



HIUKKASSUODIN- JA HIUKKASSILOITINALGORITMIT SEKÄ NIIDEN
SOVELTAMINEN AoA-MENETELMÄÄN PERUSTUVASSA
BLUETOOTH-SISÄTILAPAIKANNUKSESSA

Lasse Rintakumpu

Pro gradu -tutkielma
Lokakuu 2024

Tarkastajat:

Ohjaajan titteli (Prof./Dos./FT) ja nimi
Toisen tarkastajan titteli (Prof./Dos./FT) ja nimi

MATEMATIIKAN JA TILASTOTIETEEN LAITOS

Turun yliopiston laatujärjestelmän mukaisesti tämän julkaisun alkuperäisyys on tarkastettu Turnitin OriginalityCheck-järjestelmällä

TURUN YLIOPISTO
Matematiikan ja tilastotieteen laitos

LASSE RINTAKUMPU: Hiukkassuodin- ja hiukkassiloitinalgoritmit sekä niiden soveltaminen AoA-menetelmään perustuvassa Bluetooth-sisätilapaikannuksessa
Pro gradu -tutkielma, 81 s.
Tilastotiede
Lokakuu 2024

Tutkielmassa esitetään hiukkassuodin- ja hiukkassiloitinalgoritmien teoria Bayesilaisessa tilastotieteellisessä viitekehysessä. Lisäksi tutkielmassa käsitellään lyhyesti hiukkassuotimien varianssin estimointia.

Empiirisänä esimerkkinä tutkielmassa tarkastellaan hiukkassuodin- ja hiukkassiloitinalgoritmien käyttöä AoA-teknologiaan perustuvassa Bluetooth-sisätilapaiikkusratkaisussa.

Asiasanat: SMC-menetelmät, Monte Carlo -menetelmät, sekventiaalinen Monte Carlo, suodinongelma, hiukkassuodin, hiukkassiloitin, SIR-algoritmi, sisätilapaiikkus, BLE, AoA, triangulaatio, Bayesilainen päätely

Sisällys

1 Johdanto	3
1.1 Notaatioista	3
1.2 Suodinongelma	4
1.3 Suodin- ja siloitteluongelmien historiaa	7
1.4 Monte Carlo -menetelmistä	9
1.4.1 Monte Carlo -approksimaatio	10
1.4.2 Tärkeytysotanta	10
1.5 Bayesilainen suodin	11
1.6 Kalman-suotimen ja hiukkassuotimen yhteydestä ja eroista	12
2 Hiukkassuotimet	15
2.1 SIR-algoritmi	15
2.1.1 Suunnitteluparametrien valinta	17
2.1.2 Konvergenssituloksia	20
2.1.3 Marginaalijakauma	21
2.1.4 Aikakompleksisuus	21
2.2 Saapasremmisuodin	21
2.3 Varianssin estimoinnista	22
3 Hiukkasilottimet	27
3.1 Bayesilainen siloitin	27
3.2 Offline-algoritmit	28
3.2.1 SIR-siloitin	28
3.2.2 BS-PS-siloitin	29
3.2.3 Uudelleenpainottava hiukkassiloitin	29
3.3 Online-algoritmit	30
3.3.1 Kiinteän viipeen siloitin	30
3.3.2 Mukautuvan viipeen siloitin	31

4 Hiukkassuodin ja -siloitin sisätilapaikannuksessa	33
4.1 Sisätilapaikannuksesta	33
4.2 Teknologian kuvaus	34
4.2.1 AoA-menetelmistä	37
4.2.2 Kalibraatiosta	38
4.3 Datan kuvaus	38
4.3.1 Karttaprojektiosta	41
4.3.2 Muunnettua dataa	42
4.4 Sisätilapaikannusalgoritmi	43
4.4.1 Ongelman kuvaus	43
4.4.2 Uskottavuusmallit	45
4.4.3 Datan valinta	46
4.4.4 Dynaaminen malli	46
4.4.5 Siloittelumalli	49
4.4.6 WB-sisätilapaikannusalgoritmi	50
4.5 Empiirinen esimerkki	53
4.5.1 Koeasetelma	53
4.5.2 Parametrien valinta	57
4.5.3 Tulokset	58
5 Lopuksi	69
Liite A - Karttapolut	71
Vaihe 1	71
Vaihe 2	73
Vaihe 3	76

Luku 1

Johdanto

Hiukkassuotimet ovat joukko Monte Carlo -algoritmeja, joiden avulla voidaan ratkaista ns. suodinongelma, kun ongelma on epälineaarinen ja/tai ongelmaan liittyvä kohina ei noudata normaalijakaumaa. Hiukkassuotimille on lukuisia sovellutuksia esimerkiksi Bayesilaisessa tilastotieteessä, fysiikassa ja robotiikassa.

Tämän tutkielman tavoitteena on esittää pääpiirteittään hiukkassuotimien sekä näihin läheisesti liittyvien hiukkassiloittimien teoria. Lisäksi tutkielmana käsitellään joitakin menetelmäperheeseen kuuluvia algoritmeja ja sovelletaan näitä sisätilapai-kannuksessa.

Tutkielman ensimmäisessä luvussa kuvataan yleisellä tasolla sekä suodinongelma että sen ratkaisujen historiaa. Lisäksi esitetään joitakin Monte Carlo -menetelmiin liittyviä yleisiä tuloksia sekä Bayesilainen viitekehys suodinongelmalle. Toisessa luvussa kuvataan kaksi hiukkassuodinalgoritmia, saapasremmisuodin sekä SIR-algoritmi ja käsitellään hiukkassuotimen varianssin estimointia. Kolmannessa luvussa tarkastellaan siloitteluongelmaa ja esitetään hiukkassiloitinalgoritmeja ongelman ratkaisemiseksi. Neljäs luku keskittyy hiukkassuotimen käyttöön empiirisessä BLE/AoA-teknologiaan perustuvassa sisätilapaikannusovelluksessa. Luvussa käsitellään lisäksi sisätilapai-kannuksessa hyödynnettävää karttasovitusalgoritmia.

Hiukkassuodin- ja hiukkassiloitinalgoritmien osalta tutkielman esitykset seuraavat erityisesti Särkän kirjaa *Bayesian Filtering and Smoothing* (2013) [30], Gustafssonin artikkelia “Particle Filter Theory and Practice with Positioning Applications” (2010) [15] sekä Cappén, Godsill ja Moulines’n artikkelia “An overview of existing methods and recent advances in sequential Monte Carlo” (2007) [3]. Hiukkassuotimien varianssin estimointi seuraa Whiteleyn ja Leen artikkelia “Variance estimation in the particle filter” (2018) [20] sekä Doucet ja Olssonin artikkelia “Numerically stable online estimation of variance in particle filters” (2019) [27].

1.1 Notaatioista

Tutkielmana käytetään seuraavia yleisiä notaatioita. Hiukkassuotimien vektoreita merkitään pienellä kursivoidulla kirjaimella x , y ja w . Hiukkassuotimen hiukkas-set sisältäviä vektoreita merkitään x_k^i , missä alaindeksi viittaa aika-askeleeseen $k = \{1, \dots, T\}$ ja yläindeksi partikkeliin i , missä $i = \{1, \dots, N\}$. Vastaavasti aika-

askeleiden $k, k = \{1, \dots, T\}$ havainnot sisältäviä vektoreita merkitään $\{y_1, \dots, y_k\}$ ja painovektoreita w_k .

Pienin kursivoiduin kirjaimin viitataan myös funktioihin ja reaaliarvoisiin skaalaareihin. Pienet kirjaimet f, h, g, p ja q on varattu ensisijaisesti funktioille, p ja q erityisesti tiheysfunktioille. Skalaareihin viitataan myös isoilla kursivoiduilla kirjaimilla, esimerkiksi T ja N . Joukkoihin viitataan niin ikään isoilla kursivoiduilla kirjaimilla, esimerkiksi S . Se, käytetäänkö isoa kursivoitua kirjainta ilmaisemaan skalaaria vai joukkoa ilmenee asiayhteydestä. Matriiseja merkitään isolla lihavoidulla kirjaimella, esimerkiksi \mathbf{X} . Prosesseihin viitataan alaindeksoidulla isolla kirjaimella, esimerkiksi X_k . Taulukossa 1.1 esitetään tarkemmin tutkielman keskeisimmät merkinnät. Taulukossa 1.2 esitetään tutkielmassa käytetyt lyhenteet.

1.2 Suodinongelma

Stokastisten prosessien teoriassa suodinongelmaksi kutsutaan tilannetta, jossa halutaan muodostaa keskineliövirheen mielellä paras mahdollinen estimaatti jonkin järjestelmän tilan arvoille, kun ainoastaan osa tiloista voidaan havaita ja/tai havaintoihin liittyy kohinaa. Tavoitteena on toisin sanoen laskea jonkin prosessin posteriorijakauma kyseisten havaintojen perusteella. Ongelmaa havainnollistaa kaavio 1.1.

$$\begin{array}{ccccccc} x_1 & \longrightarrow & x_2 & \longrightarrow & x_3 & \longrightarrow & \dots & \text{piilossa olevat tilat} \\ \downarrow & & \downarrow & & \downarrow & & \\ y_1 & & y_2 & & y_3 & & \dots & \text{havainnot} \end{array} \quad (1.1)$$

Tässä tutkielmassa keskitytään erityisesti epälineaarisen, ns. Markovin piilomallin posteriorijakauman Bayesilaiseen ratkaisuun. Ongelmassa tiedetään, miten havaitut muuttujat y_k kytkeytyvät “piilossa oleviin” tilamuuttuijiin x_k sekä osataan sanoa joitain tilamuuttujien todennäköisyysrististä. Oletetaan myös, että piilossa oleville tiloille X_k pätee Markov-ominaisuus, jolloin kutakin hetkeä seuraava tila x_{k+1} riippuu menneistä tiloista $x_{1:k-1}$ ainoastaan tilan x_k välityksellä. Lisäksi havaittu tila y_k riippuu tiloista x_k ainoastaan jonkin x_k :n funktion kautta. Kun aika-avaruus on diskreetti ja aika-askeleella $k = \{1, \dots, T\}$ piilossa olevan prosessin tilaa merkitään x_k ja havaittua prosessia y_k , saadaan ongelma kuvattua malleilla

$$x_{k+1} = f(x_k, \nu_k), \quad (1.2)$$

$$y_k = h(x_k) + e_k. \quad (1.3)$$

Lisäksi tiedetään prosessin alkuhetken jakauma $x_0 \sim p_{x_0}$, tähän liittyvän kohinaprosessin jakauma $\nu_k \sim p_{\nu_k}$ sekä malliin y_k liittyvä kohina $e_k \sim p_{e_k}$. Koska hiukkassuodinalgoritmit pyrkivät ratkaisemaan juurikin epälineaarisen, ei-Gaussisen suodinongelman, voivat funktiot $f(\cdot)$ ja $h(\cdot)$ olla epälineaarisia eikä kohinan tarvitse olla normaalijakautunutta.

Mallit voidaan esittää myös yleisemmässä jakaumamuodossa

Taulukko 1.1: Symbolit ja notaatiot

Merkintä	Selitys
$1 : k$	$1, \dots, k$
k	Aika-askele, skalaari, $k = 1, \dots, T$
N	Hiukkassuotimen käyttämä partikkelen lukumäärä / otoskoko
x_k^i	Hiukkassuotimen hiukkaset sisältävä vektori, alaindeksi k määrittää aika-askeleen, yläindeksi $i = 1, \dots, N$ partikkelin
y_k	Havainnot sisältävä vektori, k määrittää aika-askeleen
w_k^i	Hiukkassuotimen painovektori, indeksit kuten hiukkasvektorissa
$p(x), q(x)$	x :n tiheysfunktioita
$\hat{p}(x), \hat{q}(x)$	x : tiheysfunktion estimaatteja
$p(x y)$	x :n tiheysfunktio ehdolla y
$x_{k k-1}$	x_k :n arvo ehdollistettuna arvoilla aika-askeleeseen $k - 1$ asti
$\mathbb{E}[x]$	x :n odotusarvo
$\mathbb{E}[x y]$	x :n ehdollinen odotusarvo ehdolla y
\mathbf{X}	Matriisi
\mathbf{X}^\top	Matriisin \mathbf{X} transpoosi
Σ	Kovarianssimatriisi
$\text{tr}(\mathbf{X})$	Matriisin \mathbf{X} jälki
$\delta(x)$	Diracin deltafunktio
$\log(x)$	Luonnollinen logaritmifunktio
$\text{Var}(x)$	Varianssifunktio
$\text{id}(x)$	Identiteettifunktio
$\text{erf}(x)$	Virhefunktio
$\mathcal{O}(\cdot)$	Algoritmin asymptoottisen suoritusajan Ordo-notaatio, suoritusajan mielessä pahin mahdollinen tapaus
$\mathcal{N}(\mu, \sigma^2)$	Normaalijakauma, keskiarvoparametri μ , varianssiparametri σ^2
$\mathcal{U}(a, b)$	Jatkuva tasajakauma, välin rajaavat parametrit a ja b
X_k	Dynaaminen prosessi
\hat{X}_k	Dynaamisen prosessin estimaatti
$\int f(x)dx$	Funktion $f(x)$ Lebesguen integraali yli \mathbb{R}^n
$\mathbb{P}(X)$	Todennäköisyysfunktio $\mathcal{A} \rightarrow \mathbb{R}$
\mathcal{A}	σ -algebra
\mathbb{R}	Reaalilukujen joukko
\mathbb{R}^n	n-ulotteinen vektoriavaruus
$\mathcal{B}(X)$	Borel-mitta, pienin X :n avoimet joukot sisältävä σ -algebra
\propto	Yhtäsuuruus normalisoivaa vakiota huomioimatta
\emptyset	Tyhjä joukko
Ω_k	Painot aika-askeleella k normalisoiva tekijä

Taulukko 1.2: Lyhenteet

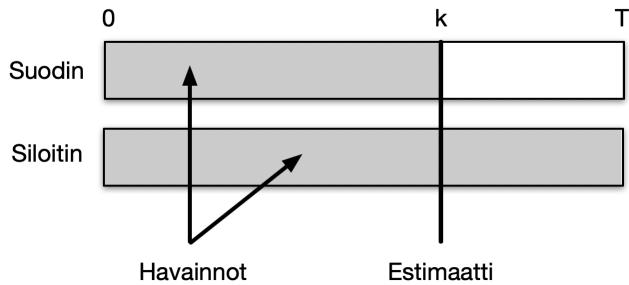
Lyhenne	Selitys
ALvar	Adaptive-Lag variance
AoA	Angle-of-Arrival
BLE	Bluetooth Low Energy
BS-PS	Backwards simulation particle smoother
CLE	Chan and Lai estimate
CR	Lithium Rechargeable
CSI	Channel State Information
EKF	Extended Kalman filter
GFSK	Gaussian frequency-shift keying
GIS	Geographic information system
GPS	Global Positioning System
IMU	Inertial measurement unit
IoT	Internet of Things
IQ, I/Q	In-phase / Quadrature
Lidar	Light detection and ranging
LPDDR	Low-Power Double Data Rate
MAC	Medium access control address
MC	Monte Carlo
MUSIC, MuSiC	MULTiple SIgnal Classification
OD	Olsson and Douc
OSCU	On-site computing unit
PCB	Printed circuit board
PKF	Position Kalman filter
RSS(I)	Received signal strength (indicator)
RTSS	Rauch-Turn-Striebel smoother
SIR	Sequential Importance Resampling
SIS	Sequential Importance Sampling
SLF	Statistically linearized Kalman filter
SMC	Sequential Monte Carlo
SNR	Signal-to-noise ratio
TDoA	Time Difference of Arrival
ToA	Time of Arrival
ToTal	Three object Triangulation algorithm
UKF	Unscented Kalman filter
WB	Walkbase
WGS	World Geodetic System

$$x_{k+1} \sim p(x_{k+1}|x_k), \quad (1.4)$$

$$y_k \sim p(y_k|x_k). \quad (1.5)$$

Tutkielman teoriaosassa käytetään ensisijaisesti yhtälöiden (1.4) ja (1.5) muotoilua. Empiirisessä osassa palataan yhtälöiden (1.2) ja (1.3) muotoiluun.

Suodinongelmaa lähellä on myös ns. siloitteluongelma (*smoothing problem*), jossa ollaan kiinnostuneita prosessin x_k posteriorijakaumasta $p(x_k|y_k)$ jokaisella aika-askeleella $\{1, \dots, k\}$, ei ainoastaan tietyllä aika-askeleella k . Hiukkassuodinalgoritmit näyttävät ratkaisevan siloitteluongelman ilmaiseksi, mutta tähän liittyy kuitenkin joidenkin mallien kohdalla mahdollista epätarkkuutta, joten tarvittaessa tasoitusongelma pitää ratkaista erikseen. Tähän ongelmaan palataan tutkielman luvussa 3. Kuva 1.1 selittää suodin- ja siloitteluongelmien eron. Kuva mukaillee Särkkää (2013) [30].



Kuva 1.1: Suodin- ja siloitteluongelma

1.3 Suodin- ja siloitteluongelmien historiaa

Tämä alaluku esittää pääpiirteitäan suodinongelmalle esitettyjen ratkaisujen historian. Lineaarisen suodinongelman osalta alaluku noudattaa Crisanin artikkelia “The stochastic filtering problem: a brief historical account” (2014) [7] sekä Grewalin ja Andrewsin artikkelia “Applications of Kalman Filtering in Aerospace 1960 to the Present” (2010) [13]. Hiukkassuotimien osalta läheinenä toimii Cappé & al (2007) [3].

Suodinongelma nousi esille insinöörityteteiden sekä sotateollisuuden käytännön ongelmista 2. maailmansodan aikana, vaikkakin suodinongelman diskreetin ajan ratkaisut juontavat jo Andrei N. Kolmogorovin 30-luvun artikkeleihin. Jatkuvan ajan tilanteessa ensimmäisen optimaalisen, kohinan sullivan suotimen esitti kybernetiikan kehittäjä, matemaatikko Norbert Wiener. Wiener-suotimena tunnettua ratkaisuaan varten Wiener muotoili seuraavat kolme ominaisuutta, jotka prosessin X_k estimaatin \hat{X}_k pitää toteuttaa.

1. *Kausaliteetti*: X_k tulee estimoida käyttäen arvoja Y_s , missä $s \leq k$.

2. *Optimaalisuus*: X_k :n estimaatin \hat{X}_k tulee minimoida keskineliövirhe $\mathbb{E}[(X - \hat{X}_k)^2]$.
3. *On-line-estimointi*: Estimaatin \hat{X}_k tulee olla saatavissa jokaisella aika-askeleella k .

Wiener sovelsi ratkaisussaan stationaaristen prosessien spektriteoriaa. Tulokset julkaistiin salaisina Yhdysvaltojen asevoimien tutkimuksesta vastanneen National Defense Research Committeeen (NDRC) raportissa vuonna 1942. Tutkimus tunnettiin sodan aikana lempinimellä ”Keltainen vaara”, sekä painopaperinsa värin etä vaikeaselkoisuutensa vuoksi. Myöhemmin Wiener esitti tuloksensa julkisesti kirjassaan *Extrapolation, Interpolation and Smoothing of Stationary Time Series* (1949). Wienerin alkuperäiset kolme perusperiaatetta päteväät edelleen kaikille suodinongelman ratkaisuille, myös hiukkassuotimille.

Kenties tärkein ja varmasti tunnetuin lineaariseen suodinongelman ratkaisu on Kalman-suodin. Suotimen kehittivät R.E. Kalman ja R.S. Bucy 1950- ja 60-lukujen taitteessa Yhdysvaltain kylmän sodan kilpavarustelutarpeisiin perustetussa Research Institute for Advanced Studies -tutkimuslaitoksessa (RIAS). Kalman-suodin on suodinongelman diskreetin ajan ratkaisu, kun taas Kalman-Bucy-suodin on jatkuvan ajan ratkaisu. Kohinan ollessa normaalijakautunutta on Kalman-suodin Wiener-suotimen tavoin lineaarisen suodinongelman optimaalinen ratkaisu. Wiener-suotimella ja Kalman-suotimella on kuitenkin erilaiset oletukset, minkä vuoksi erityisesti säättö- ja paikannusovelluksissa Kalman-suotimen käyttö on luontevampaa. Suotimien oletuksia ja oletusten välisiä eroja ei käsitellä tässä tutkielmassa, mutta alaluvussa [1.6](#) käsitellään Kalman-suotimen formaalia yhteyttä hiukkassuotimiin.

Kalman-suodinta voidaan soveltaa myös epälineaarisessa tapauksessa, kunhan suodinongelman funktiot $f(\cdot)$ ja $h(\cdot)$ ovat derivoituvia ja niihin liittyvä kohina oletetaan normaalijakautuneeksi. Tätä ratkaisua kutsutaan laajennetuksi Kalman-suotimeksi (*extended Kalman filter*, EKF). Suodin kehitettiin 60-luvulla NASA:n Apollo-ohjelman tarpeisiin, vaikkakin itse avaruusalusten laitteistot hyödynsivät lento-ratojen laskennassa Kalman-suotimen perusversiota. Laajennettu Kalman-suotimen toimintaperiaate perustuu epälineaaristen funktioiden linearisointiin Taylorin kehitelmän avulla kulloisenkin estimaatin ympärillä. Laajennettu Kalman-suodin on erityisesti paikannusovellusten *de facto*-suodinstandardi, mutta suodin ei kuitenkaan ole epälineaarisen ongelman optimaalinen estimaattori.

Kalman-suotimesta on lisäksi olemassa lukuisia muita epälineaarisiin ongelmiin soveltuivia laajennuksia, muun muassa paikkaratkaisun Kalman-suodin (*position Kalman filter*, PKF), hajustamaton Kalman-suodin (*unscented Kalman filter*, UKF) sekä tilastollisesti linearisoitu Kalman-suodin (*statistically linearized Kalman filter*, SLF). Kuitenkin jos prosessin X_t mallia ei tunneta tarkasti tai kohinaa ei voida olettaa normaalijakautuneeksi, ovat hiukkassuotimet eli sekventiaaliset Monte Carlo -menetelmät Kalman-suotimen johdannaisia parempia ratkaisuja. Vaikka tila-avaruuden dimensioiden kasvaessa kasvaa myös hiukkassuotimien vaatima laskentateho, ovat hiukkassuotimet aina sitä parempia mitä epälineaarisempia mallit ovat ja mitä kauempana normaalijakaumasta kohina on. Viimeisten vuosikymmenten aikana laskennan teho on kasvanut merkittävästi samalla kun laskennan hinta on

vastaavasti romahtanut, mikä puolaa Monte Carlo -menetelmien käyttöä entistä useammissa ongelmissa.

Jotakin suodinongelman rekursiivisia Monte Carlo -ratkaisuja löytyy jo 1950–70-luvuilta, erityisesti säätöteoriaan piiristä. Olennainen nykyalgoritmeihin periytyneyt oivallus varhaisissa suodinalgoritmmeissa oli tärkeytysotannan käyttö halutun jakau-maestimaatin laskennassa. Tärkeytysotanta-algoritmiin voidaan turvautua, kun emme pysty suoraan tekemään havaintoja jostakin jakaumasta p ja teemme sen sijaan ha-vaintoja jakaumasta q , jota painotamme niin, että tuloksena saadaan jakauman p harhaton estimaatti. Algoritmi on kuvattu tarkemmin tutkielman luvussa 2.

Tärkeytysotantaa käyttävä suodinongelman ratkaiseva SIS-algoritmi (*sequential importance sampling*) ei kuitenkaan vielä 70-luvulla löytänyt suurta käytännön suosiota. Osin tämä johtui puutteellisesta laskentatehosta, mutta algoritmi kärsi myös otosten ehtymisenä (*sample impoverishment*) tunnetusta ongelmasta. Monissa ongelmissa SIS-algoritmia käytettäessä suuri osa painoista päättyy vain tietyille partikkeleille, jolloin vastaavasti suuri osa partikkeleista ei enää estimoi haluttua jakaumaa. Tähän ongelmaan palataan myöhemmin tutkielmassa.

Merkittävän ratkaisun ehtymisongelmaan esittivät Gordon, Salmond ja Smith artikkelissaan “Novel approach to nonlinear/non-Gaussian Bayesian state estimation” (1993) [12]. Artikkelin ratkaisu kulki nimellä *bootstrap filter*, saapasremmisuodin. Saapasremmisuodin vältti otosten ehtymisen uudellenotannalla, jossa matalapainoiset partikkelit korvattiin otoksilla korkeapainoisemmista partikkeleista. Ratkaisussa painot eivät myöskään riippuneet partikkelien aiemmista poluista vaan ainoastaan havaintojen uskottavuusfunktiosta. Vastaavaa ratkaisua käytetään tämän tutkielman uudemmassa SIR-algoritmissa (*sampling importance resampling*), jossa myös uudelleenotantaan sovelletaan tärkeytysotantaa.

Sekventiaalisissa Monte Carlo -menetelmissä stokastisen prosessin posteriorijakauman esittämiseen käytettyjä otoksia kutsutaan partikkeleiksi tai hiukkasiksi ja menetelmiä siten hiukkassuotimiksi. Erityisesti myöhemmin esitettävää SIR-algoritmia kutsutaan usein hiukkassuotimeksi. Termiä hiukkassuodin käytti ensimmäisen kerran Del Moral artikkelissa “Nonlinear Filtering: Interacting Particle Resolution” (1996) [23], SMC-menetelmät termiä Liu ja Chen artikkelissa “Sequential Monte Carlo Methods for Dynamic Systems” (1998) [21]. Tässä tutkielmassa käytetään yleisemmin käytettyä termiä hiukkassuotimet.

1.4 Monte Carlo -menetelmistä

Tässä alaluvussa kuvataan lyhyesti hiukkassuotimissa käytettävien Monte Carlo -menetelmien perusperiaate todennäköisyysjakauman estimoinnissa. Lisäksi esitetään tärkeytysotanta-algoritmi (*importance sampling*), jonka tarkoituksesta on estimoida harhattomasti jakaumaa $p(x|y_{1:k})$, josta emme voi suoraan tehdä otoksia, mutta jota voimme approksimoida toisella jakaumalla q . Esitykset noudattavat Särkkää (2013) [30].

