Fox, Kautz, 2004) L. Liao, D. Fox, H. Kautz. Learning and Inferring Transportation Routines. AAAI 2004

(Macworld, 2013) Macworld. http://www.macworld.com/article/1159528/how_iphone_location_works.html Tarkastettu 23.3.2015.

(Malleswari ja muut, 2009) B. L. Malleswari, I. V. MuraliKrishna, K. Lalkishore, M. Seetha, P. Hegde Nagaratna. The Role of Kalman Filter in the Modelling of GPS Errors. Journal of Theoretical & Applied Information Technology;2009, Vol. 5 Issue 1, p95.

(Mattersoft, 2015) Mattersoft. HSL Live –rajapintadokumentaatio. https://dl.dropboxusercontent.com/u/20567085/Mattersoft%20Live%21%20interface%20description%20v1_7.pdf tarkastettu 23.3.2015.

(Moves, 2014) Moves – Activity Diary for iPhone and Android. https://www.moves-app.com. Tarkastettu 11.5.2014

(Microsoft, 2013) Geocoordinate class (Windows). http://msdn.microsoft.com/en-us/library/windowsphone/develop/windows.devices.geolocation.geocoordinate.aspx Tarkastettu 2.12.2013

(Patterson ja muut, 2003) D. Patterson, L. Liao, D. Fox, ja H. Kautz, Inferring High- Level Behavior from Low-Level Sensors, ACM UBICOMP 2003.

(Ravi ja muut, 2005) Nishkam Ravi, Nikhil Dandekar, Preetham Mysore ja Michael L. Littman. Activity recognition from accelerometer data. IAAI'05 Proceedings of the 17th conference on Innovative applications of artificial intelligence - Volume 3.

(Reddy ja muut., 2010) S. Reddy, M. Mun, J. Burke, D. Estrin, M Hansen, ja M. Srivastava. Using Mobile Phones to Determine Transportation Modes. ACM Transactions on Sensor Networks, Vol. 6, No. 2, Article 13, 2010.

(SIRI, 2014) SIRI home. http://user47094.vs.easily.co.uk/siri/ Tarkastettu 21.9.2014

(Scipy, 2013) Scipy. http://www.scipy.org Tarkastettu 22.3.2015.

(Stenneth ja muut, 2011) Transportation Mode Detection using Mobile Phones and GIS Information.

(Translink, 2015) RTTI Open API – API Reference. Translink. https://developer.translink.ca/ServicesRtti/ApiReference Tarkastettu 22.2.2015.

(VDV, 2013) Verlband Deutscher Verkehrsunternehmen. http://mitglieder.vdv.de/en/wir_ueber_uns/vdv_projekte/siri.html Tarkastettu 25.11.2013

(VR, 2013) VR. Junat kartalla –rajapinta. http://www.vr.fi/fi/index/palvelut/avoin_data/Junatkartalla-rajapinta.html Tarkastettu 20.12.2012.

(Yu ja muut, 2013) Kuifei Yu, Hengshu Zhu, Huanhuan Cao, Baoxian Zhang, Enhong Chen, Jilei Tian, Jinghai Rao. Learning to Detect the Subway Station Arrival for Mobile Users. Intelligent Data Engineering and Automated Learning – IDEAL 2013 Lecture Notes in Computer Science Volume 8206.

(Zheng ja muut, 2008a) Y. Zheng, Q. Li, Y. Chen, X. Xie, ja W. Ma. Understanding mobility based on GPS data. In Ubiquitous Computing, ACM New York, 2008, pp. 312-321.

(Zheng ja muut, 2008b) Y. Zheng, L. Liu, L. Wang, X. Xie. Learning Transportation Mode from Raw GPS Data for Geographic applications on the Web. WWW 2008