1.4.1 Monte Carlo -approksimaatio

Bayesilaisessa päätelyssä ollaan yleisesti kiinnostuttu laskemaan johonkin posterioriheysjakaumaan p liittyvää odotusarvoa

$$\mathbb{E}[g(x)|y_{1:k}] = \int g(x)p(x|y_{1:k})dx, \quad (1.6)$$

missä g on tila-avaruuden mielivaltainen funktio ja $p(x|y_{1:k})$ on havaintoihin $\{y_1, \dots, y_k\}$ liittyvä x :n posterioriheysjakauma. Odotusarvo on laskettavissa suljetussa muodossa vain harvoissa tapauksissa, suodinongelman kohdalla silloin, kun kyseessä on lineaarinen ja Gaussinen malli. Odotusarvoa voidaan kuitenkin approksimoida niin sanoituilla Monte Carlo -menetelmillä. Menetelmien perusperiaatteet ovat tehdä riippumattomia otoksia estimoitavasta jakaumasta ja laskea haluttu odotusarvo otosten avulla. Jos tehdään N otosta jakaumasta $x^i \sim p(x|y_{1:k})$, missä $i = \{1, \dots, N\}$ saadaan näiden otosten avulla laskettua odotusarvon estimaatti

$$\mathbb{E}[g(x)|y_{1:k}] \simeq \frac{1}{N} \sum_{i=1}^N g(x^i). \quad (1.7)$$

Kun otokset ovat riippumattomia ja samoin jakautuneita, konvergoi Monte Carlo -estimaatti keskeisen raja-arvolauseen nojalla ja sen estimointivirheen voidaan osoittaa olevan luokkaa $\mathcal{O}(\frac{1}{\sqrt{N}})$ riippumatta tilamuuttujan x dimensiosta. Hiukassuotimet hyödyntävät Monte Carlo -estimointia sekventiaalisesti, jolloin estimaatti lasketaan rekursiivisesti kullekin aika-askeleelle $k = \{1, \dots, T\}$. Tähän palataan luvuissa 2 ja 4.

1.4.2 Tärkeytysotanta

Tilanteessa, jossa Monte Carlo -otoksia ei voida tehdä suoraan jakaumasta p , voidaan hyödyntää jakaumaa p approksimoivaa tärkeytys- tai ehdotusjakaumaa $q(x|y_{1:k})$ sekä ns. tärkeytysontantaa. Oletetaan, että tunnetaan priorijakauma $p(x)$ ja on olemassa havaintomalli $p(y_{1:k}|x)$ sekä valittu ehdotusjakauma $q(x|y_{1:k})$, josta voidaan tehdä otoksia. Ehdotusjakaumalta edellytetään lisäksi, että sen kantaja on suurempi tai yhtä suuri kuin jakauman $p(x|y_{1:k})$ ja että se saa nollasta poikkeavia arvoja kaikkialla missä $p(x|y_{1:k})$ saa nollasta poikkeavia arvoja. Ehdotusjakauman on myös hyvä olla mahdollisimman lähellä posteriorijakaumaa p . Kirjoitetaan halutun posteriorijakauman odotusarvo integraalina

$$\int g(x)p(x|y_{1:k})dx = \int g(x) \frac{p(x|y_{1:k})}{q(x|y_{1:k})} q(x|y_{1:k})dx, \quad (1.8)$$

jolle voidaan muodostaa Monte Carlo -approksimaatio tekemällä N otosta jakaumasta $x^i \sim q(x|y_{1:k})$.

Muodostetaan näin odotusarvo

$$\mathbb{E}[g(x)|y_{1:k}] \simeq \frac{1}{N} \sum_{i=1}^N \frac{p(x^i|y_{1:k})}{q(x^i|y_{1:k})} g(x^i) = \sum_{i=1}^N w^i g(x^i), \quad (1.9)$$

missä $g(x)$ on jokin estimoinnissa hyödyllinen, mielivaltainen funktio. Tutkielman massaa käytetty notaatio x_k^i viittaa aika-askeleen k partikkeliin i , missä $i = \{1, \dots, N\}$. Tärkeytysotantaa kuvaavat nyt algoritmi 1. Kun posteriorijakauman estimaatti muodostetaan kyseisellä algoritmillä voidaan tulos kirjoittaa

$$\hat{p}(x|y_{1:k}) = \sum_{i=1}^N w^i \delta(x - x^i), \quad (1.10)$$

missä $\delta(x)$ on Diracin deltafunktio. Diracin deltafunktio kuvailee sitä, että otokset ovat diskreettejä ja kullakin otoksella on oma painonsa (eli otos muodostaa “piikin” posteriorijakaumaan).

Algoritmi 1: Tärkeytysotanta

```

begin
  for i = 1, 2, ..., N do
    begin
      | Otetaan N otosta ehdotusjakaumasta  $x^i \sim q(x|y_{1:k})$ .
      begin
        | Lasketaan normalisoimattomat painot  $w_*^i = p(y_{1:k}|x^i)p(x^i)/q(x^i|y_{1:k})$ .
        | ja normalisoidut painot  $w^i = w_*^i / \sum_{j=1}^N w_*^j$ .
        begin
          | Estimoidaan p laskemalla tiheydelle approksimaatio
          |  $\mathbb{E}[g(x)|y_{1:k}] \simeq \sum_{i=1}^N w^i g(x^i)$ .

```

1.5 Bayesilainen suodin

Suodinongelmassa ollaan kiinnostuttu tilavektorin posteriorijakauman $p(x_k|y_{1:k})$ estimoinnista. Tässä alaluvussa käydään läpi yleinen rekursiivinen, Bayesilainen posteriorijakauman laskenta. Tällaista suodinongelman ratkaisua kutsutaan myös Bayesilaiseksi suotimeksi. Koska epälineaarisessa, ei-normaalijakautuneessa tilanteessa rekursiota ei voida laskea analyttisesti, pitää estimoinnissa käyttää numeerisia menetelmiä. Hiukkassuotimissa tämä tarkoittaa jakauman sekventiaalista Monte Carlo -approksimointia ja sen käytännön toteutus esitetään luvun 4 algoritmissa. Molemmat esitykset noudattavat Gustafssonia (2010) [15].

Bayesilainen ratkaisu tilavektorin posteriorijakauman estimaatiille $\hat{p}(x_k|y_{1:k})$ saadaan seuraavalla rekursiolla (käydään läpi jokaisella aika-askeleella $k = \{1, \dots, T\}$). Lasketaan ensin

$$p(x_k|y_{1:k}) = \frac{p(y_k|x_k)p(x_k|y_{1:k-1})}{p(y_k|y_{1:k-1})}, \quad (1.11)$$

joka saadaan suoraan Bayesin kaavasta $\mathbb{P}(A|B) = \mathbb{P}(B|A)\mathbb{P}(A)/\mathbb{P}(B)$. Normalisointivakio lasketaan integraalina

$$p(y_k|y_{1:k-1}) = \int_{\mathbb{R}^{n_x}} p(y_k|x_k)p(x_k|y_{1:k-1}) dx_k, \quad (1.12)$$

joka saadaan kokonaistodennäköisyyskaavasta $\mathbb{P}(A) = \mathbb{E}[\mathbb{P}(A|X)] = \int_{-\infty}^{\infty} \mathbb{P}(A|X=x) f_X(x) dx$. Merkintä \mathbb{R}^{n_x} vastaa tässä piilossa olevan tilavektorin dimensiota n .

Lopuksi lasketaan päivitysaskel ajalle, joka saadaan edelleen kokonaistodennäköisyydellä

$$p(x_{k+1}|y_{1:k}) = \int_{\mathbb{R}^{n_x}} p(x_{k+1}|x_k)p(x_k|y_{1:k}) dx_k. \quad (1.13)$$

Rekursion avulla voimme laskea jakauman $p(x_k|y_{1:k})$ estimaatin käymällä rekursion läpi k kertaa.

1.6 Kalman-suotimen ja hiukkassuotimen yhteydestä ja eroista

Tässä alaluvussa käsitellään lyhyesti Kalman-suotimen yhteyttä hiukkassuotimeen edellä esitetyn teorian valossa. Esitys noudattaa Särkkää (2013) [30]. Merkitään kuten edellä dynaamista mallia x_k ja havaintomallia y_k ja oletataan toisin kuin edellä, että nämä ovat lineaarisia ja noudattavat normaalijakaumaa. Koska mallit ovat lineaarisia, voidaan ne nyt kirjoittaa muotoon

$$x_k = \mathbf{A}_{k-1}x_{k-1} + q_{k-1}, \quad (1.14)$$

$$y_k = \mathbf{H}_k x_k + r_k, \quad (1.15)$$

missä \mathbf{A}_{k-1} on dynaamisen mallin tilasiirtymään kuvaava matriisi ja \mathbf{H}_k on havaintojen mallimatriisi. Normaalisuusoletuksesta seuraa, että sekä mallin että prosessin kohinavektorit noudattavat normaalijakaumia $q_{k-1} \sim \mathcal{N}(0, \mathbf{Q}_{k-1})$ ja $r_k \sim \mathcal{N}(0, \mathbf{R}_k)$, missä \mathbf{Q}_{k-1} ja \mathbf{R}_k ovat kovarianssimatrssiiseja. Lisäksi oletetaan, että prosessin priorijakauma on normaali eli $x_0 \sim \mathcal{N}(m_0, \mathbf{P}_0)$. Mallit voidaan nyt kirjoittaa tiheysfunktiomuodossa

$$p(x_k|x_{k-1}) = \mathcal{N}(x_k|\mathbf{A}_{k-1}x_{k-1}, \mathbf{Q}_{k-1}), \quad (1.16)$$

$$p(y_k|x_k) = \mathcal{N}(y_k|\mathbf{H}_k x_k, \mathbf{R}_k), \quad (1.17)$$

joista voidaan edelleen johtaa suodinongelman mallit

$$p(x_k|y_{1:k-1}) = \mathcal{N}(x_k|m_k^*, \mathbf{P}_k^*), \quad (1.18)$$

$$p(x_k|y_{1:k}) = \mathcal{N}(x_k|m_k, \mathbf{P}_k), \quad (1.19)$$

$$p(y_k|y_{1:k-1}) = \mathcal{N}(y_k|\mathbf{H}_k m_k^*, \mathbf{S}_k) \quad (1.20)$$

ja ongelma ratkaista näin algoritmilla 2.

Algoritmi 2: Kalman-suodin

Result: Posteriorijakauman $p(x_{1:k}|y_{1:k})$ estimaatti.

Data: Havainnot y_k . Priorijakauman x_0 keskiarvovektori m_0 ja kovarianssimatriisi \mathbf{P}_0 .

```

begin
  for  $k = \{1, 2, \dots, T\}$  do
    begin
      Ennusteaskel.
       $m_k^* = \mathbf{A}_{k-1}m_{k-1}$ 
      if  $k < t$  then
        begin
          Päivitysaskel.
           $v_k = y_k - \mathbf{H}_k m_k^*$ 
           $\mathbf{S}_k = \mathbf{H}_k \mathbf{P}_k^* \mathbf{H}_k^\top + \mathbf{R}_k$ 
           $\mathbf{K}_k = \mathbf{P}_k^* \mathbf{H}_k^\top \mathbf{S}_k^{-1}$ 
           $m_k = m_k^* \mathbf{K}_k v_k$ 
           $\mathbf{P}_k = \mathbf{P}_k^* - \mathbf{K}_k \mathbf{S}_k \mathbf{K}_k^\top;$ 
    
```

Esitetty algoritmi on ns. Kalman-suodin, joka selkeästi toimii suodinongelman ratkaisuna, kun mallit ovat haluttua lineaarista normaalimuotoa. Jos tämä oletus ei täyty, on Kalman-suotimesta kehitetty useita versioita, joissa ei-lineaarinen malli voidaan linearisoida tietyjen ehtojen vallitessa.

Tämän alaluvun tarkoituksena oli esittää, että Kalman-suotimessa ongelma on samaa muotoa kuin hiukkassuotimessa, joten linearisoituja Kalman-suotimia ei tässä käsitellä. Hiukkassuodin myös ratkaisee suodinongelman mille hyvänsä epälineaariselle mallille.

Luku 2

Hiukkassuotimet

2.1 SIR-algoritmi

Tässä alaluvussa esitetään epälineaarisen suodinongelman ratkaisemiseksi SIR-hiukkassuodinalgoritmi. Algoritmi on numeerinen toteutus luvussa 1.5 kuvatusta Bayesilaisesta suotimesta. Esitetty algoritmi perustuu Gustafssonin (2010) [15]. Ilman uudelleenotantavaihetta kyseessä olisi SIS-algoritmi.

Algoritmi alustetaan jakaumasta $x_1^i \sim p_{x_0}$ generoiduilla N kappaleella partikkeleita. Jokaiselle partikkelille annetaan alustuksessa sama paino $w_{1|0}^i = 1/N$. Algoritmi suoritetaan jokaiselle partikkelille $i = \{1, 2, \dots, N\}$ jokaisella aika-askeleella $k = \{1, 2, \dots, T\}$.

Algoritmin ensimmäisessä vaiheessa päivitetään painot yhtälön (2.1) mukaan.

$$w_{k|k}^i = \frac{1}{c_k} w_{k|k-1}^i p(y_k | x_k^i). \quad (2.1)$$

Tämä vastaa edellä esitetyn Bayes-suotimen päivitysvaihetta (1.11). Normalisointipaino c_k lasketaan puolestaan yhtälöstä (2.2), mikä vastaa Bayes-suotimen normalisointivakion laskemista (1.12) ja asettaa painojen summaksi $\sum_{i=1}^N w_{k|k}^i = 1$.

$$c_k = \sum_{i=1}^N w_{k|k-1}^i p(y_k | x_k^i). \quad (2.2)$$

Seuraavassa vaiheessa estimoidaan p laskemalla tiheyden $p(x_{1:k}|y_{1:k})$ Monte Carlo -estimaatti yhtälön (2.3) perusteella

$$\hat{p}(x_{1:k}|y_{1:k}) = \sum_{i=1}^N w_{k|k}^i \delta(x_{1:k} - x_{1:k}^i). \quad (2.3)$$

Tämän jälkeen suoritetaan valinnainen uudelleenotanta. Uudelleenotanta voidaan tehdä jokaisella askeleella tai efektiivisen otoskoon perusteella alla kuvatun

kynnysarvoehdon $\hat{N}_{eff} < N_{th}$ täytessä, jolloin uudelleenotantaa kutsutaan adaptiiviseksi uudelleenotannaksi. Tällaista uudelleenotantaa hyödynnetään esitetyssä algoritmissa (3). Lopuksi päivitetään aika-askeleita (jos $k < T$), luodaan uudet ennusteet partikkeille ehdotusjakaumasta (2.4)

$$x_{k+1}^i \sim q(x_{k+1}|x_k^i, y_{k+1}) \quad (2.4)$$

ja päivitetään partikkelienvaihtoaineksen (2.5) sen mukaan kuinka todennäköisiä partikkelienvaihtoaineksen ovat

$$w_{k+1|k}^i = w_{k|k}^i \frac{p(x_{k+1}^i|x_k^i)}{q(x_{k+1}^i|x_k^i, y_{k+1})}. \quad (2.5)$$

Vaiheet (2.4) ja (2.5) vastaavat Bayes-suotimen aikapäivitystä (1.13).

Alla käsitellään algoritmiin liittyvän uudelleenotantamenetelmän, partikkelienvaihtoaineksen ja ehdotusjakauman valinta. Lopuksi esitetään algoritmin konvergenssia, marginaalijakaumaa sekä aikakompleksisuutta koskevia tuloksia ja käsitellään algoritmin tuottaman jakaumaestimaatin varianssin estimointia.

Algoritmi 3: SIR

Result: Posteriorijakauman $p(x_{1:k}|y_{1:k})$ estimaatti.

Data: Havainnot y_k . Generoitut $x_1^i \sim p_{x_0}$ missä $i = \{1, \dots, N\}$ ja jokainen partikkeli saa saman painon $w_{1|0}^i = 1/N$.

```
begin
    for  $k = \{1, 2, \dots, T\}$  do
        for  $i = \{1, 2, \dots, N\}$  do
            begin
                | Päivitetään painot  $w_{k|k}$ .
            begin
                | Estimoidaan  $p$  laskemalla tiheydelle approksimaatio
                |  $\hat{p}(x_{1:k}|y_{1:k}) = \sum_{i=1}^N w_{k|k}^i \delta(x_{1:k} - x_{1:k}^i)$ .
            begin
                | Lasketaan efektiivinen otoskoko  $\hat{N}_{eff}$ .
            if  $\hat{N}_{eff} < N_{th}$  then
                begin
                    | Otetaan uudet  $N$  otosta palauttaen joukosta  $\{x_{1:k}^i\}_{i=1}^N$ , missä
                    | otoksen  $i$  todennäköisyys on  $w_{k|k}^i$ .
                begin
                    | Asetetaan painot  $w_{k|k}^i = 1/N$ .
                if  $k < T$  then
                    begin
                        | Aikapäivitys.
                        | Luodaan ennusteet partikkeilleille ehdotusjakaumasta
                        |  $x_{k+1}^i \sim q(x_{k+1}|x_k^i, y_{k+1})$ ,
                        | päivitetään partikkeliin painot tärkeytysotannalla.
```

2.1.1 Suunnitteluparametrien valinta

Ennen algoritmin suorittamista valitaan ehdotusjakauma $q(x_{k+1}|x_{1:k}, y_{k+1})$, uudelleenotantamenetelmä sekä partikkeliin määärä N . Ehdotusjakauman ja uudelleenotantamenetelmän valinnassa tärkeimpänä päämäääränä on välttää otosten ehtymistä, kun taas partikkeliin määärä säätelee kompromissia algoritmin suorituskyvyn ja sen tarkkuuden sekä varianssin välillä.

2.1.1.1 Otoskoon N valinta

Yleispätevästi sääntöä otoskoon/partikkeliin lukumäärästä N valinnalle on vaikeaa antaa, sillä vaadittava estimointitarkkuus riippuu usein kässillä olevasta ongelmasta. Gordon &al. (1993) [12] esittävät kuitenkin kolme tekijää, jotka vaikuttavat partikkeliin lukumäärästä valintaan

- a. tila-avaruuden ulottuvuuksien lukumäärä n_x ,

- b. tyyppillinen pääallekäisyys priorin ja uskottavuuden välillä
- c. sekä tarvittava aika-askelten lukumäärä.

Ensimmäisen tekijän vaikutus on selvä. Mitä useammassa ulottuvuudessa otantaa tarvitsee tehdä, sen korkeammaksi on N asetettava, jotta jokainen ulottuvuuus pystytään kattamaan tehokkaasti. Tekijät (b) ja (c) puolestaan seuraavat uudelleenotannasta. Jos se osa tila-avaruutta, jossa uskottavuus $p(y_k|x_k)$ saa merkittäviä arvoja on pieni verrattuna siihen osaan, jossa priorijakauma $p(x_k|y_{1:k-1})$ saa merkittäviä arvoja, suuri osa partikkeleista saa pieniä painoja eikä näin valikoidu uudelleenotantaan.

Yleisesti ottaen N kannattaa asettaa sellaiseksi, että se paitsi tuottaa riittävän tarkan estimaatin, on se myös käytettävissä olevan laskentatehon sekä vaadittavan laskentanopeuden kannalta järkevä. Tähän palataan tutkielman empiirisessä paikannusesimerkissä.

2.1.1.2 Uudelleenotantamenetelmän valinta

Ilman uudelleenotantaa on mahdollista, että algoritmi alkaa kärsiä SIS-algoritmille tyypillisestä otosten ehtymisestä. Toisin sanoen kaikki painot alkavat keskittyä vain muutamalle partikkelille eikä algoritmi enää approksimoi tehokkaasti haluttua jakaumaa. Uudelleenotanta tarjoaa osittaisen ratkaisun tähän ongelmaan, mutta hävittää samalla informaatiota ja lisää siten satunnaisotantaan liittyvää epävarmuutta. Yleisesti ottaen uudelleenotanta kannattaa aloittaa vasta siinä vaiheessa algoritmin suorittamista, kun siitä on otosten ehtymisen kannalta hyötyä, esimerkiksi efektiivisen otoskoon pudottua jonkin kynnysarvon alapuolelle (adaptiivinen uudelleenotanta). Efektiivinen otoskoko saadaan laskettua variaatiokertoimesta c_ν kaavalla

$$N_{eff} = \frac{N}{1 + c_\nu^2(w_{k|k}^i)} = \frac{N}{1 + \frac{\text{Var}(w_{k|k}^i)}{(\mathbb{E}[w_{k|k}^i])^2}} = \frac{N}{1 + N^2 \text{Var}(w_{k|k}^i)}. \quad (2.6)$$

Näin laskettu efektiivinen otoskoko kuvaaa sitä, kuinka tasaisesti painot jakautuvat partikkeleille. Efektiivinen otoskoko maksimoituu ($N_{eff} = N$), kun kaikille painoille pätee $w_{k|k}^i = 1/N$ ja minimoituu ($N_{eff} = 1$), kun $w_{k|k}^i = 1$ todennäköisyydellä $1/N$ ja $w_{k|k}^i = 0$ todennäköisyydellä $(N-1)/N$. Normalisoitujen painojen avulla saadaan efektiiviselle otoskoolle aika-askeleella k laskennallinen approksimaatio

$$\hat{N}_{eff} = \frac{1}{\sum_{i=1}^N (w_{k|k}^i)^2}. \quad (2.7)$$

Määritelmille (2.6) ja (2.7) pätee $1 \leq \hat{N}_{eff} \leq N$. Yläraja saavutetaan, kun jokaisen partikkelin paino on sama. Alarajalle päädytään, kun kaikki paino keskittyy yksittäiselle partikkelille. Tästä saadaan määriteltyä algoritmille SIR-uudelleenotantaehto $\hat{N}_{eff} < N_{th}$. Gustafsson (2010) [15] esittää uudelleenotannan kynnysarvoksi esimerkiksi $\hat{N}_{th} = 2N/3$.

Uudelleenotanta ei muuta approksimoitavan jakauman p odotusarvoa, mutta se lisää jakauman Monte Carlo -varianssia. On kuitenkin olemassa esimerkiksi osittamiseen perustuvia uudelleenotentamenetelmiä, jotka pyrkivät minimoimaan varianssin lisäyksen. Varianssin pienennysmenetelmät jätetään tämän tutkielman ulkopuolelle.

2.1.1.3 Ehdotusjakauman valinta

Yksinkertaisin muoto ehdotusjakaumalle on $q(x_{1:k}|y_{1:k})$ eli jokaisella algoritmin suorituskerralla käydään läpi koko aikapolku $1:k$. Tämä ei kuitenkaan ole tarkoitukseenmukaista, erityisesti jos kyseessä on reaalialkainen sovellus. Kirjoitetaan sen sijaan ehdotusjakauma muodossa

$$q(x_{1:k}|y_{1:k}) = q(x_k|x_{1:k-1}, y_{1:k})q(x_{1:k-1}|y_{1:k}). \quad (2.8)$$

Jos yhtälöstä (2.8) poimitaan ehdotusjakaumaksi ainoastaan termi $q(x_k|x_{1:k-1}, y_{1:k})$, voidaan tämä kirjoittaa edelleen Markov-ominaisuuden nojalla muotoon $q(x_k|x_{k-1}, y_k)$. Tämä on suodinongelman kannalta riittävää, koska olemme kiinnostuneita posteriorijakaumasta ja arvosta x ainoastaan aika-askeleella k (siloitinongelmassa tarvitsisimme koko polun $x_{1:k}$). Alla tarkastellaan edelleen Gustafssonia (2010) [15] seuraten kahta ehdotusjakauman valintatapaa, prioriotantaa (*prior sampling*) sekä uskottavuusotantaa (*likelihood sampling*).

Ennen ehdotusjakauman tarkastelua määritellään mallille signaali-kohinasuhde uskottavuuden maksimin ja priorin maksimin välisenä suhteena

$$\text{SNR} \propto \frac{\max_{x_k} p(y_k|x_k)}{\max_{x_k} p(x_k|x_{k-1})}. \quad (2.9)$$

Yhdistetään lisäksi ehdotusjakaumia varten yhtälöt (2.1) ja (2.2), jolloin saadaan painojen päivitys muotoon

$$w_{k|k}^i \propto w_{k-1|k-1}^i \frac{p(y_k|x_k^i)p(x_k|x^{k-1})}{q(x_k|x_{k-1}^i, y_k)}. \quad (2.10)$$

Kun suhde (2.9) on matala, on prioriotanta luonnollinen valinta. Tässä käytetään ehdotusjakaumana tilavektorin ehdollista prioria eli

$$q(x_k|x_{1:k-1}, y_k) = p(x_k|x_{k-1}^i). \quad (2.11)$$

Yhtälön (2.11) perusteella saadaan edelleen prioriotannan painoiksi

$$w_{k|k}^i = w_{k-1|k-1}^i p(y_k|x_k^i) = w_{k-1|k-1}^i p(y_k|x_k^i). \quad (2.12)$$

Kun signaali-kohinasuhde on kohtalainen tai korkea, on parempi käyttää ehdotusjakaumana skaalattua uskottavuusfunktiota (2.14). Tarkastellaan ensin tekijöihin jakaoa

$$p(x_k|x_{k-1}^i, y_k) = p(y_k|x_k) \frac{p(x_k|x_{k-1}^i)}{p(y_k|x_{k-1}^i)}. \quad (2.13)$$

Kun SNR on korkea ja uskottavuusfunktio on integroituva pätee $p(x_k|x_{k-1}^i, y_k) \propto p(y_k|x_k)$, jolloin voidaan asettaa

$$q(x_k|x_{k-1}^i, y_k) \propto p(y_k|x_k). \quad (2.14)$$

Yhtälön (2.14) perusteella saadaan edelleen uskottavuusotannan painoiksi

$$w_{k|k}^i = w_{k-1|k-1}^i p(x_k^i|x_{k-1}^i). \quad (2.15)$$

2.1.2 Konvergenssituloksia

Alla esitetään kolme SIR-algoritmiin liittyvää konvergenssitulosta. Se, kuinka hyvin esitetyllä algoritmilla arvioitu posterioriteleys $\hat{p}(x_{1:k}|y_{1:k})$ approksimoi todellista tiheysfunktiota $p(x_{1:k}|y_{1:k})$, se mikä on approksimaation keskineliövirhe sekä keskeinen raja-arvolause. Tulokset 1–2 noudattavat Crisanin ja Doucet’n artikkeleita “Convergence of Sequential Monte Carlo Methods” (2000) [8] ja “A Survey of Convergence Results on Particle Filtering Methods for Practitioners” (2002) [9], tulos 3 Chopinin artikkelia “Central limit theorem for sequential Monte Carlo methods and its application to Bayesian inference” (2004) [6].

Konvergenssitulo 1: Kun $N \rightarrow \infty$ algoritmilelle pätee kaikille aika-askeleille k tulos (2.16).

$$\hat{p}(x_{1:k}|y_{1:k}) \xrightarrow{a.s.} p(x_{1:k}|y_{1:k}). \quad (2.16)$$

Konvergenssitulo 2: Keskineliövirheelle pätee asymptoottinen konvergenssi (2.17).

$$\mathbb{E} [\hat{g}(x_k) - \mathbb{E} [g(x_k)]]^2 \leq \frac{p_k \|g(x_k)\|}{N}, \quad (2.17)$$

missä g on mikä hyvänsä piilossa olevan tila-avaruuden rajoitettu Borel-mitallinen funktio ($g \in \mathcal{B}(\mathbb{R}^{n_x})$), $\|g(\cdot)\|$ kyseisen funktion supremum-normi ja p_k jokin äärellinen vakio, jolle pätee ajanhetkestä k riippumatta $p_k = p < \infty$.

Konvergenssitulo 3: Keskeinen raja-arvolause (2.18).

$$\text{Kun } N \rightarrow \infty : \sqrt{N} \left\{ \frac{1}{N} \sum_{i=1}^N \hat{g}(x_k^i) - \mathbb{E} [g(x_k^i)] \right\} \xrightarrow{D} \mathcal{N}(0, \sigma^2 < \infty), \quad (2.18)$$

missä g on jälleen mikä hyvänsä piilossa olevan tila-avaruuden rajoitettu Borel-mitallinen funktio ($g \in \mathcal{B}(\mathbb{R}^{n_x})$). Konvergenssituloksia ei tämän tutkielman puitteissa todisteta.

2.1.3 Marginaalijakauma

Edellä kuvattu algoritmi 3 tuottaa approksimaation koko prosessin posteriorijakaumalle $p(x_{1:k}|y_{1:k})$. Jos halutaan tietää ainoastaan posteriorijakauman $p(x_k|y_{1:k})$ estimaatti, voidaan käyttää yksinkertaisesti viimeisestä tilasta x_k laskettua estimaatia

$$\hat{p}(x_k|y_{1:k}) = \sum_{i=1}^N w_{k|k}^i \delta(x_k - x_k^i). \quad (2.19)$$

Toinen, tarkempi vaihtoehto on käyttää laskennassa tärkeytyspainoa

$$w_{k+1|k}^i = \frac{\sum_{j=1}^N w_{k|k}^j p(x_{k+1}^i|x_k^j)}{q(x_{k+1}^i|x_k^i, y_{k+1})} \quad (2.20)$$

painon (2.5) sijaan. Tällöin jokaisella aikapäivitysaskeleella lasketaan painot kaikien mahdollisten tila-aika-avaruuspolkujen yli. Samoin kuin uudelleenotanta tämä pienentää painojen varianssia, mutta lisää algoritmin aikakompleksisuutta.

2.1.4 Aikakompleksisuus

Algoritmin perusmuodon aikakompleksisuus on $\mathcal{O}(N)$. Uudelleenotantamenetelmän tai ehdotusjakauman valinta ei suoraan vaikuta aikakompleksisuuteen. Sen sijaan marginalisointi tärkeytyspainolla (2.20) lisää algoritmin aikakompleksisuutta $\mathcal{O}(N) \rightarrow \mathcal{O}(N^2)$, koska jokaisen partikkelin kohdalla painot lasketaan jokaisen tila-aika-avaruuspolun yli. On selvää, että erityisesti isoilla otoskoon N arvoilla ei yllä esitetty marginalisointi enää ole mielekästä, erityisesti reaalialkaisissa sovellutuksissa.

Tällaisia tilanteita varten algoritmista on olemassa $\mathcal{O}(N \log(N))$ -versioita, jotka perustuvat esimerkiksi N :n kappaleen oppimiseen (*N-body learning*). Näiden algoritmien käsitteily jää tämän tutkielman ulkopuolelle, mutta katsauksen algoritmeista ovat esittäneet esimerkiksi Klaas & al. artikkelissa “Toward Practical N^2 Monte Carlo: the Marginal Particle Filter” (2005) [19].

2.2 Saapasremmisuodin

Saapasremmisuodin 4 eli *bootstrap filter* on SIR-algoritmin muunnelma, jossa tärkeytysotannanssa (kts. 1) käytetään dynaamista mallia $p(x_k|x_{k-1})$.

Algoritmi 4: Saapasremmisuodin

Result: Posteriorijakauman $p(x_{1:k}|y_{1:k})$ estimaatti.

Data: Havainnot y_k . Generoitu $x_1^i \sim p_{x_0}$ missä $i = \{1, \dots, N\}$ ja jokainen partikkeli saa saman painon $w_{1|0}^i = 1/N$.

begin

```
for k = {1, 2, ..., T} do
    for i = {1, 2, ..., N} do
        begin
            Luodaan uudet estimaatit dynaamisesta mallista  $x_k^i \sim p(x_k|x_{k-1}^i)$ ;
        begin
            Päivitetään hiukkasten painot  $w_k^i$  uskottavuusfunktion  $p(y_k|x_k^i)$ 
            mukaan.
        begin
            Estimoidaan  $p$  laskemalla tiheydelle approksimaatio
             $\hat{p}(x_{1:k}|y_{1:k}) = \sum_{i=1}^N w_{k|k}^i \delta(x_{1:k} - x_{1:k}^i)$ .
        if k < t then
            begin
                Aikapäivitys. Suoritetaan uudelleenotanta kuten SIR-algoritmissa
                3.
```

Saapasremmisuodin on edellä esitettyä SIR-algoritmia yksinkertaisempi toteuttaa, mutta epäinformatiivisen tärkeytysjakauman vuoksi algoritmi saattaa vaatia SIR-algoritmia suuremman määrään hiukkasia. Saapasremmisuodin esitetään tässä sen historiallisen tärkeyden vuoksi, sillä kyseessä oli ensimmäinen uudelleenotataantaa hyödyntävä hiukkassuodinalgoritmi. Suotimien käytännön toteutukseen palataan luvussa 4.

2.3 Varianssin estimoinnista

Hiukkassuotimen varianssin estimoinnissa ollaan kiinnostuneita jakaumaestimaatin $\hat{p}(x_{1:k}|y_{1:k})$ varianssista. Yksinkertaisin tapa estimoida hiukkassuodinalgoritmin varianssia on ajaa algoritmi $M > 1$ kertaa. Koska ajot ovat toisistaan riippumattomia, voidaan estimaatin varianssi laskea kullekin aika-askeleelle k näiden ajojen k -hetken estimaattien otosvarianssina:

$$\hat{\sigma}_{MC}^2 = \text{Var}(\hat{p}(x_{1:k}|y_{1:k})) = \frac{1}{M-1} \sum_{i=1}^M (x_k^i - \bar{x}_k)^2, \quad (2.21)$$

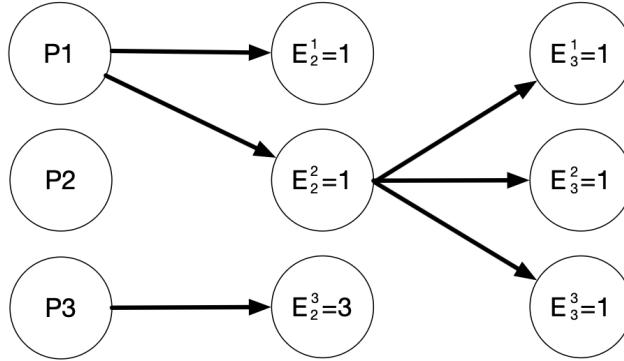
missä x_k^i on k :nen aika-askeleen piste-estimaatti ajolle $i = 1, \dots, M$ ja \bar{x}_k pisteeestimaattien aritmeettinen keskiarvo laskettuna kaikkien M ajon yli. Tällaisen Monte Carlo -varianssin estimointinen on kuitenkin laskennallisesti tehotonta. Monissa käytännön sovellutuksissa jo yhden hiukkassuodinalgoritmin ajaminen vaatii runsaasti laskentatehoa, jolloin Monte Carlo -varianssin laskeminen ei ole mahdollista.

Varianssia ei voida myöskään laskea analyttisesti, mutta koska keskeisen raja-arvolauseen (2.18) nojalla tiedetään, että asymptoottinen varianssi

$$\lim_{N \rightarrow \infty} N \text{Var}(\hat{g}(x_k^i)) \quad (2.22)$$

on olemassa, on sen estimointiin kehitetty lähi vuosina joitakin approksimatiivisia menetelmiä. Alla käsitellään Chan ja Lain (2013) [4] sekä Leen ja Whitleyn (2018) ehdottamaa varianssin estimointitapaa [20].

Ajetaan SIR-algoritmi kuten esitetty algoritmissa 3, mutta merkitään kyllakin aika-askeleella $k = 1, \dots, T$ jokaisen partikkelin $n = 1, \dots, N$ kohdalla indeksillä E_k^n kunkin partikkelin kantaisää eli sitä partikkelia, josta kyseinen partikkeli on uudelleenotantojen kautta polveutunut aika-askeleesta $k = 1$ lähtien. Kaavio 2.1 havainnollistaa tästä partikkalien polveutumista.



Kuva 2.1: Esimerkki Eeva-indekseistä, kun $T = 3$ ja $N = 3$.

Varianssiestimaatti voidaan laskea näiden, Leen ja Whitleyn Eeva-indekseiksi nimeämien indeksien perusteella seuraavasti. Merkitään ensin Eeva-indekseillä painotettua summaa

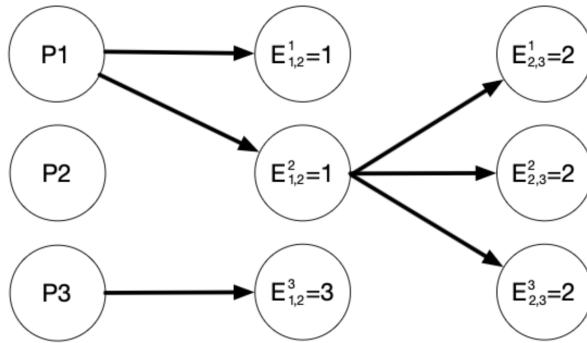
$$\hat{\mu}_k = \sum_{i=1}^N w_k^i x_k^i [E_k], \quad (2.23)$$

missä merkintä $x_k^i [E_k]$ tarkoittaa, että partikkeleista x_k valitaan Eeva-indeksien osoittamat partikkelimat eli ne partikkelimat joille $i = E_k^j$, kun $j = 1, \dots, N$. Sama partikkeli voidaan luonnollisesti valita summaan useamman kerran. Tämän perusteella lasketaan edelleen varianssi

$$\hat{\sigma}_{CLE}^2 = \frac{1}{N(N-1)} \sum_{i=1}^N w_k^i (x_k^i [E_k] - \hat{\mu}_k)^2. \quad (2.24)$$

Kyseessä harhaton ja konsistentti asymptoottisen varianssin estimaatti. Tarkemmin $N\hat{\sigma}_{CLE}^2$ konvergoi asymptoottiseen varianssiin (2.22). Tätä ei tutkielman puitteissa todisteta.

Yllä esitetty varianssiestimaatti saattaa kuitenkin kärsiä indeksien ehtymisestä. Toisin sanoen kun $k \rightarrow \infty$, polveutuvat kaikki hiukkaset lopulta samasta kantaisästä eli indeksit E_k^1, \dots, E_k^N ovat kaikki yhtäsuuria. Tämän vuoksi on mielekästä johtaa indeksit ainoastaan tietystä aiemmasta aika-askeleesta aika-askeleen $k = 1$ sijaan. Olsson ja Douc (2019) [27] ehdottavat tähän tarkoitukseen Henok-indeksiä $E_{m,k}^n$, jossa m merkitsee aika-askeleen $m \leq k$ partikkelia, josta kyseinen partikkeli polveutuu. Partikkelen kantaisien sukupolvi määritetään viipeällä λ niin, että $m = k - \lambda$. Kaavio 2.2 havainnollistaa partikkelen polveutumista, kun $\lambda = 1$.



Kuva 2.2: Esimerkki Henok-indekeistä, kun $T = 3$, $N = 3$ ja $\lambda = 1$.

Nyt varianssi voidaan laskea kuten yhtälössä (2.24) edellä, mutta Eeva-indeksit korvataan Henok-indekseillä. Saadaan painotettu summa

$$\hat{\mu}_k = \sum_{i=1}^N w_k^i x_k^i [E_{k(\lambda)}] \quad (2.25)$$

ja varianssi

$$\hat{\sigma}_{OD}^2 = \frac{1}{N-1} \sum_{i=1}^N w_k^i (x_k^i [E_{k(\lambda)}] - \hat{\mu}_k)^2, \quad (2.26)$$

missä $k(\lambda) := k - \lambda$ eli se aika-askelel, jota käytetään hiukkasten kantaisänä. Viive λ on varianssiestimaatin suunnitteluparametri. Pienillä viipeillä estimaatti on harhainen, mutta harha laskee viipeen kasvaessa. Olsson ja Douc suosittavat viipeen ylärajaksi arvoa $\lambda = 20$, jolloin estimaattorin harha on käytännössä kokonaan hävitetty. Tätä suuremmilla viivearvoilla estimaatti voi alkaa kärsiä samasta epätarkkuudesta kuin Eeva-indekeihin perustuva CLE-estimaatti.

Mastrotaro ja Olsson (2023) [22] laajentavat estimaattia edelleen niin, että viive λ valitaan mukautuvasti. Mastrotaron ja Olssonin ALVar-estimaatti (*Adaptive-Lag Variance*) lasketaan kuten OD-variansi (2.26) edellä, mutta kunkin algoritmin ajokerran jälkeen asetetaan seuraavan ajanhetken λ seuraavasti:

$$\lambda_{k+1} \leftarrow \arg \max_{\lambda \in [0, \lambda_k + 1]} \hat{\sigma}_{k+1, \lambda}^2 \quad (2.27)$$

ja ehtyneet Henok-indeksit määritellään niin, että indeksi on ehtynyt (so. kaikki indeksivektorin indeksit ovat yhtäsuuria), jos jompikumpi seuraavista ehdoista täytyy:

- i) Henok-indeksit $E_{m,k-1}^i$ ovat ehtyneet,
- ii) Henok-indeksit $E_{m-1,k}^i$ ovat ehtyneet ja on olemassa $\lambda' \in [0, \lambda - 1]$, joka täyttää ehdon $\hat{\sigma}_{k+1,\lambda}^2 < \hat{\sigma}_{k+1,\lambda'}^2$, koska myös ehtyneiden indeksien kantaisät ovat ehtyneitä.

Nyt indeksi λ_{k+1} voidaan valita rekursiivisesti niin, että se tuottaa suurimman varianssiestimaatin, joka on kuitenkin rajoitettu ylhäältä arvoon $\lambda_k + 1$. Tämä indeksi ei ole koskaan ehtynyt, joten se on myös parhaan varianssiestimaatin tuottava valinta.

Esitettyjä varianssiestimaatteja voidaan hyödyntää paitsi algoritmien ja parametrivalintojen vertailussa myös mukautuvan viipeen hiukkassiloittimen viipeen valinnassa (kts. luku 3). Varianssiestimaatti lasketaan myös luvun 4 empiirisessä esimerkissä.

Luku 3

Hiukkassilottimet

Tässä luvussa käsitellään suodinongelman läheisesti liittyvän siloitteluongelman ratkaisemista hiukkassiloitinalgoritmien avulla. Kuten hiukkassuotimien kohdalla, myös tässä luvussa esitetään ongelma ensin yleisessä Bayesilaisessa muodossa, jonka jälkeen siirrytään käsittelemään hiukkasmenetelmiin pohjautuvia siloitinalgoritmeja. Luvussa käsiteltävät algoritmit jaetaan kahteen pääkategoriaan, offline-algoritmeihin, joita sovelletaan hiukkassuodinalgoritmin ajon jälkeen ja online-algoritmeihin, jotka suoritetaan hiukkassuodinalgoritmin rinnalla.

Siloitinongelman esittely seuraa Särkkää (2013) [30]. Algoritmien käsiteily pohjautuu SIR- ja BS-PS-siloittimien osalta Särkkää (2013) [30] ja SIR-siloittimen osalta Kitagawan artikkeliin “Monte Carlo filter and smoother for non-Gaussian nonlinear state space models” (1996) [18]. Kiinteän viipeen silotin seuraa niin ikään Kitagawaa (1996) [18]. Uudelleenpainottava silotin perustuu Doucetin &al. artikkeliin “On sequential Monte Carlo sampling methods for Bayesian filtering” (2000) [11]. Mukautuvan viipeen silotin seuraa puolestaan Alenlövin ja Olssonin artikkelia “Particle-Based Adaptive-Lag Online Marginal Smoothing in General State-Space Models” (2019) [1].

3.1 Bayesilainen silotin

Bayesilaisen siloittimen tarkoitus on laskea tilan x_k marginaaliposteriorijakauma $p(x_k|y_{1:T})$ aika-askeleella k , kun käytössä on havaintoja aika-askeleeseen T asti, missä $T > k$. Ero Bayesilaiseen suotimeen (kts. alaluku 1.5) on siinä, että suodinongelmassa havaintoja on saatavilla ainoastaan aika-askeleeseen k asti, kun taas siloitinongelmassa myös tulevat havainnot ovat saatavilla. Ajassa taaksepäin etenevät rekursiiviset yhtälöt ongelman ratkaisemiseksi voidaan esittää muodossa

$$p(x_{k+1}|y_{1:k}) = \int_{\mathbb{R}^{n_x}} p(x_{k+1}|x_k)p(x_k|y_{1:k}) dx_k \quad (3.1)$$

ja

$$p(x_k|y_{1:T}) = p(x_k|y_{1:k}) \int \frac{p(x_{k+1}|x_k)p(x_{k+1}|y_{1:T})}{p(x_{k+1}|y_{1:k})} dx_{k+1}, \quad (3.2)$$

missä $p(x_k|y_{1:k})$ on suodintiheys aika-askeleella k ja $p(x_{k+1}|y_{1:k})$ prediktiivinen jakaus ajanhetkelle $k+1$. Kuten suodinongelman kohdalla, voidaan ongelma ratkaista suljetussa muodossa, kun mallit ovat lineaarisia. Tällöin kyseessä on Rauch-Turn-Striebel-siloitin (RTSS), josta käytetään myös nimitystä Kalman-siloitin. Samoin kuten Kalman-suotimen kohdalla, ongelma voidaan tiettyjen ehtojen vallitessa lineaarisoida. Näitä linearisoituja siloittimia ei käsitellä tässä tutkielmassa. Hiukkassuotimen tavoin hiukkassiloitin ratkaisee siloitteluongelman mille hyvänsä epälineaariseelle mallille.

3.2 Offline-algoritmit

Tässä alaluvussa käsitellään lyhyesti muutamaa ehdotettua offline-hiukkassiloitinalgoritmia. Offline-siloittimet estimoivat siloittintiheyttä aika-askeleella $k < T$, kun havaintodata on käytössä koko ajanjaksona $1 \dots T$. Alla esitetyt algoritmit siis olettavat, että kaikki mahdollinen tuleva data on jo niiden käytössä.

3.2.1 SIR-siloitin

Kuten aiemmin mainittua, näyttää hiukkassuodinalgoritmit, erityisesti SIR-algoritmi 3, ratkaisevan siloitteluongelman ilmaiseksi, kunhan tallennamme aika-askeleella k koko otoshistorian $x_{0:k}^i$. Tällöin voimme estimoida täytä siloitteluposteriorijakaumaa seuraavasti:

$$p(x_{0:T}|y_{1:T}) \approx \sum_{i=1}^N w_T^i \delta(x_{0:T} - x_{0:T}^i). \quad (3.3)$$

Nyt aika-askeleen k siloitinkuvaaja saadaan laskettua

$$p(x_k|y_{1:T}) \approx \sum_{i=1}^N w_T^i \delta(x_k - x_k^i), \quad (3.4)$$

missä x_k^i on $x_{0:T}^i$:n k :s elementti. Koska uudelleenotanta hävittää otoshistorian, pitää uudelleenotanta suorittaa koko otoshistoriasta $x_{0:k}^i = (x_{0:k-1}^i, x_k^i)$ pelkän aika-askeleen k otoksen x_k^i sijaan. Koska nyt koko otoshistoria pitää tallentaa, vaatii SIR-siloitin NkT muistia N sijaan. Vastaavasti myös uudelleenotannan aikakompleksisuus kasvaa [18].

SIR-siloittimen suurin ongelma on kuitenkin sen tuottamien estimaattien epätkkuus. Kun aika-askeleiden määrä kasvaa, johtaa koko otoshistorian uudelleenotanta kaiken painon kasautumiseen historian tietylle otoksiin, jolloin SIR-siloittimen tuottamat estimaatit eivät enää estimoit haluttua (siloitteluposteriorijakaumaa [18].

3.2.2 BS-PS-siloitin

Backward-simulation particle smoother (BS-PS) eli taaksepäin simuloiva hiukkassiloitin estimoii hiukkassuotimen tulosten perusteella tehokkaammin siloitinjakaumaa. Tässä algoritmissa hiukkasten historia simuloidaan aika-askeleesta T taaksepäin ensimmäiseen aika-askeleeseen:

Algoritmi 5: Taaksepäin simuloiva hiukkassiloitin

Result: Posteriorisiloitinjakauman $p(x_k|y_{1:T})$ estimaatti.

Data: Suodinkakaumia edustavat hiukkaset ja näihin liittyvät painot w_k^i, x_k^i , missä $i = 1, \dots, N$ ja $k = 1, \dots, T$.

```

begin
  begin
    Valitaan  $\tilde{x}_T = x_T^i$ 
    for  $k = \{T - 1, \dots, 0\}$  do
      begin
        Lasketaan uudet painot
         $w_{k|k+1}^i \propto w_k^i p(\tilde{x}_{k+1}|x_k^i)$ 
      begin
        Valitaan  $\tilde{x}_k = x_k^i$  todennäköisyydellä  $w_{k|k+1}^i$ .
    
```

Nyt siloittelujakaumaa voidaan estimoida seuraavasti:

$$p(x_{0:T}|y_{1:T}) \approx \frac{1}{S} \sum_{i=1}^N \delta(x_{0:T} - \tilde{x}_{0:T}^i), \quad (3.5)$$

missä $S, j = 1, \dots, S$ on algoritmin 5 toistokertojen määrä. Koska $\tilde{x}_{0:T}^j$ pitää sisällään kaikki otospolut, saadaan marginaalijakauma aika-askeleella k yhtälöstä (3.5) yksinkertaisesti valitsemalla sen k :net elementit. Sekä algoritmin aikakompleksisuus että muistivaade on $\mathcal{O}(STN)$.

3.2.3 Uudelleenpainottava hiukkassiloitin

Uudelleenpainottavassa hiukkassiloittimessa (tunnetaan myös nimellä marginaalihiukkassiloitin, kts. Doucet & al. (2000) [11]) siloitinjakaumaa estimoidaan käyttämällä SIR-hiukkassuodinalgoritmista 3 saatuja hiukkasia, mutta ne painotetaan uudelleen käyttäen dataa aika-askeleesta T alkaen, edeten ajassa taaksepäin.

Algoritmi 6: Uudelleenpainottava hiukkassiloitin

Result: Posteriorisiloitinjakauman $p(x_k|y_{1:T})$ estimaatti.

Data: Suodinjakaumia edustavat hiukkaset x_k^i ja näihin liittyvät painot w_k^i , missä $i = 1, \dots, N$ ja $k = 1, \dots, T$.

begin

begin

 Asetetaan $w_{T|T}^i = w_T^i$, jokaiselle $i = 1, \dots, N$;

for $k = \{T - 1, \dots, 0\}$ **do**

begin

 Lasketaan uudet painot

$$w_{k|T}^i = \sum_j w_{k+1|T}^j \frac{w_k^i p(x_{k+1}^j | x_k^i)}{\sum_l w_k^l p(x_{k+1}^l | x_k^l)}$$

Halutun siloitinjakauma estimaatti aika-askeleella k saadaan painotettuna keskiarvona $p(x_k|y_{1:T}) \approx \sum_i w_{k|t}^i \delta(x_k - x_k^i)$. Algoritmin aikakompleksisuus on $\mathcal{O}(N^2)$, minkä vuoksi uudelleenpainottava hiukkassiloitin on usein SIR-siloitinta tehokkaampi ratkaisu.

3.3 Online-algoritmit

Yllä esitettyt offline-algoritmit ratkaisevat siloitinongelman niin, että kaikki data ajanhetkeen T asti on saatavilla. Käytännössä siloitin siis ajetaan suodinalgoritmin jälkeen. Käytännön sovellutuksissa tämä ei ole aina mahdollista, jos siloittelujakauman pitää olla saatavilla reaalialkaiseksi. Online-siloittimet ratkaisevat siloitinongelman niin, että saatavilla on dataa aika-askeleeseen $k + L \leq T$ asti, missä L on dataan lisätty L :n aika-askeleen viive. Online-algoritmit voidaan edelleen jakaa kiinteän viipeen siloittimiin (*fixed-lag smoother*) ja mukautuvan viipeen siloittimiin (*adaptive-lag smoother*). Nimensä mukaisesta kiinteän viipeen siloitinalgoritmessa viive L valitaan suunnitteluparametrina, kun taas mukautuvan viipeen siloittimet pyrkivät valitsemaan optimaalisen viipeen johonkin laskennalliseen kriteeriin perustuen.

3.3.1 Kiinteän viipeen siloitin

Yksinkertaisin tapa toteuttaa kiinteän viipeen siloitin on käyttää SIR-siloitinta niin, että maksimiaika-askel T korvataan valitulla viipeellä $k + L \leq T$ [18]. Nyt yhtälön (3.3) jakauma saadaan muotoon

$$p(x_{0:(k+L)}|y_{1:(k+L)}) \approx \sum_{i=1}^N w_{k+L}^i \delta(x_{0:(k+L)} - x_{0:(k+L)}^i) \quad (3.6)$$

ja nykyisen aika-askeleen k siloitinjakauma lasketaan tästä jakaumasta kuten SIR-siloittimessa (kts. yhtälö (3.4)). Kiinteän viipeen siloitin välttää SIR-siloittimen approksimaatio-ongelmat. Kun viipeelle L pätee $k + L \ll T$ parantaa viipeen pidentäminen tiettyyn pisteesseen asti jakauman approksimaatiota. Kitagawa (1996)

suosittelee 10–20 aika-askeleen viivettä ja esittää 50 aika-askelta viipeen ylärajaksi [18]. Paremman estimaatin vastapainona pidemmän viipeen valinta lisää myös viivettä, joka dataa tuottavaan järjestelmään pitää lisätä. Siloittimien tulokset ovat saatavilla vasta L :n aika-askeleen jälkeen, mikä ei aina ole käytännössä mahdollista tai haluttua. Pidempi viive myös lisää algoritmin muistivaatimuksia, joskin muistivaatimukset pysyvät aina pienempinä kuin SIR-siloittimessa.

Kiinteän viipeen siloitinta (viipeellä $L = 1$) voidaan hyödyntää myös prediktivisenä siloittimena, jossa siloittelujakaumaa $p(x_{0:(k+1)}|y_{1:(k+1)})$ käytetään suodinjakauman $p(x_{1:(k)}|y_{1:k})$ laskennassa [25]. Ydinajatuksena on muokata SIR-algoritmia 3 niin, että aika-askeleen k painoja w_k^i painotetaan edelleen seuraavasta aika-askeleesta $k + 1$ lasketuilla painoilla. Näin algoritmi painottaa jo nykyhetkessä niitä hiukkasia, joiden uskottavuus on seuraavalla aika-askeleella suurempi. Tämä prediktivinen siloitin voidaan toteuttaa lisäämällä SIR-algoritmiin painotusvaiheen jälkeen seuraava ala-algoritmi:

Algoritmi 7: Prediktivinen siloitin (viive=1)

Result: Prediktivisellä siloittimella lasketut painot painotettu \tilde{w}_k^i .
Data: Viipeen $L = 1$ avulla saadut havainnot y_{k+1} . Partikkelit x_k^i ja niitä vastaavat painot w_k^i .

```

begin
  for  $i = \{1, 2, \dots, N\}$  do
    begin
      | Luodaan simuloidut hiukkaset  $\tilde{x}_{k+1}^i$  ehdotusjakaumasta  $q(\tilde{x}_{k+1}^i|x_k^i, y_{k+1})$ 
    |
    begin
      | Lasketaan simuloiduille hiukkasille painot  $\tilde{w}_{k+1}^i$ 
    |
    begin
      | Päivitetään nykyiset painot  $\tilde{w}_k^i = w_k^i \tilde{w}_{k+1}^i$ 
    |
    begin
      | Korvataan nykyiset painot  $w_k$  siloitetuilla painoilla  $\tilde{w}_k^i$ ;
  |

```

Kun hiukkasten määrä N pysyy samana, lisää prediktivinen siloitin suodinjakauman laskemisen tarkkuutta. Vastaavasti prediktivinen siloitin mahdollistaa saman suodinjakauman estimaatin tarkkuuden kuin SIR-algoritmi pienemmällä määräällä hiukkasia, kuitenkin vain tuplalten uskottavuusfunktiota laskettaessa vaadittavan laskentatehon ja muistitarpeen.

3.3.2 Mukautuvan viipeen siloitin

Yllä esitetyssä kiinteän viipeen siloittimessa on valittu viive L algoritmin suunnitteluparametri. Valittu viive on aina kompromissi: liian suuri viive kasvattaa siloitinjakauman estimaatin epätarkkuutta ja hidastaa laskentaa, kun taas liian pieni viive saattaa johtaa niin ikään epätarkkuuteen. Lisäksi valittu viive ei välttämättä johda jokaisella aika-askeleella optimaaliseen tai edes hyvään laskentatulokseen. Mukautuvan viipeen siloittimet yrityvät ratkaista tämän ongelman mukauttamalla

kunakin ajanhetkenä valittua viivettä johonkin laskennalliseen kriteeriin perustuen. Eräään version mukautuvan viipeen siloittimesta esittävät Alenlöv ja Olsson artikkelissa “Particle-Based Adaptive-Lag Online Marginal Smoothing in General State-Space Models” (2019) [1]. Siloitin hyödyntää hiukkassuotimen varianssiestimaattia viipeen valinnassa.

Yksinkertaisin versio siloittimesta on esitetty algoritmissa 8. Perusidea on viivästyttää aika-askeleen k siloitinjakauman luomista, kunnes tarjolla on viipeet $S = 1, \dots, s$, joiden varianssi

$$\sigma_{s|k}^2 = \sum_{i=1}^N \frac{w_k^i}{\Omega_k} \left\{ \tilde{x}_{s|k} - \sum_{j=1}^N \frac{w_k^j}{\Omega_k} \tilde{x}_{s|k} \right\}^2 \quad (3.7)$$

pysyy tietyn valitun rajan ϵ yläpuolella, missä $\tilde{x}_{s|k}$ on kyseiselle viipeellä laskettu marginaalisiloitinjakauman painovektori (kts. algoritmi 6). Kun tämä ehto ei enää täyty, käytetään suurimmalle kriteerin $\sigma_{s|k}^2 < \epsilon$ täyttämälle viipeelle laskettuja painoja siloitinjakauman estimointiin kaikilla $k' \geq k$. Varianssin estimoinnista kts. alaluku 2.3.

Algoritmi 8: Mukautuvan viipeen siloitin

Result: Siloittelijakauman estimaatti viipeellä s , tarkemmin $\sum_i^N w_k^i \tilde{x}_{s|k}^i \Omega_k$.

Data: Olkoon S joukko kullakin viipeellä s laskettuja painoja $\tilde{x}_{s|k}^i$. Alustetaan

$$S \leftarrow \emptyset.$$

```

begin
  for  $k = \{1, 2, \dots, T\}$  do
    begin
       $\lfloor$  Ajetaan SIR-algoritmi 3 aika-askeleella  $k$ 
    begin
       $\lfloor$  Jokaiselle  $s \in S$  lasketaan painovektori kuten algoritmissa 6.
    begin
       $\lfloor$   $S \leftarrow S \cup \{s\}$  Jokaiselle  $s \in S$  lasketaan painovektori kuten
       $\lfloor$  algoritmissa 6.
    begin
       $\lfloor$  Jokaiselle  $s$  lasketaan varianssiestimaatti  $\hat{\sigma}_{s|k}^2$ . Jos  $\hat{\sigma}_{s|k}^2 < \epsilon$  poistetaan
       $\lfloor$   $s$  joukosta  $S$  ja käytetään siloitinjaukauman estimaattia
       $\lfloor \sum_i^N w_k^i \tilde{x}_{s|k}^i \Omega_k$  kaikille aika-askeleilla  $k' \geq k$ .

```

Myös tähän siloittimeen liittyy suunnitteluparametrien valinta. Vaikka itse viivettä L ei valita, pitää parametri ϵ valita. Pienempi ϵ tuottaa suurempia viipeitä ja tätten parempia estimaatteja, mutta on myös laskennallisesti sekä muistin käytöltään raskaampi. Alenlöv ja Olsson ehdottavat ϵ -arvoja väliltä $(.5, 10^{-3})$.

Luku 4

Hiukkassuodin ja -siloitin sisätilapaikannuksessa

Sisätilapaikannus tarkoittaa nimensä mukaisesti ihmisten tai esineiden automaattista paikantamista sisätiloissa. Koska GPS-järjestelmät toimivat sisätiloissa huonosti tai eivät lainkaan, tarvitaan rakennusympäristöihin muita paikannusratkaisuja. Tässä luvussa käydään läpi sisätilapaikannuksen yleisiä periaatteita sekä joitakin ehdotettuja ratkaisuja. Tämän jälkeen keskitytään Walkbase-ohjelmistoyrityksen sisätilapaikan-nusteknologian tarpeisiin kehitettyyn hiukkassuodinalgoritmiin, jonka tarkkuutta ja tehokkuutta testataan koeympäristössä.

4.1 Sisätilapaikannuksesta

Sisätilapaikannus tarkoittaa teknisiä ratkaisuja ja menetelmiä, joilla paikannetaan ihmisiä, laitteita tai esineitä sisätiloissa, joissa perinteinen GPS-signaali ei ole riittävä tai saatavilla. Sisätilapaikannuksessa hyödynnetään useita erilaisia teknologioita, kuten radiotaajuksia, magneettikenttiä, ultraääntä ja optisia menetelmiä. Kattava esitys sisätilapaikannukseen käytettävistä teknologioista löytyy Oguntala & al. artikkelista “Indoor location identification technologies for real-time IoT-based applications: An inclusive survey” (2018) [26], johon myös tämä alaluku perustuu.

Optisessa paikannuksessa hyödynnetään kameroita tai muita optisia sensoreita, kuten esimerkiksi Lidar-valotutkia halutun kohteen sijainnin määrittämiseen. Magneettikenttiin perustuvat paikannusratkaisut hyödyntävät rakennusten sisäisten metallirakenteiden maapallon magneettikenttään luomia paikallisia muutoksia. Näitä muutoksia voidaan käyttää vertailutietona paikannuksessa, mittamalla ympäröivän magneettikentän vahvuutta ja vertaamalla sitä ennalta tallennettuihin karttoihin. Ultraäänipohjainen paikannus puolestaan hyödyntää äänialtojen kulkuaikaa lähettimen ja vastaanottimien välillä.

Yleinen teknologiavalinta sisätilapaikannuksessa ovat erilaiset Bluetooth-standardiin tai muuhun radioteknologiaan (kuten esimerkiksi millimetriaaltoihin) perustuvat ratkaisut. Näistä suurin osa hyödyntää lähettimen ja vastaanottimen/vastaanottimien välisiä radiosignaaleja laitteiden välisen etäisyyden tai kulman mittaamiseen. Radioteknologiaan perustuvilla järjestelmillä voidaan kotuullisin

kustannuksin saavuttaa hyvä tasapaino paikannustarkkuuden ja energiankulutuksen välillä. Radioteknologiaan perustuvat järjestelmät ovat myös yksityisyyden näkökulmasta yksinkertaisempia toteuttaa kuin esimerkiksi kameroiden avulla tapahtuvaan paikannukseen perustuvat järjestelmät.

4.2 Teknologian kuvaus

Turkulainen teknologia- ja analytiikkayritys Walkbase käyttää Bluetooth/BLE-sisätilapaikannusta asiakkaiden käyttäytymistä koskevan datan keräämiseen erityisesti ruokakaupoissa sekä tavarataloissa. Tyypillisessä asennusskenaariossa lähettimet (tagit) kiinnitetään liikkeen ostoskärryihin ja paikantimet kiinnitetään liiketilan kattoripustuksiin.

Walkbase on kehittänyt sisätilapaikannukseen oman laitteisto- ja ohjelmistoratkaisunsa, jonka tavoitteena on tarjota kaikissa ympäristöissä alle metrin paikannustarkkuus. Koska tagit kiinnitetään tunnetulle korkeudelle ostoskärryihin, riittää paikannustarkkuutta laskettaessa tarkastella ainoastaan leveys- ja pituusasteita. Kun tagin todellinen sijainti $P_k = (P_{\text{lon}_k}, P_{\text{lat}_k})$ tiedetään diskreetillä aika-askeleella k , voidaan sijaintiestimaatin $\hat{x}_k = (\hat{x}_{\text{lon}_k}, \hat{x}_{\text{lat}_k})$ paikannusvirhe ϵ_{pos_k} laskea

$$\epsilon_{\text{pos}_k} = d(P_k, \hat{x}_k) = \sqrt{(P_{\text{lon}_k} - \hat{x}_{\text{lon}_k})^2 + (P_{\text{lat}_k} - \hat{x}_{\text{lat}_k})^2}, \quad (4.1)$$

eli käyttää paikannusvirheenä yksinkertaisesti Euklidista etäisyyttä. Kun hyödynnetään myöhemmin alaluvussa 4.3.1 esitettävää lineaarista interpolaatiota, saadaan näin paikannusvirhe suoraan metreinä. Haluttu, alle metrin paikannustarkkuus saavutetaan, kun missä hyvänsä testiasetelmassa kaikkien asetelmaan liittyvien paikannusvirheiden 50. persentiili on $\leq 1\text{m}$.

Walkbasen paikannusratkaisu koostuu kolmesta eri laitteistokomponentista, AT-2-Bluetooth-lähetin-vastaanottimesta, jotka kiinnitetään ostoskärryihin, XR-2-Bluetooth-lähetin-vastaanottimista, jotka kiinnitetään tilan kattoripustuksiin sekä OSCU-laskentayksiköstä, joka luo paikkadataa XR-2-vastaanotinten tuottaman datan perusteella ja lähetää paikkadatan edelleen palvelinkeskukseen.

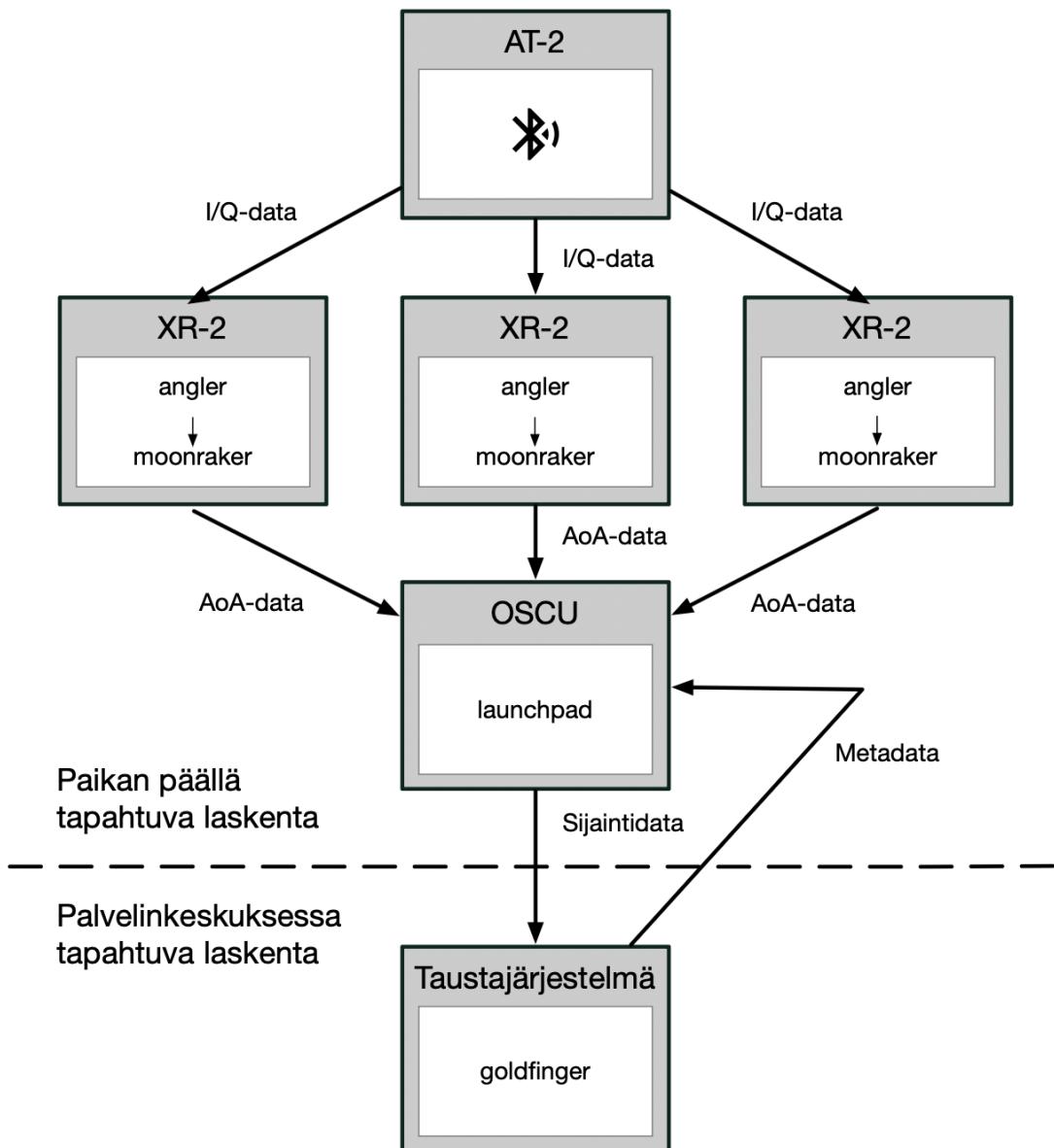
AT-2 on Bluetooth 5.1 (BLE) -strandardin mukainen lähetin-vastaanotin, joka toimii 2.4Ghz taajuusalueella. Walkbasen suunnittelemän laitteen PCB-kehäantenni kykenee lähetämään GFSK-moduloitua dataa 2Mbps nopeudella. Laite saa virtansa yhdestä CR-2477-paristosta. XR-2.1 on Bluetooth 5.1 (BLE) -strandardin mukainen lähetin-vastaanotin, joka toimii 2.4Ghz taajuusalueella. Walkbasen suunnittelemän laitteen 16 antennista koostuva antenniryhmä kykenee lähetämään GFSK-moduloitua dataa 2Mbps nopeudella. Laite saa virtansa ethernet-lähiverkosta 802.3af-standardin mukaisesti. Laitteen vaatima laskenta tapahtuu Raspberry Pi Compute Module 4 -piirilevytietokoneella. Lisäksi laite sisältää inertiamittausyksikön, jota voidaan käyttää asennetun laitteen kallistumis- ja nyökkäämiskulman (*roll* ja *pitch*) arviomiseen.

OSCU-laskentayksikkönä käytetään Ubuntu-käyttöjärjestelmällä toimivaa, paikannusjärjestelmää varten rakennettua tietokonetta. Koska OSCU-laskentayksikön

laskentateho on rajallista, on paikannusalgoritmin aikakompleksisuus yksi käytettävän algoritmin ydinkriteereistä.

Tarvittavasta laskennasta vastaava ohjelmistojärjestelmä koostuu neljästä ohjelmistokomponentista. C-ohjelmointikielellä toteutettu *angler* laskee AT-2-tagin lähetämän I/Q-datan perusteella signaalien tulokulman, Go-ohjelmointikielellä toteutettu *moonraker* lähetää tulokulmadatan paikallisverkon yli OSCU-laskentayksikölle, jossa Go-ohjelmointikielellä toteutettu *launchpad* luo siitä sijaintidataa, jonka se lähetää edelleen palvelinkeskuksen taustajärjestelmään.

Palvelinkeskuksen taustajärjestelmässä Go-ohjelmointikielellä toteutettu *goldfinger* prosessoi sijaintidatan Walkbasen analytiikka-alustan käyttämään muotoon. *Goldfinger* vastaa myös siitä, että kaikki *launchpad*-sovelluksen vaatima metadatta on *launchpad*-sovelluksen käytössä. Kaavio 4.1 kuvailee järjestelmän laitteisto- ja ohjelmistoarkkitehtuurit.

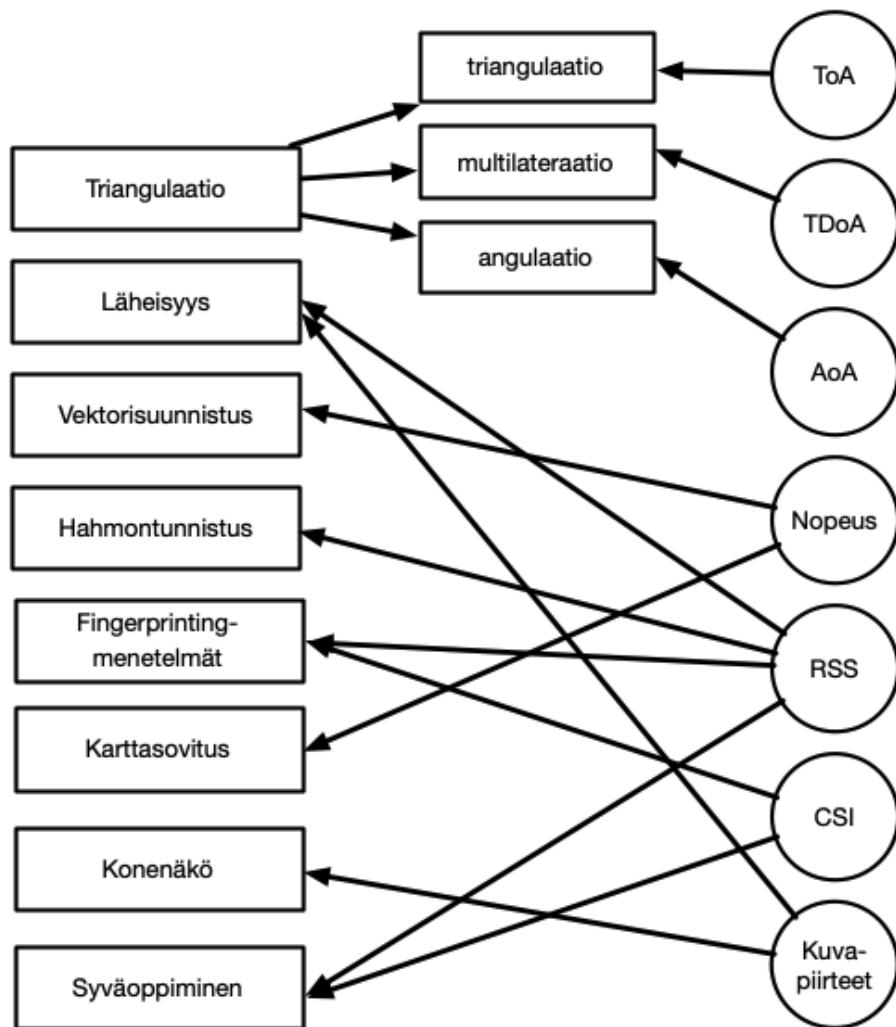


Kuva 4.1: Järjestelmäarkkitehtuuri

Tässä luvussa keskitytään *launchpad*-sovelluksen hyödyntämään paikannusalgoritmiin, mutta sitä ennen käsitellään lyhyesti tulokulman laskentamenetelmiä ja *angler*-sovelluksen toimintaa. Immateriaalioikeussyistä tutkielmanmassa ei hyödynnetä Go-ohjelointikielillelä toteutettua ohjelmakoodia. Sen sijaan paikannusalgoritmi on toteutettu R-ohjelointikielillelä.

4.2.1 AoA-menetelmistä

Riippuen käytetystä teknologiasta ja teknologian tuottamasta datasta, voidaan sisätilapaikannuksessa soveltaa lukuisia eri paikannustekniikoja ja -algoritmeja. Oguntalaa (2018) [26] mukailevassa kaaviossa 4.2 on esitetty pääpiirteittään sisätilapaikannusmenetelmät käytetyn datan (kaaviossa oikealla) mukaan jaoteltuna.



Kuva 4.2: Paikannusmenetelmien luokittelu

Walkbasen toteuttama paikannusratkaisu perustuu AoA-menetelmään. Tämä on paikannusteknologia, joissa lähettimen ja vastaanottimen välinen kulma estimoidaan signaalin saapumiskulman perusteella. Tässä estimoinnissa hyödynnetään signaalin vaihe-eroja, kun sama signaali vastaanotetaan usealla eri antennilla.

Walkbase käyttää radiosignaalien tulokulman estimoinnissa MUSIC-algoritmia (*Multiple Signal Classification*), joka arvioi tulokulmia analysoimalla signaalien auto-korrelatiomatriisia ja etsimällä sen ominaisvektoreita. Esitys MUSIC-algoritmista löytyy esimerkiksi Hayesin kirjasta *Statistical Digital Signal Processing and Modeling*

(1996) [16]. Guniam & al (2023) [14] puolestaan esittävät artikkelissa “Analysis and Design of a MuSiC-Based Angle of Arrival Positioning System” tapoja optimoida MUSIC-algoritmia soveltumaan monimutkaisiin, heijastuksia sisältäviin sisätilaympäristöihin.

Angler-sovellus hyödyntää MUSIC-algoritmin toteutuksessa omaa hiukkassuodinalgoritmiaan signaalisketrin analysointiin sekä tulokulmien estimointiin. MUSIC-algoritmia tai *angler*-sovelluksen hiukkassuodinalgoritmia ei käsitellä tarkemmin tämän tutkielman puitteissa.

4.2.2 Kalibraatiosta

Koska *angler*-sovellus laskee suuntimakulman aina antenniryhmän määräytystä reunaasta nähdent, on tärkeää, että laitteen asemointi karttapohjoiseen nähdent on tiedossa. Koska laitteen tarkan kulman arvioiminen asennusympäristössä on haastavaa eikä laite sisällä kompassia, kalibroidaan kukin laite SIR-kalibraatioalgoritmilla, jonka on mahdollista ottaa huomioon myös laitteen IMU-intertiamittausyksiköstä saatavat kallistumis- ja nyökkäämiskulmat.

Jos IMU-dataa hyödynnetään, tuottaa kalibraatioalgoritmi jokaiselle XR-2-laitteelle rotaatiomatriisiin \mathbf{R}_l . Muussa tapauksessa kalibraatioalgoritmi tuottaa jokaiselle laitteelle atsimuuttiakulman kohdistuksen η . Kalibraatioalgoritmia ei käsitellä tämän tutkielman puitteissa. Paikannusongelman yksinkertaistamiseksi myöskään IMU-yksikön tuottamaa dataa ei sisällytetä kalibraatioon ja jokaisen XR-2-laitteen oletetaan olevan lattiaan nähdent vaakatasossa. Atsimuuttikulman kalibraatiokohdista lisää seuraavassa alaluvussa 4.3.

4.3 Datan kuvaus

Koeasetelmaa varten AT-2-tagit on asetettu lähetämään I/Q-dataotoksia 6hz taajuudella, joka vastaa järjestelmän tuotantokäyttöä. Parempi sijaintitarkkuus saavutettaisiin korkeammalla taajuudella, mutta käytännön sovellutuksissa ei tagien haluttu akun kesto salli 6hz korkeampaa lähetystäajuuutta.

Angler-sovellus koostaa jokaisesta tagin lähetämästä dataotoksesta s atsimuuttieli suuntimakulman θ_s ja korkeuskulman γ_s , jotka noudattavat suuntimakulman osalta von Mises -jakaumaa ja korkeuskulman osalta katkaistua normaalijakaumaa:

$$\theta_s \sim \text{von Mises}(\mu_{\theta_s}, \kappa_k), \quad (4.2)$$

$$\gamma_s \sim \mathcal{N}_{\text{katkaistu}}(\mu_{\gamma_s}, \sigma_s^2). \quad (4.3)$$

Jakaumat on kuvattu tarkemmin alaluvussa 4.4.2. Jokaisen XR-2-laitteen *angler*-sovellus lisää dataan myös sekvenssinumeron, jonka perusteella samasta AT-2-tagin lähetämästä I/Q-dataoksesta lasketut usean eri vastaanottimen laskemat tulokulmat voidaan yhdistää samaan I/Q-dataotokseen. XR-2-laitteen lähetämä tulokulmadata on kuvattu alla.

Muuttuja	Kuvaus	Esimerkkiarvo
id	havainnon yksilöivä tunniste	317092
ts	havainnon aikaleima	2024-04-08 21:38:20.998+00
locator_mac	XR-2-laitteen MAC-osoite	2c:e3:10:00:07:a6
asset_tag_mac	AT-2-tagin MAC-osoite	2c:e3:10:00:63:89
sequence_nr	kulmadatan I/Q-dataotokseen yhdistäävä juokseva numerointi	2066
azimuth_location	atsimiuuttikulman θ jaukauman sijaintiparametri μ_θ (rad)	0.39
azimuth_scale	atsimiuuttikulman θ jaukauman skaalaparametri κ	80.98
elevation_location	jaukauman sijaintiparametri μ_γ (rad) korkeuskulman γ	0.13
elevation_scale	jaukauman skaalaparametri σ^2	0.012
quality_sndr	signaali-kohinasuhde	22.0
rssi	signaalin vahvuus (dBm)	-81
distance	arvioitu etäisyys lähettimeen (m)	18.6

Taulukko 4.1: Tulokulmamuuttujat

Etäisyys on estimoitu signaalin vahvuudesta käyttäen propagaatiomallia. Etäisyyttä tai signaalin vahvuutta ei käytetä paikantamiseen, joten tämän mallin käsittely jätetään tutkielman ulkopuolelle. Munoz (2009) luku 2 sisältää yleiskatsauksen propagaatiomalleista [24].

Launchpad-sovelluksessa tulokulmadataan yhdistetään XR-2-laitteen MAC-osoitteiden perusteella tarvittavat XR-2-laitteita koskevat metadatamuuttujat. Näihin kuuluvat laitteen korkeus, laitteen suuntimakulma ja karttakoordinaatit. Metadata on kuvattu taulukossa 4.2.

Muuttuja	Kuvaus	Esimerkkiarvo
locator_mac	XR-2-laitteen MAC-osoite	b8:27:eb:66:0d:2a
lat	vastaanottimen sijainti (leveyspiiri)	60.448265
lon	vastaanottimen sijainti (pituuspiirit)	22.294823
direction	suuntimakulma η (astetta)	34
height	vastaanottimen korkeus (m)	2.22

Taulukko 4.2: Metadata

Atsimiuuttikulma θ lasketaan aina vastaanottimen tietyltä sivulta, joten se vastaa napapohjoista ainoastaan siinä tapauksessa, että vastaanottimen kyseinen sivu on asetettu kohtisuoraan napapohjoiseen nähden. Käytännössä vastaanottimien asettaminen tiettyyn kulmaan ei ole aina mahdollista eikä vaihe-erojen mittaan

kannalta edes suotavaa. Tämän vuoksi jokaiselle vastaanottimelle on tietokantaan tallennettu oma suuntimakulma η . Toisin kuin tulokulmadatan kulmat, on tämä tallennettu tietokantaan astineina. Kokeessa käytetään napapohjoisesta laskettuja kulmia Φ , jotka lasketaan jokaiselle havainnolle havainnon vastaanottimen suuntimakulman avulla

$$\Phi = (\theta + \eta \times \frac{\pi}{180^\circ}) \mod 2\pi. \quad (4.4)$$

Lisäksi saatavilla on PostGIS-muotoon tallennettua polygonidataa, joka kuvaa koeympäristön pohjapiirrustusta sekä koeympäristössä esiintyviä liikkumisen estäviä kohteita, kuten hyllyjä tai pöytää. Näitä hyödynnetään alaluvun [4.4.4.2](#) karttasovitusalgoritmissa.

Havaintomuuttujien ohella koetilanteesta on tallennettu testipolku, jota pitkin AT-2-tagia liikutetaan koetilanteessa. Testipolkudata pitää sisällään karttaan piirrettyn janan päätet- ja sisäpisteet. Tallennettu sijainti perustuu koeympäristön lattiaan pohjapiirrustusten sekä laser-mittausten avulla tehtyihin merkintöihin. Näin saadut testimuuttujat on kuvattu taulukossa [\(4.3\)](#).

Muuttuja	Kuvaus	Esimerkkiarvo
path_lat	polkupisteen sijainti (leveyspiiri)	60.44819
path_lon	polkupisteen sijainti (pituuspiiri)	22.29493

Taulukko 4.3: Testimuuttujat

Alaluvussa [4.4.4.2](#) kuvatulla interpolatiolla metreiksi muutettuja testimuuttuja käytetään paikannusalgoritmien paikannusvirheen laskemisessa. Koska koeasetelmasa (kts. alaluku [4.5.1](#)) ei laitteen sijaintia tallenneta dataa kerättäessä, käytetään paikannusvirhettä laskettaessa janojen päätepisteiden joukosta tiivistettyä joukkoa. Paikannusvirheen laskenta on kuvattu algoritmissa [9](#).

Algoritmi 9: Paikannusvirheen laskeminen

Result: Aika-askeleen k paikannusvirhe ϵ_{pos_k} (m);
Data: Metreiksi muutetut testimuuttujat sisältävä pistejoukko P . Aika-askeleen k sijaintiestimaatti \hat{x}_k . Tihennysparametri s , vakioarvo $s = 1000$.

```
begin
  begin
    Tihennetään pistejoukko  $P$ . Lisätään joukosta  $P$  luodulle janoille pisteitä lineaarisella interpolatiolla, kunnes joukossa on  $s$  pistettä. Saadaan joukko  $S$ ;
  begin
    Asetetaan  $\hat{\epsilon}_{\text{pos}_k} = \infty$ ;
  for  $i = \{1, 2, \dots, s\}$  do
    begin
      Lasketaan sijaintiestimaatin  $\hat{x}_k$  ja pisteen  $S_i$  välinen Euklidinen etäisyys  $d_i$ .
      if  $d_i < \hat{\epsilon}_{\text{pos}_k}$  then
        begin
           $\hat{\epsilon}_{\text{pos}_k} = d_i$ ;
    begin
      Palautetaan aika-askeleen  $k$  paikannusvirhe  $\epsilon_{\text{pos}_k} = \hat{\epsilon}_{\text{pos}_k}$ ;
```

Algoritmiin palataan alaluvun 4.5 empiirisessä esimerkissä.

4.3.1 Karttaprojektiosta

Kaikki yllä esitetyssä datassa esiintyvät sijaintikoordinaatit on tallennettu tietokantaan WGS 84 -tasokoordinaattijärjestelmässä. Hiukkassuotimiin perustuva paikanusalgoritmia sovellettaessa on kuitenkin monin paikoin tarve syöttää parametreja metrijärjestelmässä. Tästä syystä leveys- ja pituusasteisiin perustuvat koordinaatit muunnetaan laskentaa varten metreiksi ja metreinä esitetyt sijaintitulokset muutetaan tulosten esittämistä varten takaisin WGS 84 -koordinaattijärjestelmään.

Muunnos tapahtuu lineaarisella interpolatiolla. Määritellään ensin kerrospolygonin rajausalue. Koska kaikki käytetyt koordinaatit ovat tämän rajausalueen sisällä, voidaan tästä rajausaluesta käyttää metrikonversiossa. Rajausalue koostuu neljästä kulmapisteestä. Poimitaan näistä pisteistä minimit ja maksimit sekä pituus- että leveyskoordinaateille. Näin saadaan neljä arvoa $B_{\text{lonlat}} = \{\text{lon}_{\min}, \text{lon}_{\max}, \text{lat}_{\min}, \text{lat}_{\max}\}$. Määritellään rajausalueen sivujen pituus metreinä geodeettisen etäisyyden avulla, jolloin saadaan kaksi metreissä laskettua etäisyyttä $D_m = \{d_{\text{lon}}, d_{\text{lat}}\}$. Käytetään näitä kulmapisteitä sekä metreinä laskettuja etäisyyksiä interpolointiaan koordinaatit WGS 84 -koordinaattijärjestelmästä metreissä esitetyille arvoalueille $[0, \max(x_m)]$ ja $[0, \max(y_m)]$ seuraavasti:

$$x_m = f(x_{\text{lon}}; B_{\text{lonlat}}, D_m) = \frac{x_{\text{lon}} - \text{lon}_{\min}}{\text{lon}_{\max} - \text{lon}_{\min}} \times d_{\text{lon}}, \quad (4.5)$$

$$y_m = f(y_{\text{lat}}; B_{\text{lonlat}}, D_m) = \frac{y_{\text{lat}} - \text{lat}_{\min}}{\text{lat}_{\max} - \text{lat}_{\min}} \times d_{\text{lat}}, \quad (4.6)$$

missä x_m vastaa leveyskoordinaatteja metreissä ja y_m pituuskoordinaatteja metreissä. Vastaavasti käänös takaisin WGS 84 -koordinaattijärjestelmään tapahtuu seuraavasti:

$$x_{\text{lon}} = f(x_m; B_{\text{lonlat}}, D_m) = \frac{x_m}{d_{\text{lon}}} \times (\text{lon}_{\max} - \text{lon}_{\min}) + \text{lon}_{\min}, \quad (4.7)$$

$$y_{\text{lat}} = f(y_m; B_{\text{lonlat}}, D_m) = \frac{y_m}{d_{\text{lat}}} \times (\text{lat}_{\max} - \text{lat}_{\min}) + \text{lat}_{\min}. \quad (4.8)$$

Empiirisessä esimerkissä jätetään takaisinmuunnot tekemättä, jotta koeympäristön sijaintia ei voi paikallistaa sen koordinaattien perusteella.

4.3.2 Muunnettua dataa

Kun dataan on tehty yllä esitetyt konversiot ja metadata on liitetty jokaisen aika-askeleen k dataan, saadaan data lopulliseen, empiirisessä esimerkissä käytettävään muotoon, jossa jokaisen aika-askeleen k kohdalla on käytettävissä seuraava data:

Muuttuja	Kuvaus	Esimerkkiarvo
id	havainnon yksilöivä tunniste	317092
ts	havainnon aikaleima	2024-04-08 21:38:20.998+00
locator_mac	XR-2-laitteen MAC-osoite	2c:e3:10:00:07:a6
asset_tag_mac	AT-2-tagin MAC-osoite	2c:e3:10:00:63:89
sequence_nr	kulmadatan I/Q-dataotokseen yhdistäävä juokseva numerointi atsimuuttikulman θ	2066
azimuth_location_mdf	jaukauman muunnettua sijaintiparametri Φ (rad)	0.39
azimuth_scale	atsimutikulman θ	80.98
elevation_location	jaukauman skaalaparametri κ	0.13
elevation_scale	korkeuskulman γ	0.012
quality_snrd	jaukauman sijaintiparametri μ_γ (rad)	22.0
rssi	signaali-kohinasuhde	-81
distance	arvioitu etäisyys lähettimeen (m)	18.6
x_m	vastaanottimen sijainti (x -koordinaatti, metreinä)	1.24
y_m	vastaanottimen sijainti (y -koordinaatti, metreinä)	0.78
height	vastaanottimen korkeus (m)	2.22

Taulukko 4.4: Muunnettua data

4.4 Sisätilapaikanusalgoritmi

4.4.1 Ongelman kuvaus

Tarkoituksena on estimoida liikkuvan AT-2-tagin sijaintia. Merkitään estimoitavaa tilasarjaa $x_{1:k} = \{x_1, \dots, x_k\}$, missä sijainnit on estimoitu sekunnin tarkkuudella. Lisäksi merkitään x_0 testilaitteen lähtösijaintia. Jokainen tilasarjan havainto koostuu suuntimakulmasta sekä pituus- että leveyskoordinaateista (x_k^x, x_k^y) . Määritellään tilalle liikkuvan AT-2-tagin kulkua kuvaava vektorisuunnistukseen (*dead reckoning*) perustuva dynaaminen malli (4.9)

$$x_{k+1} = f(x_k, \nu_k) = x_k + D_k \begin{bmatrix} \cos \psi_k \\ \sin \psi_k \end{bmatrix} + \nu_k, \quad (4.9)$$

missä D_k on AT-2-tagin sekunnin aika-askeleen $k \rightarrow k+1$ aikana kuljettama matka ja ψ_k AT-2-tagin liikkeen suuntimakulma kyseisellä aika-askeleella. ν_k on kohinaa,

joka syntyy mittausvirheestä ja jolle voidaan olettaa $\sim \mathcal{N}(\mu_x, \sigma_x^2)$. Jos laite on paikallaan, yksinkertaistuu malli muotoon $x_{k+1} = f(x_k) = \text{id}(x_k) = x_k$, missä $\text{id}(x)$ on identiteettifunktio.

Vastaavasti $y_{k_{i,j}} = \{y_{k_{1,1}}, \dots, y_{k_{n_s,z}}\}$ kuvaa AT-2-tagin ja XR-2-laitteiden välillä yhden sekuntin aika-askeleen aikana laskettuja kulmahavaintoja, missä $i = 1, \dots, n_s$ käy läpi kaikki n_s XR-2-laitetta ja $j = 1, \dots, z$ kustakin laitteesta vastaanotetut kulmahavainnot. z ilmaisee tässä AT-2-tagin lähetystaajuutta, joka koeasetelmissä on $z = 6$. Näin ollen jokainen havainto koostuu (maksimissaan) $6 \times n_s$ kappaleesta kulmahavaintoja.

Lisäksi tunnetaan XR-2-laitteisiin $\{s^1, \dots, s^{n_s}\}$ liittyvät leveys- ja pituuskoordinaatit (λ, ϕ) , jotka on muutettu alaluvussa 4.3.1 esitettyllä interpolatiolla metreiksi. Saadaan

$$u = \begin{bmatrix} \lambda^1 & \phi^1 \\ \lambda^1 & \phi^1 \\ \vdots & \vdots \\ \lambda^{n_s} & \phi^{n_s} \end{bmatrix}, \quad (4.10)$$

missä jokainen koordinaattipari on toistettu z kertaa. Määritellään nyt havaintomalli

$$y_{k_{i,j}} = h(x_k, u) + e_k = \text{atan2} \left(\begin{bmatrix} \phi^1 - x_k^y \\ \phi^1 - x_k^y \\ \vdots \\ \phi^{n_s} - x_k^y \end{bmatrix}, \begin{bmatrix} \lambda^1 - x_k^x \\ \lambda^1 - x_k^x \\ \vdots \\ \lambda^{n_s} - x_k^x \end{bmatrix} \right) + e_k, \quad (4.11)$$

missä

$$\text{atan2}(y, x) = \begin{cases} \arctan(\frac{y}{x}) & \text{jos } x > 0, \\ \arctan(\frac{y}{x}) + \pi & \text{jos } x < 0 \text{ ja } y \geq 0, \\ \arctan(\frac{y}{x}) - \pi & \text{jos } x > 0 \text{ ja } y < 0, \\ +\frac{\pi}{2} & \text{jos } x = 0 \text{ ja } y > 0, \\ -\frac{\pi}{2} & \text{jos } x = 0 \text{ ja } y < 0, \\ \text{ei määritelty} & \text{jos } x = 0 \text{ ja } y = 0 \end{cases} \quad (4.12)$$

ja kohina noudattaa moniulotteista normaalijakaumaa $e_k \sim \mathcal{N}(0, \Sigma)$.

Määrittelemätön atan2-tapaus, jossa $x = 0$ ja $y = 0$ on käytettyllä mittaustarkkuudella käytännössä mahdoton. Jos tapaus halutaan kuitenkin välittää, voidaan nolla-arvot tarpeen vaatiessa korvata joillakin hyvin lähellä nollaa olevalla arvolla.

Kumpikaan funktiosta $h(\cdot)$ ja $f(\cdot)$ ei ole lineaarinen, joten SIR-algoritmi on sopiva valinta ongelman ratkaisemiseksi. Koetuloksia arviodaan ensisijaisesti paikanusvirheen avulla. Paikannusvirhe ϵ_{pos_k} lasketaan jokaisen aika-askeleen k posteriorijakaumaestimaatista \hat{p}_k painotettuna keskiarvona

$$\epsilon_{\text{pos}_k} = d \left(P_k, \sum_{i=1}^N w_i^k x_k^i \right), \quad (4.13)$$

missä w_i^k on ajanketken k partikkeliiden normalisoitu paino ja $d(x_k^i, y_k)$ partikkeliiden ja testilaitteen todellisen sijainnin P_k välisen etäisyyden algoritmilla 9 laskeva funktio.

4.4.2 Uskottavuusmallit

Jokainen yhtä tulokulmaa vastaava *angler*-sovelluksen tuottama havaintodatarivi pitää sisällään neljä parametrimuuttuja `azimuth_location_mdf`, `azimuth_scale`, `elevation_location` ja `elevation_scale`. Koska *angler*-sovellus on kirjoitettu varta vasten tuottamaan dataa hiukkassuodinpaikannusalgoritmia varten, ovat nämä valmiiksi hiukkassuotimen uskottavuusmallin parametreja.

Angler-sovelluksessa XR-2-laitteen ja AT-2-tagin välinen atsimuuttikulma siimuloidaan von Mises -jakaumasta, jolloin muuttujat `azimuth_location_mdf` ja `azimuth_scale` vastaavat tämän jakauman sijainti- ja skaalaparametreja μ ja κ . Näistä edellistä voidaan pitää itse atsimuuttikulman estimaattina. Määritellään siis jokaiselle tulokulmalle h sekä XR-2-laitteelle l seuraava atsimuuttikulman uskottavuusmalli:

$$L_{\theta_{h,l}}(y_{h,l}|x_{h,l}; \mu_{h,l}, \kappa_{h,l}) = \frac{e^{\kappa_{h,l} \cos(x_{h,l} - \mu_{h,l})}}{2\pi I_0(\kappa_{h,l})}, \quad (4.14)$$

missä I_0 on 0:s ensimmäisen lajin Bessel-funktio ja $x_{h,l}$ on jokaisen hiukkasen $n = 1, \dots, N$ sekä XR-2-laitteen l välinen suuntimakulma.

Vastaavasti *angler*-sovelluksessa XR-2-laitteen ja AT-2-tagin välinen korkeuskulma simuloidaan katkaistusta normaalijakaumasta, jolle $a = 0$ ja $b = \pi/2$. Nyt muuttujat `elevation_location` ja `elevation_scale` vastaavat tämän jakauman sijainti- ja skaalaparametreja μ ja σ^2 ja näistä edellistä voidaan pitää itse korkeuskulman estimaattina. Määritellään jokaiselle tulokulmalle h sekä XR-2-laitteelle l seuraava korkeuskulman uskottavuusmalli:

$$L_{\gamma_{h,l}}(y_{h,l}|x_{h,l}; \mu_{h,l}, \sigma_{h,l}^2, a = 0, b = \pi/2) = \frac{1}{\sqrt{2\pi}} \exp \left(-\frac{(x_{h,l} - \mu_{h,l})}{2\sigma^2} \right) / \sqrt{\sigma^2} Z_{h,l}, \quad (4.15)$$

missä $Z_{h,l} = \frac{1}{2}(1 + \text{erf}(\frac{b - \mu_{h,l}}{\sqrt{\sigma^2}})) - \frac{1}{2}(1 + \text{erf}(\frac{a - \mu_{h,l}}{\sqrt{\sigma^2}}))$ ja $x_{h,l}$ on jokaisen hiukkasen $n = 1, \dots, N$ sekä XR-2-laitteen l välinen korkeuskulma.

Uskottavuusmallit (4.14) ja (4.15) kertomalla saadaan yhdistetty uskottavuusmalli hiukkassuotimen aika-askeleelle k sekä XR-2-laitteelle l

$$L_{k,l} = \prod_{h=1}^z L_{\theta_{h,l}} \times \prod_{h=1}^z L_{\gamma_{h,l}}, \quad (4.16)$$

josta voidaan edelleen laskea jokaisen hiukkasen uskottavuus aika-askeleella k kertomalla XR-2-laitteita vastaavat uskottavuudet keskenään

$$L_k = \prod_l^{n_s} L_{k,l}. \quad (4.17)$$

Koska yllä esitettyjen mallien uskottavuudet ovat käytännössä erittäin pieniä, käytetään numeerisista syistä itse algoritmissa logaritmoituja uskottavuusmalleja, jolloin $l_{k,l} = \sum \log(L_{\theta_{k,l}}) + \sum \log(L_{\gamma_{k,l}})$ ja $l_k = \sum_l^{n_s} l_{k,l}$.

4.4.3 Datan valinta

Koska Walkbasen paikannusjärjestelmä toimii monimutkaisessa, heijastuksia sisältävässä radioympäristössä, suoritetaan jokaisella aika-askeleella k vielä ylimääräinen datan valinta. Valinnan tarkoituksesta on jäättää mahdollisia heijastuksia sisältävät tai muuten selkeästi virheelliset tulokulmat paikannusdatan ulkopuolelle.

Yksinkertaisin tapa valita käytettävät kulmat on asettaa alaraja signaalin vahvuudelle, jolloin ainoastaan tiettyä kynnys-dBm-arvoa suuremman signaalin vahvuuden omaavat tulokulmat valitaan paikannukseen. Datan valinta suoritetaan kunkin aika-askeleen alussa ja tyypillisiä kynnysarvoja ovat esimerkiksi -100 dBm, -90 dBm ja -80 dBm. Jos kynnysarvo ei ole käytössä, käytetään ohjelmakoodissa arvoa -120 dBm, joka jäättää kaikki tulokulmat dataan. Datan valintaan palataan parametrien valinnan yhteydessä alaluvussa 4.5.2.

4.4.4 Dynaaminen malli

Dynaamisen mallin tehtäväնä on liikuttaa hiukkasia aika-askeleiden k ja $k+1$ välillä. Dynaaminen malli perustuu vektorisuunnistusmalliin (4.9). Vastaavaa mallia hyödytää muun muassa Solin (2016) [29]. Koska AT-2-tagi ei kuitenkaan lähetä nopeus- tai suuntadataa, asetetaan vektorisuunnistusmallin nopeutta ilmaiseva skalaarimuuttuja $D_k = 0$, jolloin malli yksinkertaistuu satunnaiskävelymalliksi

$$x_{k+1} = x_k + v_k, \quad (4.18)$$

missä v_k on kohinaa. Kohina luodaan jokaiselle aika-askeleelle seuraavasti. Ensin luodaan etäisyysvektori d_k katkaistusta normaalijakaumasta, jonka sijaintiparametri on 0, katkaisukohdat $a = 0$ ja $b = 10$ ja jonka keskihajonta q on paikannusalgoritmin suunnitteluparametri:

$$d_k \sim \mathcal{N}_{\text{katkaistu}}(\mu = 0, \sigma = q, a = 0, b = 10). \quad (4.19)$$

Tämän jälkeen luodaan suuntavektori p tasajakaumasta

$$p_k \sim \mathcal{U}(0, 2\pi) \quad (4.20)$$

ja lopulta kohinavektori

$$v_k = (\cos(p_k) \times d_k, \sin(p_k) \times d_k), \quad (4.21)$$

missä vektorin ensimmäinen elementti vastaa sijaintien leveysasteiden kohinaa ja toinen elementti pituusasteiden kohinaa.

4.4.4.1 Paikallaanolon havaitseminen

Virransäästyösyistä AT-2-tagi lähettilä dataa ainoastaan liikkueessaan. Jos laite ei ole 10 sekunnin aikana havainnut IMU-yksikkönsä perusteella kiihtyvyyttä, lopettaa laite datan lähetämisen. Laite on tällöin valmiusmoodissa. Tätä tietoa voidaan hyödyntää vaimentamalla dynaamisen mallin kohinaa, kun laitteen tiedetään olevan paikallaan. Jos paikannusalgoritmiin lisätään tarvittava 10 sekunnin viive, voidaan tätä viivettä hyödyntää paikallaanolon tehokkaampaan havaitsemiseen.

Jos yksikään XR-2-laite ei ole havainnut tagia aika-askeleen k aikana, voimme olettaa laitteen olevan valmiusmoodissa ja siten myös paikoillaan. Huomattavaa on kuitenkin, ettei päinvastainen päde. Paikoillaan oleva laite ei välittämättä ole valmisumoodissa, sillä valmiusmoodiin siirtyminen kestää 10 sekuntia. Tähän tie-toon perustuen voidaan havaita paikallaanolo jo ennen kuin laite lopettaa datan lähetämisen ja vaimentaa dynaamista mallia algoritmin 10 avulla. Algoritmi ajetaan jokaisella aika-askeleella k ennen hiukkasten siirtoa dynaamisen mallin avulla.

Algoritmi 10: Paikallaanolon havaitsemisalgoritmi

Result: Positiivinen kokonaislukumuuttuja m , joka osoittaa kuinka monaksi aika-askeleeksi dynaamista mallia tulee vimentaa. Jos $m = 0$ mallia ei vaimenneta.;

Data: Tagin tulokulmadata $L + 1$ aika-askeleelle (nykyinen aika-askel + L aika-askelta tulevaisuuteen). L tulee asettaa niin, että paikallaan oleva tagi ehtii valmiustilaan. Jos saatavilla, edellinen muuttujan m arvo. Algoritmin ensimmäisellä ajokerralla asetetaan $m = 0$;

```
begin
  begin
    | Jos  $m > 0$ , asetetaan  $m = f(m) = m - 1$  ja pysäytetään algoritmi. Jos
      |  $m = 0$  jatketaan algoritmin suorittamista.;

  for  $l = \{k + 1, \dots, k + L\}$  do
    begin
      | Merkitään  $o_l$  kulmahavaintojen määärää aika-askeleella  $l$ ;
      if  $o_l = 0$  ja  $m = 0$  then
        begin
          | Asetetaan  $m = l - k$ ;

      else
        if  $o_l = 0$ ,  $m > 0$  ja  $m + 1 = l - k$  then
          begin
            | Asetetaan  $m = l - k$ ;
```

Yllä esitetty algoritmi etsii ensin ensimmäisen aika-askeleen, jonka aikana ei ole tallennettu lainkaan kulmahavaintoja ja asettaa muuttujan m arvon vastaamaan tästä aika-askelta. Jos algoritmi löytää useita peräkkäisiä aika-askeleita, joiden aikana ei ole tallennettu lainkaan kulmahavaintoja, valitsee se näistä suurimman. Koska koeasetelmassa AT-2-laitteen valmiustila aktivoituu 10 sekunnin kohdalla, valitaan $L = 10$.

Jos paikallaanolon havaitsemisalgoritmin nojalla dynaamista mallia päädytään vimentamaan (so. $m > 0$), asetetaan dynaamisen mallin keskihajonta-arvo $q_k = \frac{q}{10}$. Dynaamista mallia ei siis täysin poisteta käytöstä, jotta paikannusalgoritmi pystyy tehokkaammin hyödyntämään myös vaimennettujen aika-askeleiden dataa. Näin sijaintiestimaatti konvergoi mahdollisimman lähelle todellista sijaintia, johon tagi on pysähtynyt.

4.4.4.2 Karttasovitusalgoritmi

Dynaamista mallia sovellettaessa voimme myös hopyntää saatavilla olevaa sisätilan karttadataa. Määritellään kaksi polygonityyppiä, lattiapolygoni F sekä estepolygonit $E = \{E_1, \dots, E_n\}$. Lattiapolygoni määrittää alueen, jonka sisällä hiukkassuodinalgoritmien hiukkasten pitää pysyä. Vastaavasti estepolygonien joukko määrittää lattiapolygonin sisällä alueet, joiden sisälle hiukkassuodinalgoritmin hiukkaset eivät voi siirtyä ja joiden läpi tagi ei voi kulkea. Käytännössä estepolygonit kuvaavat tilan seiniä sekä tiedossa olevia esteitä, kuten hyllyjä ja pöytää.

Tarkastellaan ensin jokaisen $i = 1, \dots, N$ hiukkasen kohdalla, siirtääkö dynaaminen malli hiukkasen polygonin F ulkopuolelle tai jonkin polygoneista E sisäpuolelle ja asetetaan näitä hiukkasia vastaavat painot nollaan.

$$w_{k+1}^i = \begin{cases} w_{k+1}^i & \text{jos } x_{k+1}^i \in F, \\ 0 & \text{jos } x_{k+1}^i \notin F \vee x_{k+1}^i \in E \end{cases} \quad (4.22)$$

Tämän jälkeen tarkastellaan jokaisen jäljellä olevan ($w_{k+1}^i \neq 0$) mallin siirtämän hiukkasen polkua $x_{k+1}^i - x_k^i$. Jos tämä polku ylittää yhden tai useamman estepolygonin E asetetaan kyseiselle hiukkaselle rangaistus seuraavasti:

$$w_{k+1}^i = \begin{cases} \frac{1}{P} \times w_{k+1}^i & \text{jos } x_{k+1}^i \text{ ylittää estepolygonin,} \\ w_{k+1}^i & \text{muulloin} \end{cases} \quad (4.23)$$

Tässä rangaistus P on algoritmin suunnitteluparametri. Vastaavaa toteuttaa ovat hyödyntäneet muun muassa Davidson &al (2010) [10], jotka ehdottavat rangaistusarvoa $P = \frac{1}{1000}$. Rangaistuksen valintaan palataan parametrien valinnan yhteydessä 4.5.2.

Koska erityisesti (4.22) asettaa hiukkasten painoja nollaan, eivät karttasoviteet hiukkaset välttämättä enää estimoijia suodinjakamaa tehokkaasti. Bojja &al. [2] ehdottavat suoritettavaksi uudelleenotantaa karttasovituksen jälkeen. Haluttaessa uudellenotanta suoritetaan seuraavasti. Merkitään nollapainoisten hiukkasten lukumäärää N_0 . Nyt otetaan uudet N_0 otosta palauttaen joukosta $\{x_{1:k}^i\}_{i=1}^N$, missä otoksen i todennäköisyys on $w_{k|k}^i$. Korvataan nollapainoiset hiukkaset näin saadulla otoksella. Uudelleenotantaa ei käytetä empiirisessä esimerkissä, sillä se lisää tarvittavaa laskentatehoa ja hankaloittaa varianssin estimointia.

4.4.5 Siloittelumalli

Koska haluttu sisätilapaikannusalgoritmi on online-algoritmi ja käytössä olevat laskennalliset resurssit ovat rajallisia, käytetään siloitteluun algoritmissa 7 esitettyä prediktiviivistä siloitinta. Alaluvussa 4.5 esitettävässä empiirisessä esimerkissä on tästä huolimatta kaikki data algoritmin saatavilla, joten viivettä ei tarvitse lisätä itse algoritmiin. Siloittelu saavutetaan yksinkertaisesti päivittämällä aika-askeleella $k < T$ painot seuraavan lauseen mukaan

$$w_k = w_k^i \bar{w}_{k+1}^i, \quad \text{missä } \bar{w}_{k+1}^i = \frac{\sum_{j=1}^N w_k^j p(x_{k+1}^i | x_k^j)}{q(x_{k+1}^i | x_k^i, y_{k+1})}. \quad (4.24)$$

Empiirisessä esimerkissä alla hyödynnetään koetarkoituksessa vain tästä yksinkertaista siloittelumallia.

4.4.6 WB-sisätilapaikannusalgoritmi

Alla esitetään luvun 3 SIR-algoritmiin (3) sekä aiemmin tässä luvussa esitettyihin algoritmeihin perustuva WB-sisätilapaikannusalgoritmi kokonaisuudessaan. Luettavuuden parantamiseksi algoritmi on jaettu kahteen osaan niin, että algoritmin runko kuvataan algoritmissa 11 ja sijaintiestimaatin luominen algoritmissa 12.

Algoritmin priorijakaumana p_{x_0} käytetään kahta toisistaan riippumatonta otosta tasajakaumista, joista toinen vastaa leveys- ja toinen pituusasteita. Jakaumien alkupisteet valitaan niin, että ne vastaavat pienimpiä paikantimien leveys- ja pituusasteista. Vastaavasti pääteläpisteet valitaan niin, että ne vastaavat suurimpia paikantimien leveys- ja pituusasteita.

$$p_{x_{0_{\text{lat}}}} \sim \mathcal{U}(\min \lambda, \max \lambda), \quad (4.25)$$

$$p_{x_{0_{\text{lon}}}} \sim \mathcal{U}(\min \phi, \max \phi). \quad (4.26)$$

Koska järjestelmän on tarkoitus toimia ainoastaan paikantimien muodostaman suo-rakaiteen sisäpuolella, ovat valitut jakaumien pääteläpisteet riittävät. Kummastakin jakaumasta tehdään \sqrt{N} otosta, jolloin N partikkelia x_0^i saadaan näiden otosten permutatioina. Alaluvussa 2.1.1.2 määritellyn signaali-kohinasuhteen (2.9) voi todeta olevan hyvin matala, sillä koe-esimerkin dynaaminen malli (4.18) noudattaa satunnaiskulkua ja on siten pelkkää kohinaa. Tämän perusteella WB-sisätilapaikannusalgoritmi käyttää ehdotusjakaumanaan tilavektorin ehdollista prioria $p(x_k|x_{k-1})$. Hiukkasten määrä N sekä muiden suunnitteluparametrien valinnasta kts. alaluku 4.5.2 alla.

Koska esitetty priorijakauma p_{x_0} on epäinformatiivinen ja monimutkainen koeympäristö aiheuttaa monin paikoin heijastuksia, on mahdollista, että ensimmäisten aika-askeleiden aikana algoritmi tuottaa huonoja sijaintiestimaatteja. Tästä syystä algoritmia ajettaessa suoritetaan ensimmäisellä $b = 3$ aika-askeleella sisäänajo, jonka aikana karttasovitusta ei käytetä ja jonka aikana kohina-arvona käytetään arvoa $q_k = q \times 10$. Myöskään sijaintiestimaatteja ei sisäänajoaika-askeleilla luoda. Siitä lähtien sijaintiestimaatit $\hat{x}_{1:b}$ korvataan ensimmäisellä sisäänajon jälkeisellä sijaintiestimaatilla \hat{x}_{b+1} . Näin estetään mahdollisia huonoja alkuarvoja vaikuttamasta pitkälle tuleviin sijaintiestimaatteihin.

Algoritmin tuottamien estimaattien varianssi estimoidaan käyttäen ALVar-varianssiestimaattia. Käytännössä tämä tarkoittaa varianssin estimointia käyttäen OD-varianssiestimaattia (2.26) niin, että viiveparametri λ valitaan mukautuvasti käyttäen yhtälöä (2.27). Koska empiirisessä esimerkissä estimoitavia ulottuvuuksia on kaksi, leveys- ja pituuskoordinaatit, lasketaan varianssi kummallekin näistä erikseen. Saadaan pituuskoordinaattien varianssiestimaatti $\hat{\sigma}_{k_x}^2$ ja leveyskoordinaattien varianssiestimaatti $\hat{\sigma}_{k_y}^2$.

Lasketaan lisäksi kovarianssiestimaatti

$$\hat{\text{Cov}}_k = \frac{1}{N-1} \sum_{i=1}^N w_k^i (x_{k_x}^i [E_{k(\lambda)}] - \hat{\mu}_{k_x})(x_{k_y}^i [E_{k(\lambda)}] - \hat{\mu}_{k_y}), \quad (4.27)$$

Taulukko 4.5: WB-sisätilapaikannusalgoritmin suunnitteluparametrit

Parametri	Selitys
q	Dynaamisen mallin kohina-arvo
map_matching	Käytetäänkö karttasovitusalgoritmia, T/F
N	Hiukkasten määrä, kokonaisluku
P	Jos karttasovitusalgoritmi on käytössä, rangaistusparametri P , liukuluku
resampling	Adaptiivisen uudelleenotannat kynnysarvo, liukuluku välillä $[0, 1]$. Jos 0, uudelleenotantaa ei käytetä
rssi_threshold	Datan valinnassa käytettävä signaalin vahvuuden kynnysarvo, kokonaisluku
smoothing	Käytetäänkö prediktivistä siloitinta, T/F
b	Sisäänajoaika-askeleiden lukumäärä, pidetään vakioarvoisena $b = 3$

missä $\hat{\mu}_{k_x}$ on painotettu summa (2.25) laskettuna pituuskoordinaateille ja $\hat{\mu}_{k_y}$ painotettu summa (2.25) laskettuna leveyskoordinaateille. Viiveparametri λ on edelleen valittu mukautuvasti käyttäen yhtälöä (2.27). Yhdistetään estimoidut varianssit ja kovarianssi kovarianssimatriisiin estimaatiaksi

$$\hat{\Sigma}_k = \begin{bmatrix} \hat{\sigma}_{k_x}^2 & \hat{\text{Cov}}_k \\ \hat{\text{Cov}}_k & \hat{\sigma}_{k_y}^2 \end{bmatrix} \quad (4.28)$$

ja lasketaan aika-askeleen k varianssiestimaatti kovarianssimatriisiin Frobeniuksen normina

$$\hat{\sigma}_k = \|\hat{\Sigma}_k\| = \sqrt{\text{tr}(\hat{\Sigma}_k * \hat{\Sigma}_k)}, \quad (4.29)$$

jolloin saadaan jokaiselle aika-askeleelle yksi skalaariarvoisen varianssiestimaatti. Tässä käytetään Frobeniuksen normia, koska se tarjoaa yksinkertaisen ja intuitiivisen tavan yhdistää (ko)varianssiavtot yhdeksi skalaariarvoiseksi estimaatiaksi.

Ohessa esitetyn WB-sisätilapaikannusalgoritmin suoritusnopeus on perusmuodossaan luokkaa $\mathcal{O}(N)$. Taulukossa 4.5 on esitetty algoritmin suunnitteluparametrit, joiden valintaan palataan empiirisessä esimerkissä alla.

Koeasetelmassa käytetty algoritmin 11 toteutus on ohjelmoitu R-kielellä. Algoritmin toteutus on pääosin vektorisoitu ja tehokas. For-silmukkaa on käytetty ainoastaan aika-askeleiden läpikäyntiin. Koska tämän silmukan muuntaminen vektorisoituuun muotoon ei ole mahdollista, voidaan toteutusta pitää näiltä osin hyvin optimoituna. Kaikki algoritmin datan käsittely on toteutettu suorituskyvyltään erinomaisella `data.table`-kirjastolla.

Algoritmi 11: WB-sisätilapaikannusalgoritmi

Result: Tagin sijaintiestimaatti \hat{x} kullekin aika-askeleelle $k = 1, \dots, T$, näitä vastaavat varianssiestimaatit $\hat{\sigma}_k^2$;

Data: Taulukossa 4.4 esitetty data y_k kullekin aika-askeleelle $k = 1, \dots, T$ alaluvussa 4.4.4.2 esitetty polygonimetadata. Taulukossa 4.5 esitettyt parametrit.;

```
begin
  begin
    Luodaan priorijakauma  $p_{x_0}$  otantana jakaumista (4.25), asetetaan painot
     $w_0 = 1/N$ ;
  for  $k = \{1, \dots, T\}$  do
    if  $k > b$  then
      begin
        Ajetaan paikallaanolon havaitsemisalgoritmi 10. Jos havaitaan
        paikallaanolo, päivitetään kohina-arvo  $q_k = \frac{q}{10}$ , muussa
        tapauksessa  $q_k = q$ ;
      begin
        Sovitetaan partikkeleihin dynaaminen malli (4.18)  $x_{k+1} = x_k + v_k$ ,
        missä  $v_k$  on luotu alaluvussa 4.4.4 esitettyllä menetelmällä sekä
        kohina-arvolla  $q_k$ ;
      else
        begin
          Sovitetaan partikkeleihin dynaaminen malli (4.18)  $x_{k+1} = x_k + v_k$ ,
          missä  $v_k$  on luotu alaluvussa 4.4.4 esitettyllä menetelmällä sekä
          kohina-arvolla  $q_k = q \times 10$ ;
        begin
          Jos karttasovitusalgoritmi on käytössä ja  $k > b$ , päivitetään painot  $w_k$ 
          alaluvun 4.4.4.2 karttasovitusalgoritmilla ja rangaistusarvolla  $P$ ;
        begin
          Päivitetään painot ja luodaan sijaintiestimaatti algoritmilla 12;
        begin
          Lasketaan ALVar-varianssi  $\hat{\sigma}_k^2$ .
        begin
          Lasketaan efektiivinen otoskoko  $\hat{N}_{eff}$ .
        if  $\hat{N}_{eff} < N_{th}$  then
          begin
            Otetaan uudet  $N$  otosta palauttaen joukosta  $\{x_{1:k}^i\}_{i=1}^N$ , missä
            otoksen  $i$  todennäköisyys on  $w_{k|k}^i$ .
          begin
            Asetetaan painot  $w_{k|k}^i = 1/N$ .
          begin
            Päivitetään Henok-indeksit  $E$  uudelleenotannan perusteella.
```

Algoritmi 12: WB-sisätilapaikannusalgoritmin sijaintiestimaatin luominen

Result: Tagin sijaintiestimaatti \hat{x}_k , päivitettyt painot w_k ;

Data: Hiukkaset x_k , painot w_k ;

for $i = \{1, 2, \dots, N\}$ **do**

begin

 Päivitetään painot $w_{k|k}$ alaluvun 4.4.2 uskottavuusmalleilla.

begin

 Estimoidaan p laskemalla tiheydelle approksimaatio

$$\hat{p}(x_{1:k}|y_{1:k}) = \sum_{i=1}^N w_{k|k}^i \delta(x_{1:k} - x_{1:k}^i).$$

if $k < T \ \& \ k > b$ **then**

begin

 Päivitetään paino w_k^i prediktiviisellä siloittimella $w_k^i = w_k^i \bar{w}_{k+1}^i$.

if $k > b$ **then**

begin

 Luodaan sijaintiestimaatti $\hat{x}_k = \sum_{i=1}^N w_i^k x_k^i$. Estimaatti lasketaan erikseen

 pituus- ja leveyskoordinaateille.

if $k = b + 1$ **then**

begin

 Asetetaan sijaintiestimaatit $\hat{x}_{1:b} = \hat{x}_k$.

Koska algoritmin tuottamat uskottavuudet ja painot ovat pieniä, on laskentatarkkuusongelmien välttämiseksi toteutuksessa käytetty logaritmoituja painoja ja uskottavuusfunktiota. Tämä ei vaikuta itse algoritmin toimintaan, mutta estää numeristen ongelmien syntymisen.

Paikannusvirheen laskemisessa on etäisyysfunktiona käytetty raster-kirjaston pointDistance()-funktiota. Algoritmissa 9 kuvattu testipolin tihennys on tehty ennen algoritmin ajoa ja tihennettyä testipolkua käytetään algoritmin syöteenä. Ti-hennys on tehty smoothr-kirjaston smooth_densify()-funktiolla. Ohjelmaissa on korostettu tiiviyden sijaan luettavuutta ja koodi on kommentoitu kattavasti.

Algoritmien R-koodi löytyy osoitteesta https://github.com/rintakumpu/progradu/analyysi/R/pf_positioning.R.

4.5 Empiirinen esimerkki

4.5.1 Koeasetelma

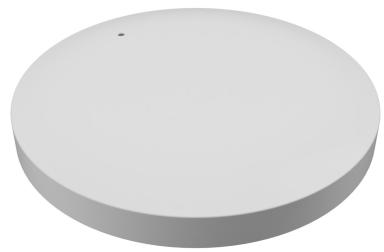
Esimerkissä käytettiin edellä kuvattua WB-hiukkassuodinalgoritmia AT-2-lähettimen sijainnin estimointiin. Paikannukseen käytetty data kerättiin ruokakauppaympäristössä ostoskärryn kiinnitetyn lähettimen sekä kattoripustuksiin asennettujen vastaanottimien avulla. Paikannusesimerkissä lähettimenä toimi 6hz taajuudella havaintoja lähetvä AT-2 paikannustagi (kuva 4.3), vastaanottimena testiympäristöön asennetut $n_s = 33$ Walkbase XR-2 -vastaanotinta (kuvat 4.4 ja 4.5). Jokainen vastaanotin sisälsi kuusitoista antennia, joiden vastaanottamien lähetinsignaalien

perusteella vastaanottimet laskivat signaalin tulokulman suhteessa vastaanottimeen.

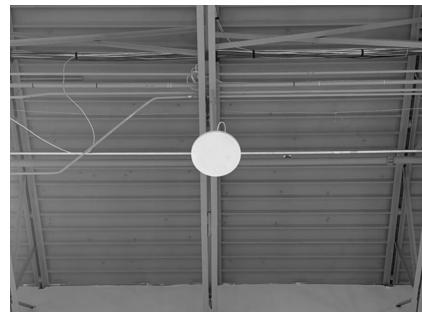
Havaintoja kerättiin yhteensä $n_{obs} = 12379$ kappaletta ja ne kattoivat $T = 432$ sekuntia. Havaintojen aikaleimat tallennettiin sekunnin tuhannesosan tarkkuudella. Jokainen havainto koostui taulukossa 4.4 kuvatuista muuttujista. Esimerkissä analysoitiin ja vertailtiin algoritmin eri suunnitteluparametrikombinaatioiden suorituskykyä sekä suorituskyvyn että paikanmuistikulmien näkökulmasta. Vertailuarvona käytettiin perinteistä triangulaatio-algoritmia.



Kuva 4.3: Walkbase AT-2



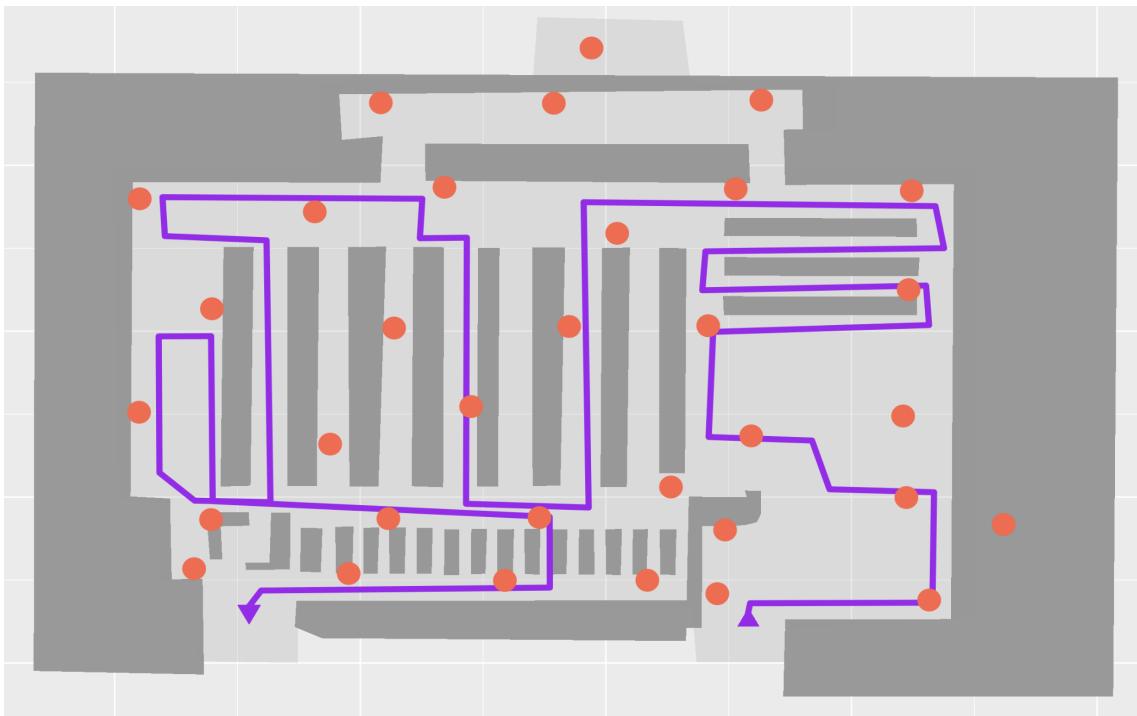
Kuva 4.4: Walkbase XR-2



Kuva 4.5: Walkbase XR-2 asennettuna

Koeympäristön pohjapiirustus on esitetty kuvassa 4.6. Piirustuksessa XR-2-paikantimet on kuvattu punaisilla ympyröillä ja kuljettu testipolku on merkitty violetilla janalla. Piirustus luotiin käyttäen `ggplot2`-kirjastoa ja se löytyy RDS-muotoon tallennettuna osoitteesta

<https://github.com/rintakumpu/progradu/analyysi/R/data/sitemap.RDS>.



Kuva 4.6: Koeasetelman pohjapiirustus

Dataa kerättäessä AT-2-tagit kiinnitettiin ostoskärryyn 1.2 metrin korkeudelle (kts. kuvat 4.7 ja 4.8) ja testipolku käveltiin mahdollisimman tasaisella nopeudella. Data kerättiin aikaan, jolloin testiympäristön käyttöaste oli alhainen. Tällä minimoitiin radiosignaalien tielle osuvien ihmisten vaikutus signaaleihin. Testipolku suunniteltiin niin, että se kattoi vaihtelevia radioympäristöjä. Tällä minimoitiin yksittäisen sijaintiestimatiin radioympäristön vaikutus paikannusvirheeseen.

Kerätty tulokulmadata löytyy osoitteesta
<https://github.com/rintakumpu/progradu/analyysi/data/y.csv>. Pohjapiirustus- ja polkudata löytyvät osoitteista
https://github.com/rintakumpu/progradu/analyysi/data/exclusion_polygons.csv (ekskluusiopolygonit),
https://github.com/rintakumpu/progradu/analyysi/data/inclusion_polygons.csv (lattiapolygonit) ja
https://github.com/rintakumpu/progradu/analyysi/data/test_path.csv (testipolku). Datassa koordinaatit on valmiiksi interpoloitu metreiksi, jotta testiympäristön sijaintia ei voi paikallistaa koordinaattien perusteella. Interpolointiin käytetty ohjel makoodi löytyy osoitteesta
https://github.com/rintakumpu/progradu/analyysi/R/interpolate_coordinates.R . Osoitteesta
<https://github.com/rintakumpu/progradu/analyysi/analyysi.Rmd> löytyy itse analyysikoodin sisältävä R Markdown -muistikirja.



Kuva 4.7: Walkbase AT-2 kiinnitettyynä



Kuva 4.8: Walkbase AT-2 kiinnitettyynä

4.5.2 Parametrien valinta

Kokeessa oli tarkoituksena testata kunkin taulukossa 4.5 esitetyn parametrin vaikuttusta algoritmin paikannusvirheeseen, ajoaikaan sekä varianssiin. Koska kaikkien parametrikombinaatioiden testaaminen ei ollut mahdollista (eikä mielekäästää), suoritettiin kerättyyn dataan perustuva paikannus kolmessa vaiheessa. Kussakin vaiheessa paikannusalgoritmi ajettiin jokaisella vaiheeseen liittyvällä suunnitteluparametrikombinaatiolla $r = 30$ kertaa ja tulokset laskettiin näiden 30 ajon aritmeettisena keskiarvona.

Ensimmäisessä vaiheessa tarkasteltiin partikkeli määrä $N = 100, 1000, 10000$ sekä uudelleenotannan kynnysarvon $resampling = 0, 1/10, 2/3, 1$ vaikutusta paikannuskeskivirheeseen. Karttasovitusalgoritmia ei käytetty, kuten ei myöskään prediktivistä siloitinta eikä signaalin vahvuuden kynnysarvoa. Dynaamisen mallin kohina-arvo $q = 2$ pidettiin vakiona. Ensimmäisessä vaiheessa tarkasteltiin siis 12 eri suunnitteluparametrikombinaatiota.

Toisessa vaiheessa valittiin ensimmäisen vaiheen tulosten perusteella parhaimpana paikannusvirheen suhteessa suoritusaikeaan ja varianssiin tuottava uudelleenotannan kynnysarvo $resampling$ sekä partikkeli määriä N . Nämä pidettiin vakioarvoisina ja testattiin karttasovitusalgoritmia $map_matching = T, F$, karttasovitusalgoritmin rangaistusarvoa $P = 1, 100, 1000$ sekä dynaamisen mallin kohina-arvoa $q = 0.75, 1.5, 2, 2.5$. Signaalin vahvuuden kynnysarvoa ei käytetty, kuten ei käytetty myöskään prediktivistä siloitinta. Koska rangaistusarvo P oli käytössä ainoastaan karttasovitusalgoritmia käytettäessä, tarkasteltiin toisessa vaiheessa 16 eri suunnitteluparametrikombinaatiota.

Viimeisessä vaiheessa valittiin edellisten vaihdeiden tulosten perusteella parhaimpana paikannusvirheen suhteessa suoritusaikeaan ja varianssiin tuottavat parametrit testattujen joukosta ja testattiin datan valinnassa käytettävästä signaalin vahvuuden kynnysarvoa $rssi_threshold = -100, -90, -80$ sekä prediktivistä siloitinta $smoothing = T, F$ eli kuutta eri suunnitteluparametrikombinaatiota.

Tulosten vertailukohtana käytettiin Pierlot & al. artikkelissa “A New Three Object Triangulation Algorithm Based on the Power Center of Three Circles” (2011) esittämää ToTal-triangulaatioalgoritmia [28]. Triangulaatio-algoritmia ei käsitellä tässä tarkemmin, mutta se on esitetty algoritmissa 13. Algoritmia varten valittiin kullakin aika-askeleella k ne kolme XR-2-laitetta ja kulmahavaintoa, joiden RSSI-arvo

oli korkein. Käytetty algoritmin toteutus löytyy osoitteesta <https://github.com/rintakumpu/progradu/analyysi/R/total.R>.

Paikannusalgoritmi ajettiin RStudion versiossa 2023.09.0+463 R-ohjelmointikielen versiolla 4.2.0. Tietokoneena käytettiin vuoden 2021 mallia olevaa MacBook Pro -kannettavaa, jossa oli Apple M1 Pro -prosessori sekä 32 gigatavua LPDDR5-muistia. Suoritusnopeuden mittaanmiseen käytettiin `Sys.time()`-funktiota.

Algoritmi 13: ToTal (Three object Triangulation algorithm)

Result: AT-2-tagin sijaintiestimaatti (x_R, y_R) .

Data: Kolmen XR-2-laitteen koordinaatit (x_i, y_i) , $i = \{1, 2, 3\}$ ja näitä vastaavat vastakkaiset kulmahavainnot $\Phi'_1, \Phi'_2, \Phi'_3$.

begin

 Lasketaan muokatut koordinaatit

$$x'_1 = x_1 - x_2, \quad y'_1 = y_1 - y_2, \quad x'_3 = x_3 - x_2, \quad y'_3 = y_3 - y_2.$$

begin

 Lasketaan kotangentit

$$T_{12} = \cot(\Phi'_2 - \Phi'_1), \quad T_{23} = \cot(\Phi'_3 - \Phi'_2), \quad T_{31} = \frac{1 - T_{12}T_{23}}{T_{12} + T_{23}}.$$

begin

 Lasketaan muokatut ympyröiden keskipisteet (x'_{ij}, y'_{ij})

$$x'_{12} = x'_1 + T_{12}y'_1, \quad y'_{12} = y'_1 - T_{12}x'_1$$

$$x'_{23} = x'_3 - T_{23}y'_3, \quad y'_{23} = y'_3 + T_{23}x'_3$$

$$x'_{31} = (x'_3 + x'_1) + T_{31}(y'_3 - y'_1), \quad y'_{31} = (y'_3 + y'_1) - T_{31}(x'_3 - x'_1).$$

begin

 Lasketaan $k'_{31} = x'_1x'_3 + y'_1y'_3 + T_{31}(x'_1y'_3 - x'_3y'_1)$.

begin

 Lasketaan nimittääjä D (jos $D = 0$ palautetaan virhe).

$$D = (x'_{12} - x'_{23})(y'_{23} - y'_{31}) - (y'_{12} - y'_{23})(x'_{23} - x'_{31}).$$

begin

 Lasketaan ja palautetaan sijaintiestimaatti (x_R, y_R) .

$$x_R = x_2 + \frac{k'_{31}(y'_{12} - y'_{23})}{D} \quad y_R = y_2 + \frac{k'_{31}(x'_{23} - x'_{12})}{D}.$$

4.5.3 Tulokset

Ensimmäisessä vaiheessa suoritettiin paikannus partikkeliin määräällä $N = 100, 1000, 10000$ sekä uudelleenotannan kynnysarvolla $resampling = 0, 1/10, 2/3, 1$. Kun kynnysarvo oli 0, uudelleenotantaa ei käytetty, jolloin SIR-algoritmin sijaan paikannus suoritettiin SIS-algoritmilla. Kun kynnysarvo oli 1, uudelleenotanta suoritettiin jokaisella aika-askeleella. Arvoilla 1/10 ja 2/3 käytettiin adaptiivista uudelleenotantaa. Tulokset on esitetty kuvassa 4.9 sekä taulukoissa 4.6 ja 4.7. Tulostaulukon sarake $< 1m$ kuvailee suurinta persentiiliä/100, jolla saavutettiin haluttu metrin paikannusvirhe. Ajojen tulokset on esitetty karttapolkuina liitteen A alaluvussa Vaihe 1.

Taulukko 4.6: Vaiheen 1 tulokset, paikannusvirhe

N	resampling	Mediaani (m)	Mediaanin 95%-luottamusväli	<1m
100	0.00	1.95	[1.87, 2.03]	0.28
1000	0.00	1.05	[1.03, 1.07]	0.48
10000	0.00	0.88	[0.87, 0.88]	0.55
100	0.10	0.86	[0.86, 0.87]	0.57
1000	0.10	0.86	[0.86, 0.87]	0.57
10000	0.10	0.86	[0.86, 0.86]	0.58
100	0.67	0.88	[0.87, 0.89]	0.56
1000	0.67	0.86	[0.86, 0.87]	0.58
10000	0.67	0.86	[0.86, 0.86]	0.58
100	1.00	0.88	[0.87, 0.89]	0.56
1000	1.00	0.86	[0.86, 0.86]	0.58
10000	1.00	0.86	[0.86, 0.86]	0.58

Taulukko 4.7: Vaiheen 1 tulokset, varianssi ja ajoaika

N	resampling	Varianssi	MC-varianssi	Ajoaika (s)
100	0.00	NA	22.54	7.64
1000	0.00	NA	2.02	15.35
10000	0.00	NA	0.13	85.44
100	0.10	0.07	0.14	6.97
1000	0.10	0.00	0.03	14.40
10000	0.10	0.00	0.01	82.68
100	0.67	0.06	0.09	7.16
1000	0.67	0.00	0.02	14.50
10000	0.67	0.00	0.00	81.78
100	1.00	0.06	0.09	7.19
1000	1.00	0.00	0.02	14.48
10000	1.00	0.00	0.00	83.31

Ensimmäisen vaiheen tuloksista huomataan, ettei hiukkasten määrellä ole juuri-kan vaikutusta mediaanipaikannusvirheeseen, kun uudelleenotanta on käytössä (so. käytämme SIR-algoritmia). Samoin adaptiivisen uudelleenotannan kynnysarvolla ei ole juurikaan vaikutusta mediaanipaikannusvirheeseen.

Se, ettei partikkelien määrään kasvattaminen automaatisesti paranna paikan-nustarkkuutta viittaa siihen, että koeasetelma on herkkä priorijakauman valinnalle. Tuloksista huomataan lisäksi, että algoritmin aikakompleksisuus on uudelleenotan-nasta riipumatta luokkaa $\mathcal{O}(N)$, kuten tukielman teoriaosassa todettiin. Samoin hiukkasten määrään kasvattaminen pienentää variansseja, kuten teoriaosassa todettiin.

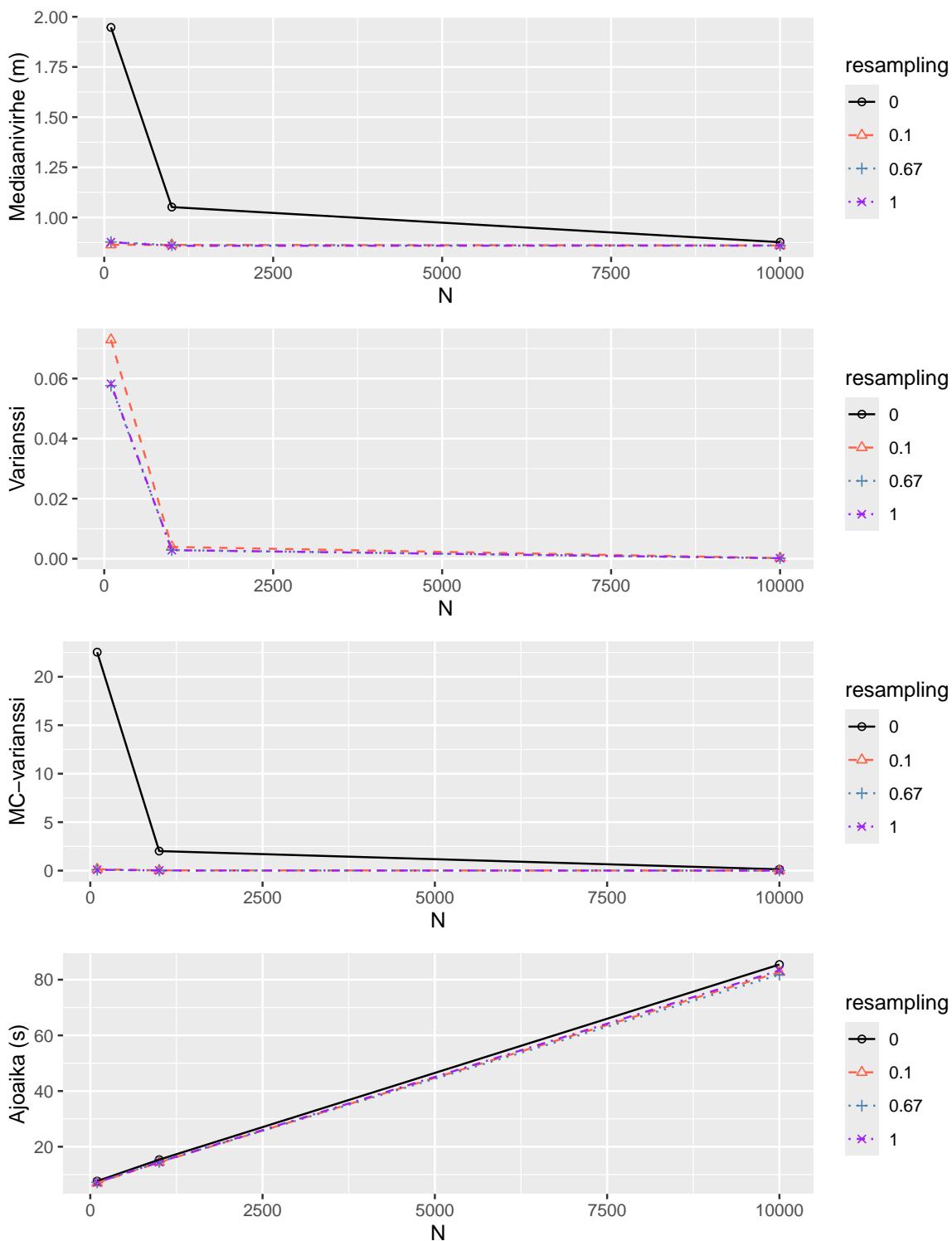
Koska MC-varianssi on laskettu itse sijaintiestimaateista ja ALVar-varianssi kaikista partikkeleista, on näiden kahden varianssin suuruusluokka eri. Huomataan kuitenkin, että nämä kakso varianssiestimaattia ovat hyvin korreloituneita, korrelatio-kertoimilla $\rho_{\text{Pearson}} \approx 0.98$ ja $\rho_{\text{Spearman}} \approx 0.97$. Voidaan siis olettaa ALVar-varianssin estimoivan hyvin hiukkassuotimen todellista varianssia. Kun uudelleenotanta ei ole käytössä, ALVar-varianssia ei lasketa.

Ensimmäisen vaiheen tulosten perusteella valitaan seuraavaan vaiheeseen ne N - ja *resampling*-parametriarvot, jotka tuottavat parhaimman mediaanipaikan-nusvirheen. Jos kahden eri paikannusvirheen 95%-luottamusvälit ovat päällekkäiset, valitaan arvoista ensin se, joka tuottaa paremman ALVar-varianssin. Jos myös ALVar-varianssit ovat samat, valitaan arvoista se, joka on ajojaltaan nopeampi. Näin päädytään parametriyhdistelmään $N = 1000$ ja *resampling* = 2/3.

Toisessa vaiheessa $N = 1000$ ja *resampling*=2/3 pidettiin vakioarvoisina ja testattiin karttasovitusalgoritmia *map_matching* = T, F , karttasovitusalgoritmin rangaistusarvoa $P = 1, 100, 1000$ sekä dynaamisen mallin kohina-arvoa $q = 0.75, 1.5, 2, 2.5$. Signaalin vahvuuden kynnysarvoa ei käytetty, kuten ei käytetty myöskään prediktivis-tä siloitinta. Koska rangaistusarvo P oli käytössä ainoastaan karttasovitusalgoritmia käytettäessä, tarkasteltiin toisessa vaiheessa 16 eri suunnitteluparametrikombinaatio-ta. Tulokset on esitetty kuvassa 4.10 sekä taulukoissa 4.8 ja 4.9. Ajojen tulokset on esitetty karttapolkuina liitteen A alaluvussa [Vaihe 2](#).

Toisen vaiheen tuloksista huomataan, että karttasovituksen käyttäminen parantaa paikannusvirhettä. Syy tähän on helppo havaita liitteenä olevista karttapoluilta. Paikannus ottaa nyt huomioon sisätilaympäristön, eikä enää luo estimaatteja sijain-teihin, joihin tagin on fyysisesti mahdotonta päästä. Samoin rangaistusarvo P :n lisääminen parantaa paikannusvirhettä. Tämä on odotettua, sillä isomman rangaistusarvon käyttäminen ei ainoastaan estää fyysisesti mahdottomia sijainteja vaan estää myös osan fyysisesti mahdottomista siirtymistä kahden peräkkäisen sijaintiestimaatin välillä.

Vastaavasti liikemallin kohina-arvon q pienentäminen parantaa paikannusvirhet-tä. Paikannusvirheen mielessä paras kohina-arvo on tulosten perusteella $q = 0.75$. Tämä vastaa hyvin kirjallisuuudessa esitettyjä keskimääräisiä kävelynopeuksia (kts. esim. [17]), kun huomioidaan, että kyse on kauppaympäristöstä. Vaikka parametri on tutkitun parametriavaruuden reunalla, ei tästä syystä alempia q -arvoja ole enää tarpeen tutkia, sillä alemmat arvot eivät enää vastaisi realistisia kävelynopeuksia. Liian pienellä q -arvolla algoritmi ei myöskään tutkisi signaaliympäristöä tarpeeksi tehokkaasti.



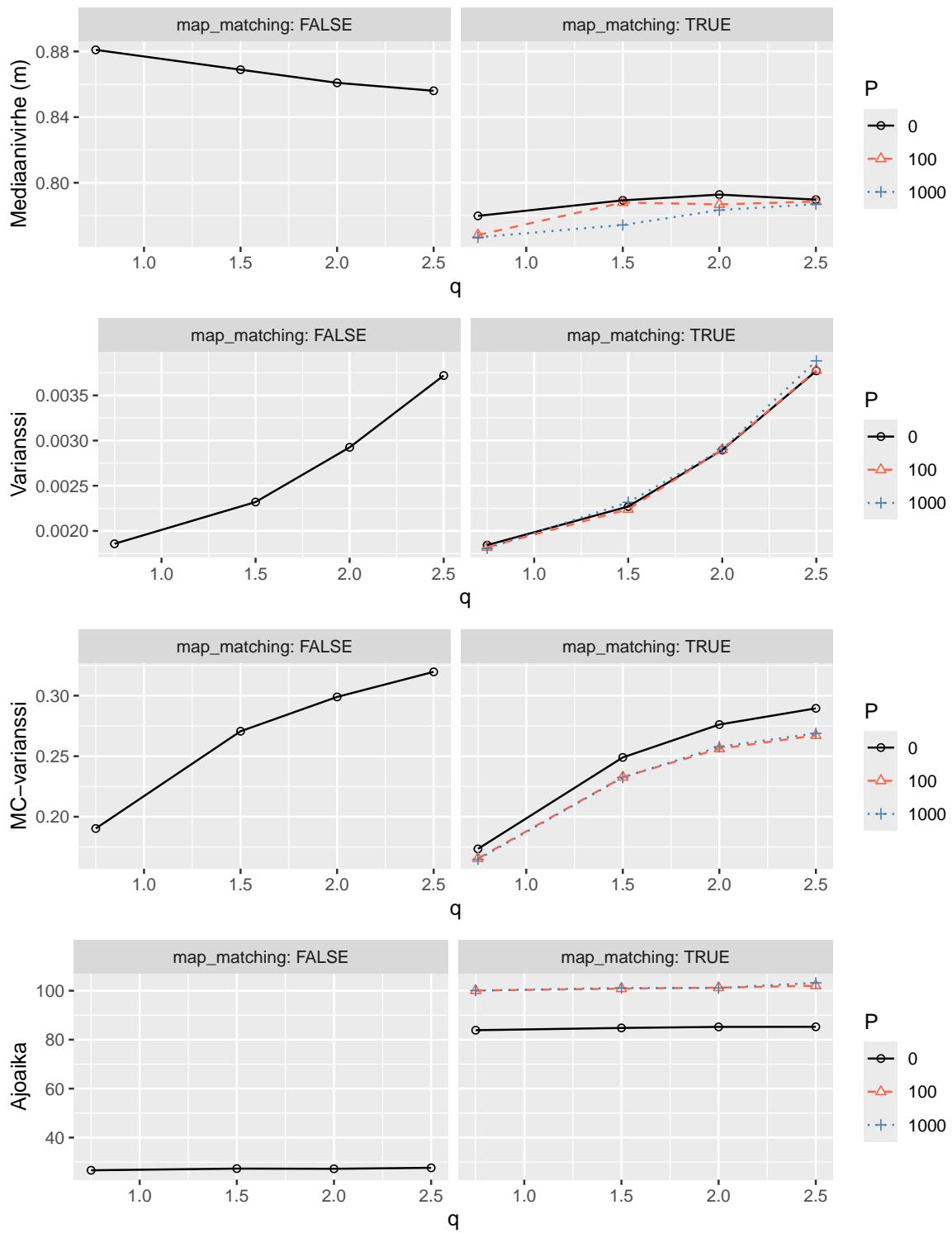
Kuva 4.9: Vaiheen 1 tulokset

Taulukko 4.8: Vaiheen 2 tulokset, paikannusvirhe

map_matching	P	q	Mediaani (m)	Mediaanin 95%-luottamusväli	<1m
TRUE	0	0.75	0.78	[0.77, 0.79]	0.63
TRUE	100	0.75	0.77	[0.76, 0.77]	0.63
TRUE	1000	0.75	0.77	[0.76, 0.77]	0.63
TRUE	0	1.50	0.79	[0.79, 0.79]	0.61
TRUE	100	1.50	0.79	[0.78, 0.79]	0.62
TRUE	1000	1.50	0.77	[0.77, 0.78]	0.62
TRUE	0	2.00	0.79	[0.79, 0.79]	0.61
TRUE	100	2.00	0.79	[0.78, 0.79]	0.62
TRUE	1000	2.00	0.78	[0.78, 0.79]	0.62
TRUE	0	2.50	0.79	[0.79, 0.79]	0.61
TRUE	100	2.50	0.79	[0.78, 0.79]	0.62
TRUE	1000	2.50	0.79	[0.78, 0.79]	0.61
FALSE	0	0.75	0.88	[0.88, 0.89]	0.57
FALSE	0	1.50	0.87	[0.87, 0.87]	0.57
FALSE	0	2.00	0.86	[0.86, 0.86]	0.58
FALSE	0	2.50	0.86	[0.85, 0.86]	0.57

Taulukko 4.9: Vaiheen 2 tulokset, varianssi ja ajoaika

map_matching	P	q	Varianssi	MC-varianssi	Ajoaika (s)
TRUE	0	0.75	0.002	0.173	83.866
TRUE	100	0.75	0.002	0.166	100.051
TRUE	1000	0.75	0.002	0.165	100.054
TRUE	0	1.50	0.002	0.249	84.796
TRUE	100	1.50	0.002	0.233	100.845
TRUE	1000	1.50	0.002	0.232	101.135
TRUE	0	2.00	0.003	0.276	85.220
TRUE	100	2.00	0.003	0.256	101.251
TRUE	1000	2.00	0.003	0.258	101.105
TRUE	0	2.50	0.004	0.290	85.257
TRUE	100	2.50	0.004	0.267	102.044
TRUE	1000	2.50	0.004	0.269	103.218
FALSE	0	0.75	0.002	0.190	26.638
FALSE	0	1.50	0.002	0.271	27.327
FALSE	0	2.00	0.003	0.299	27.254
FALSE	0	2.50	0.004	0.320	27.642



Kuva 4.10: Vaiheen 2 tulokset

Taulukko 4.10: Vaiheen 3 tulokset, paikannusvirhe

rssi_threshold	smoothing	Mediaani (m)	Mediaanin 95%-luottamusväli	<1m
-120	TRUE	0.78	[0.77, 0.79]	0.60
-100	TRUE	0.78	[0.77, 0.79]	0.60
-90	TRUE	0.78	[0.77, 0.78]	0.61
-80	TRUE	0.81	[0.8, 0.81]	0.59
-120	FALSE	0.77	[0.76, 0.77]	0.63
-100	FALSE	0.77	[0.76, 0.77]	0.63
-90	FALSE	0.76	[0.76, 0.77]	0.62
-80	FALSE	0.79	[0.78, 0.79]	0.59

Taulukko 4.11: Vaiheen 3 tulokset, varianssi ja ajoaika

rssi_threshold	smoothing	Varianssi	MC-varianssi	Ajoaika (s)
-120	TRUE	0	0.04	59.12
-100	TRUE	0	0.04	58.73
-90	TRUE	0	0.07	57.52
-80	TRUE	0	0.07	51.68
-120	FALSE	0	0.08	49.35
-100	FALSE	0	0.08	49.32
-90	FALSE	0	0.08	48.90
-80	FALSE	0	0.12	46.35

Huomataan myös, että parhaimman paikannusvirheen tuottava rangaistusarvo $P = 1000$ on tutkitun parametriavaruuden reunalla, joten mahdollisesti isommalla P -arvolla voitaisiin vielä parantaa paikannusvirhettä. Pienempien P -arvojen paikannusvirheiden luottamusvälit ovat kuitenkin päälekäisiä arvon $P = 1000$ kanssa, joten paikannusvirheeseen saatu lisähyöty ei todennäköisesti olisi tilastollisesti merkitseväää. Näiden tulosten perusteella valitaan viimeiseen vaiheeseen kiinteät arvot $\text{map_matching} = T$, $P = 1000$ ja $q = 0.75$.

Viimeisessä vaiheessa testattiin datan valinnassa käytettävää signaalin vahvuuden kynnysarvoa $\text{rssi_threshold} = -120, -100, -90, -80$ sekä prediktivistä siloitinta $\text{smoothing} = T, F$ eli kahdeksaa eri suunnitteluparametrikombinaatiota. Tulokset on esitetty kuvassa 4.11 sekä taulukoissa 4.10 ja 4.11. Ajojen tulokset on esitetty karttapolkuina liitteen A alaluvussa [Vaihe 3](#).

Taulukko 4.12: Tulosten perusteella valitut suunnitteluparametrit

Suunnitteluparametri	Arvo
N	1000
resampling	0.67
map_matching	TRUE
P	1000
q	0.75
smoothing	FALSE
rssi_threshold	-90

Tuloksista huomataan, ettei siloittelun käyttäminen paranna paikannusvirhettä. Tämä on odotettua, koska liikemallina on käytetty satunnaiskulkua. Liitteenä olevista karttapoluista kuitenkin huomataan, että siloittelu tuottaa odotetusti sileämpää polkuja, mikä saattaa olla käytännössä haluttu ominaisuus. Sen sijaan signaalin vahvuuden kynnysarvon nostaminen arvoon $rssi_threshold = -90$ parantaa hieman paikannustarkkuutta. Kynnysarvon käyttö myös nopeuttaa algoritmin ajoaikaa, sillä kynnysarvoa käytettäessä uskottavuusfunktio (4.17) lasketaan pienemmällä määrällä kulmadataa. Valitaan näin kolmannen vaiheen tulosten perusteella parametrit $smoothing=F$ ja $rssi_threshold=-90$.

Taulukossa 4.12 on esitetty yhteenvetona kaikkien koevaiheiden tulosten perusteella valitut suunnitteluparametrit. Täydet eri koevaiheiden tulokset löytyvät csv-muodossa osoitteista

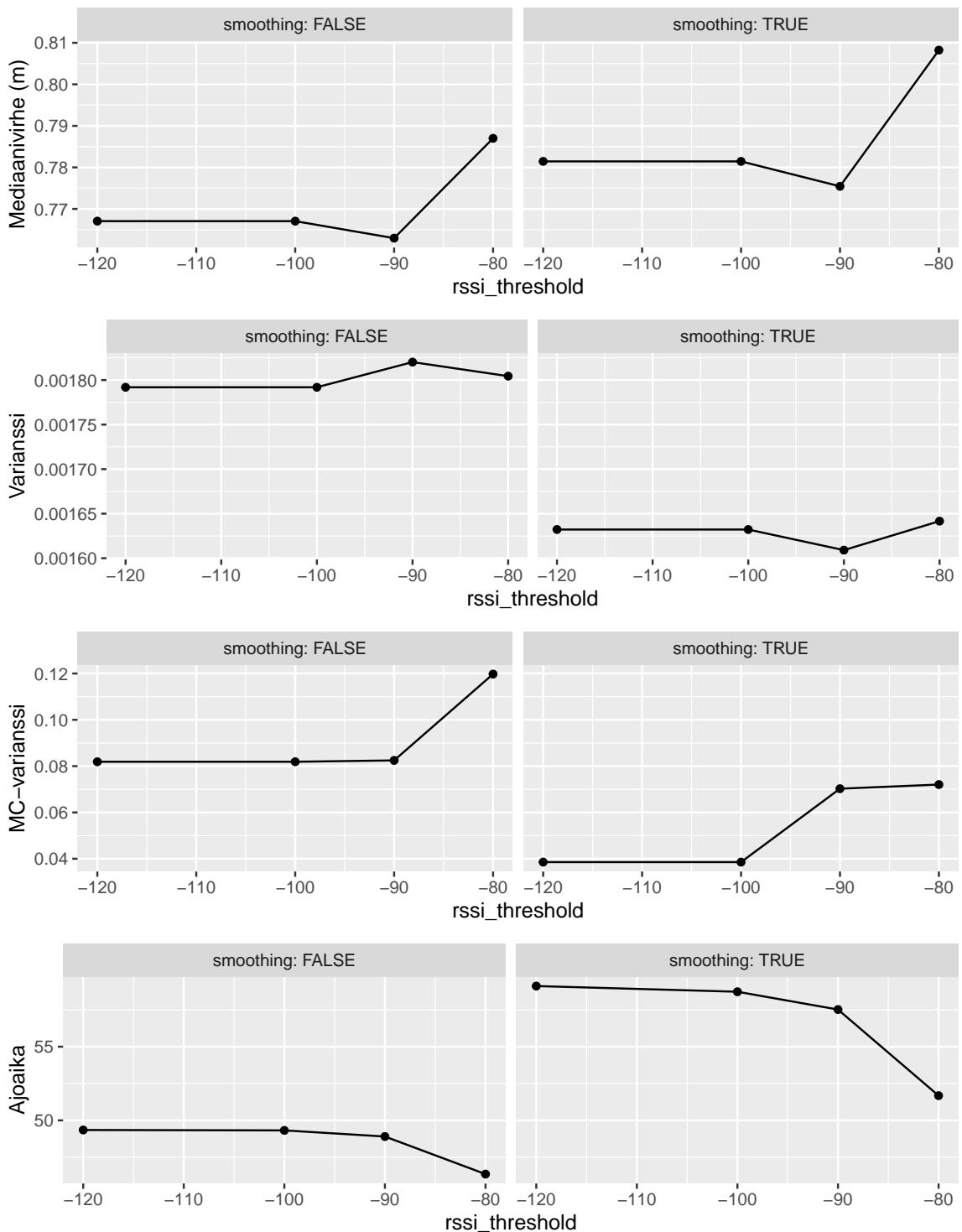
https://github.com/rintakumpu/progradu/blob/main/analyysi/data/tulokset_vaihe1_ci.csv (vaihe 1)

https://github.com/rintakumpu/progradu/blob/main/analyysi/data/tulokset_vaihe2_ci.csv (vaihe 2) ja

https://github.com/rintakumpu/progradu/blob/main/analyysi/data/tulokset_vaihe3_ci.csv (vaihe 3). Kansio <https://github.com/rintakumpu/progradu/blob/main/analyysi/data/> sisältää lisäksi RDS-muotoon tallennettuna kaikkien ajojen tulokset.

Tulosten perusteella voidaan todeta, että WB-sisätilapaikannusalgoritmi tuottaa halutun paikannusvirheen. Algoritmin testaamisessa on käytetty todelliseen liiketilaan luotua testiasetelemaa, joka vastaa hyvin järjestelmän todellista käyttötarkoitusta ja kattaa useita erilaisia radioympäristöjä. Tämän perusteella voidaan algoritmin myös olettaa yleistyvä hyvin muihin monimutkaisiin sisätilaympäristöihin. Taulukossa 4.13 on esitetty ToTal-triangulaatioalgoritmilla 13 luoduista sijaintiestimaateista laskettujen paikannusvirheiden tunnuslukuja.

Triangulaatioon verrattuna WB-sisätilapaikannusalgoritmi tuottaa huomattavasti parempia tuloksia. Tämän voi havaita myös vertaamalla kuvan 4.12 ToTal-karttapolku liitteen A WB-sisätilapaikannusalgoritmin tuottamiin karttapolkuihin.



Kuva 4.11: Vaiheen 3 tulokset

Taulukko 4.13: ToTal-triangulaatioalgoritmin tuottamia paikannusvirheen tunnuslukuja

	Mediaani	Aritmeettinen keskiarvo	Otoskeskihajonta	<1m persentiili
ToTaL	1.61	2.35	4.15	0.34



Kuva 4.12: ToTal-triangulaatioalgoritmin tuottamat sijaintiestimaatit

Luku 5

Lopuksi

Tässä tutkielmassa on esitetty pääpiirteittäin hiukkassuodin- ja hiukkassiloitinalgoritmien teoria Bayesilaisessa tilastotieteellisessä viitekehysessä. Tutkielmassa on lisäksi käyty läpi uudelleenotantaa efektiivisen otoskoon perusteella hyödyntävä SIR-suodinalgoritmi sekä käsitelty algoritmin varianssin estimointia. Tutkielmassa on myös esitetty SIR-algoritmin parametrien valintaan, suorituskykyyn sekä konvergenssiin liittyviä tuloksia.

Tutkielmassa on lisäksi esitetty WB-sisätilapaikannusalgoritmi, joka toteuttaa SIR-algoritmin, estimoi jakaumaestimaatin varianssin ja hyödyntää sisätilapai-kannuksen karttasovitusalgoritmia. Tutkielmassa on lopuksi tarkasteltu miten eri suunnitteluparametrien valinnat vaikuttavat algoritmin suorituskykyyn kattavan, todelliseen ongelmaan ja dataan perustuvan paikannuskokeen avulla.

Algoritmia ja järjestelmää voitaisiin mahdollisesti edelleen parantaa esimerkiksi käyttämällä tagia, jonka kiihtyyvyysmittarin tuottama data olisi AT-2-tagia luotettavampaa. Kiihtyyvyssataa voitaisiin hyödyntää esimerkiksi informatiivisen liikemallin luomisessa.

Koe-esimerkin perusteella tutkielmassa käsitelty ALVar-varianssi estimoi hyvin jakaumaestimaatin varianssia. ALVar-varianssiestimaattia voitaisiin edelleen hyödyntää esimerkiksi adaptiivisen viipeen siloittimen toteuttamisessa. Vastaavasti KALMAN-SUODIN.

Varianssiestimaattia olisi mahdollista hyödyntää WB-sisätilapaikannusalgoritmin dynaamisen mallin kohinaparametrin q säätämisessä niin, että kohinaa lisättäisiin tai vähennettäisiin varianssiestimaatin perusteella. Tämä voitaisiin toteuttaa esimerkiksi käyttämällä lähtökohtaisesti pienempää q -arvoa ja lisäämällä siihen keskihajonnan estimaatti, jota on painotettu suunnitteluparametrilla α , ts. ennen dynaamisen mallin sovitusta päivtettäisiin kohinaparametri $q_k = q_k + \alpha\sqrt{\hat{\sigma}_{k-1}^2}$.

Varianssiestimaattia voisi hyödyntää myös hiukkasten lukumäärän N adaptiivisessa valinnassa niin, että matala varianssi vähentää estimoinnissa käytettävien hiukkasten määrää. Tällaista varianssiin perustuva hiukkasmäärän muuttamista ehdottavat muun muassa Lee ja Whiteley (2018) [20]. Hiukkasten määrään muuttamista XXX Vastaavasti voisi parantaa myös arvioimalla konvergenssia esimerkiksi ***.

Algoritmin suorituskykyä voitaisiin mahdollisesti myös parantaa käyttämällä RSSI-kynnysarvon sijaan tai ohella esimerkiksi menetelmää, jossa datasta poistettaisiin kullakin aika-askeleella ne kulmavainiot, jotka poikkeavat kulmavainion suuntakonsensuksesta. Jättämällä osa havainnoista uskottavuusfunktioiden ulkopuolelle nopeuttaisi algoritmin laskentaa, mutta saattaisi parantaa myös algoritmin tarkkuutta.

Liitteenä olevia polkuja tarkastelemalla puolestaan huomataan, että nyt toteutetun algoritmin sijaintestimaatilla on taipumus jäädä osassa testiympäristöä jälkeen itse tagin sijainnista. Tätä ongelmaa voitaisiin mahdollisesti lieventää käyttämällä mm. Yi Chenging &al. artikkelissa “Improved Particle Filter Algorithm for Multi-Target Detection and Tracking” (2024) [5] esittämää menetelmää, jossa partikkeliteitä jaetaan ns. seurantahiukkasiin sekä etsintähiukkasiin, joista ainoastaan edellisiä käytetään sijaintestimaatin luomisessa ja jälkimmäisten annetaan liikkua suuremmilla kohina-arvoilla. Näin mahdollistetaan satunnaiskulkumallilla laajempi signaalivaruuden tutkinta ja nopeampi ongelmatilanteesta toipuminen ilman, että sijaintestimaatit kärsivät liikemalliin lisätystä kohinasta.

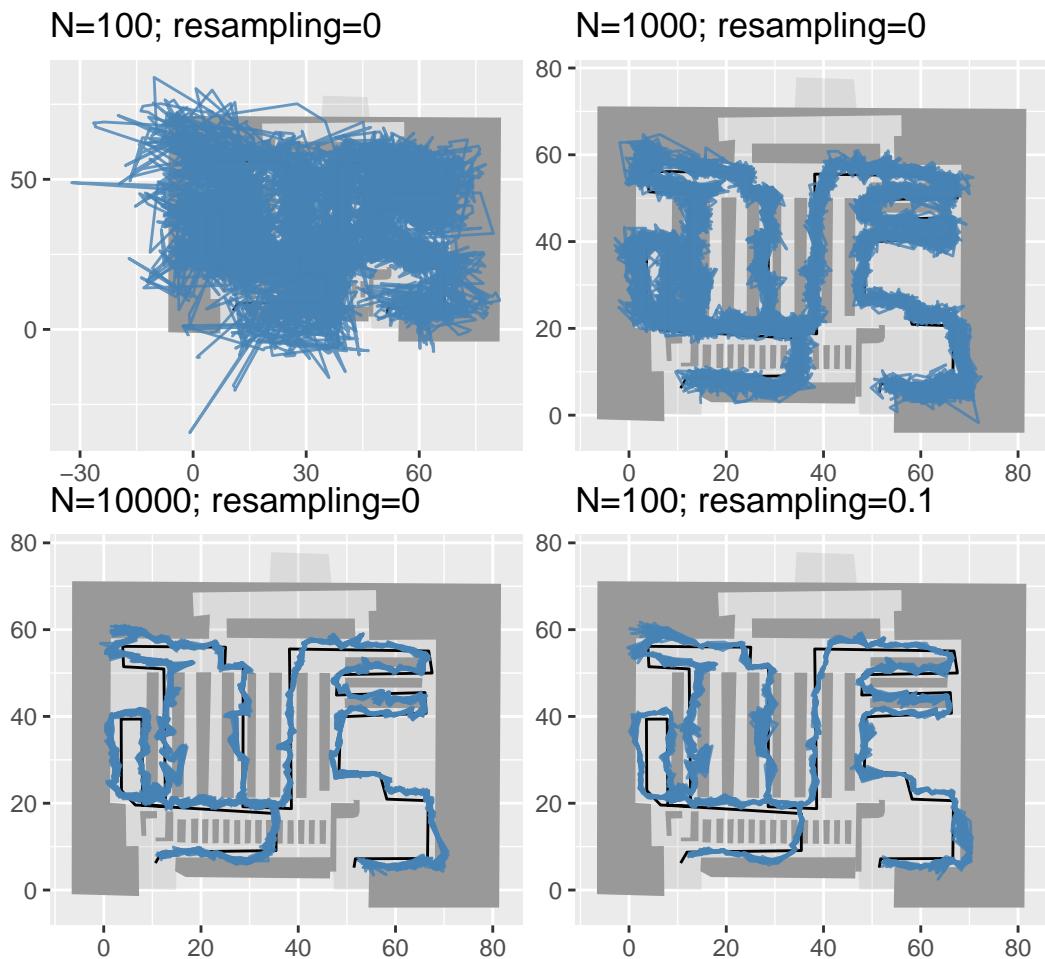
Ongelma voitaisiin ratkaista myös käyttämällä ekskluusiopolygonien ohella ns. pehmeitä rajoitteita (kts. esim. TODO), jolloin ekskluusiopolygonien ympärille luotaisiin rajoitevyöhyke, jolle osuvat hiukkaset saisivat rangaistuksen riippuen siitä, kuinka lähellä itse ekskluusiopolygonin reunaa ne ovat. Rangaistarvon luomisessa voitaisiin käyttää esimerkiksi eksponentiaalista tai kvadraattista hajoamista.

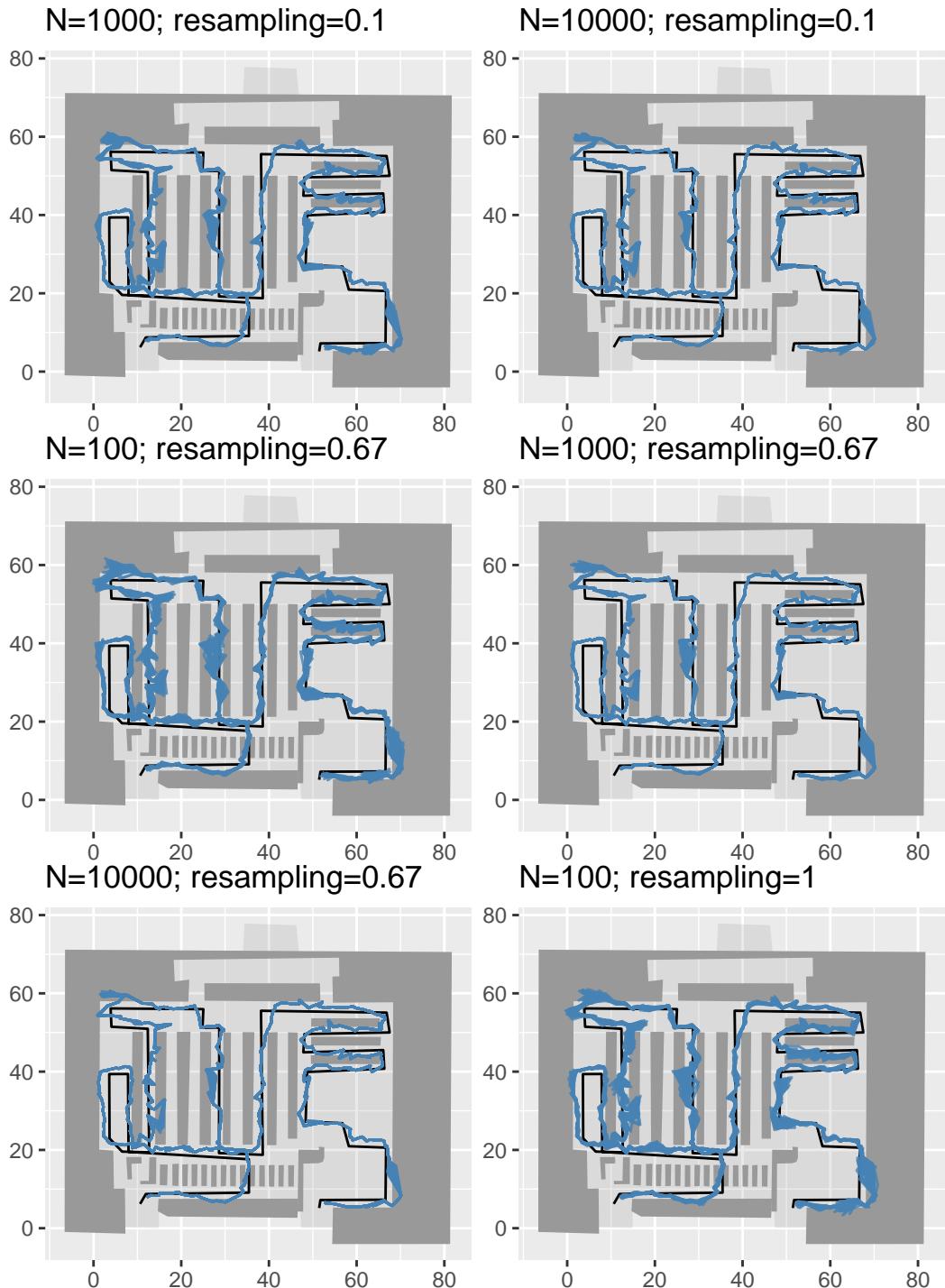
Kokeen tuloksia voisi puolestaan myös Vastaavasti parametrien vaikutusta voisi tarkastella sensitiivisyyssanalyysin avulla.

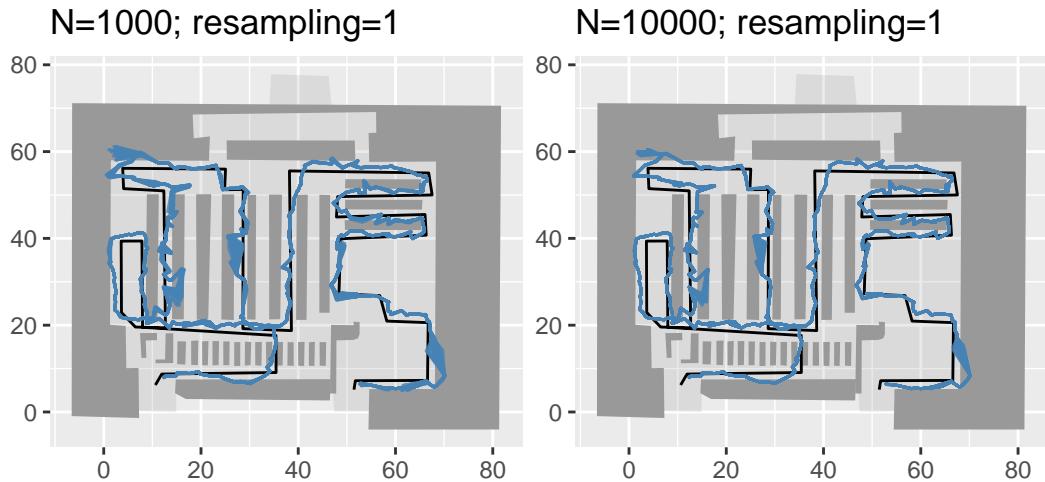
Liite A - Karttapolut

Liite sisältää tutkielman tulososioon liittyvät karttapolut. Kukin kartta käsittää $r = 30$ ajoa kullekin suunnitteluparametrikombinaatiolla. Karttojen otsikossa on mainittu ainoastaan testatut suunnitteluparametrit, vakioarvoiset parametrit on esitettty luvussa 4.5.3.

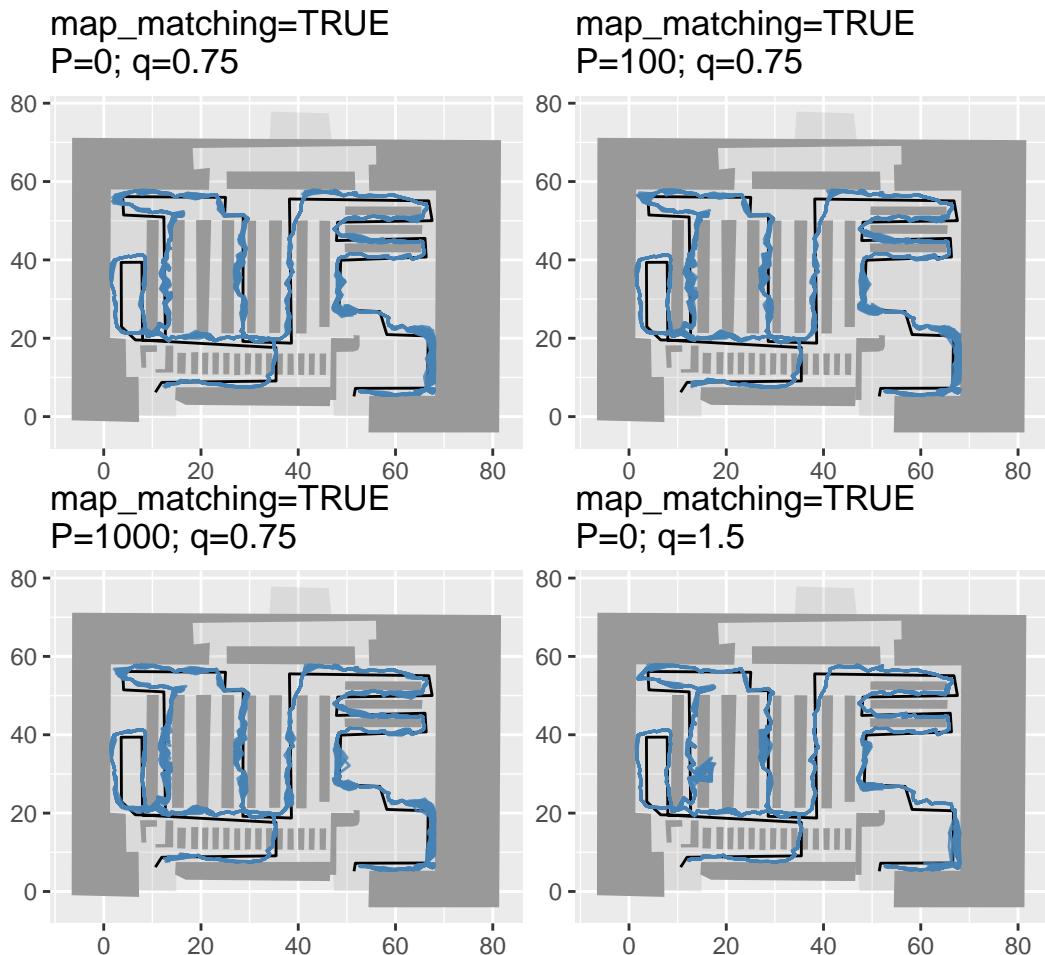
Vaihe 1

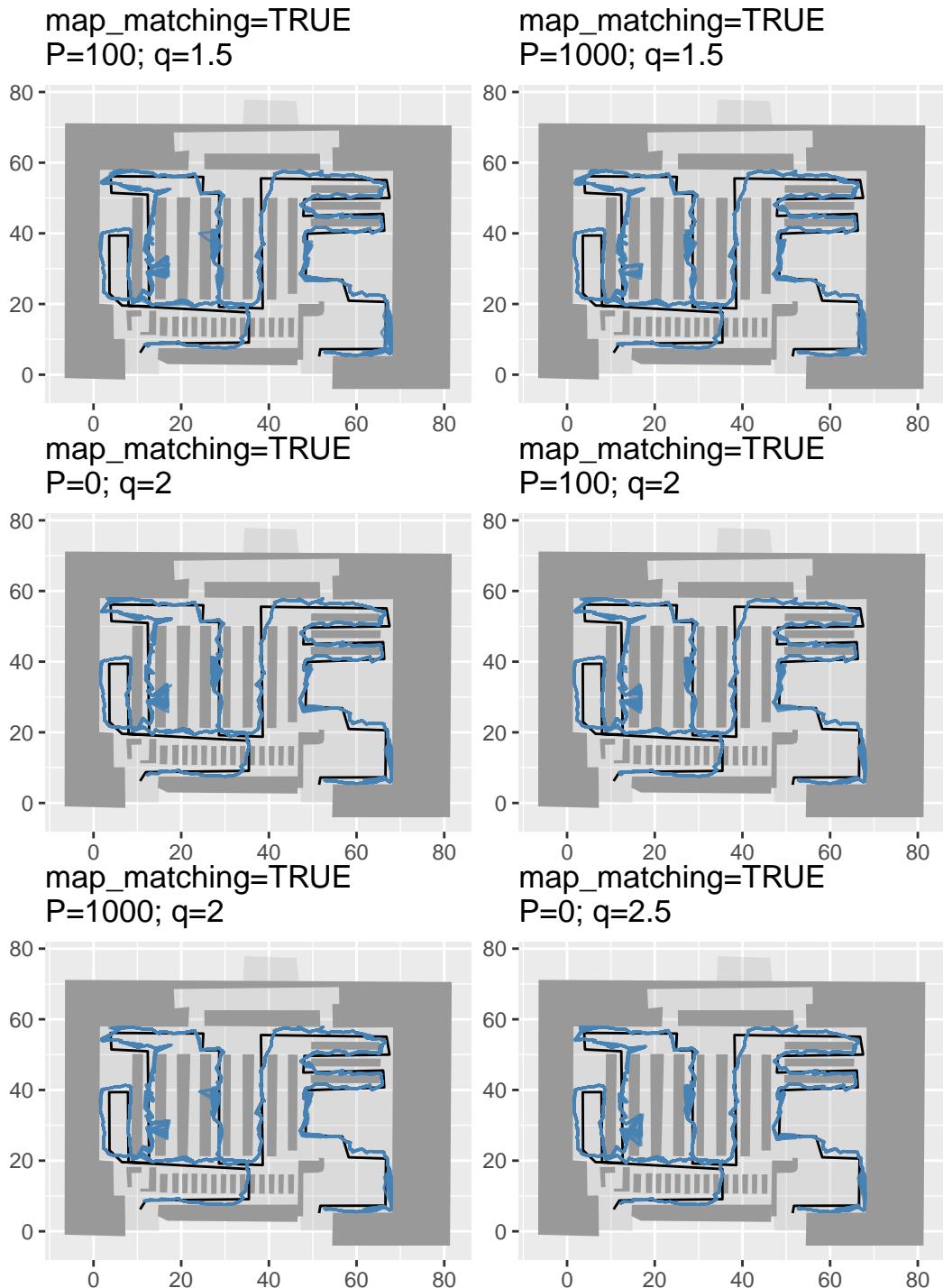


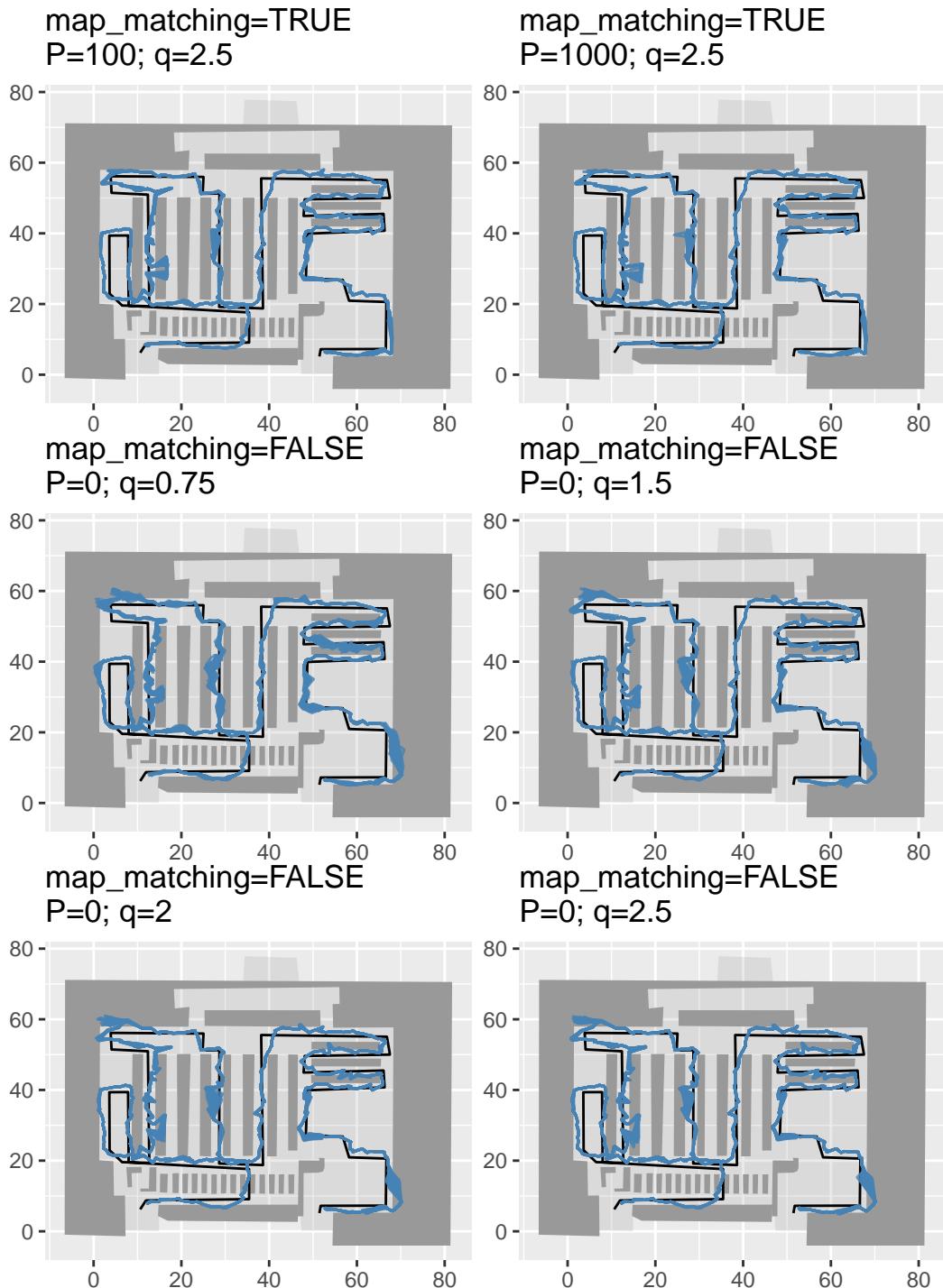




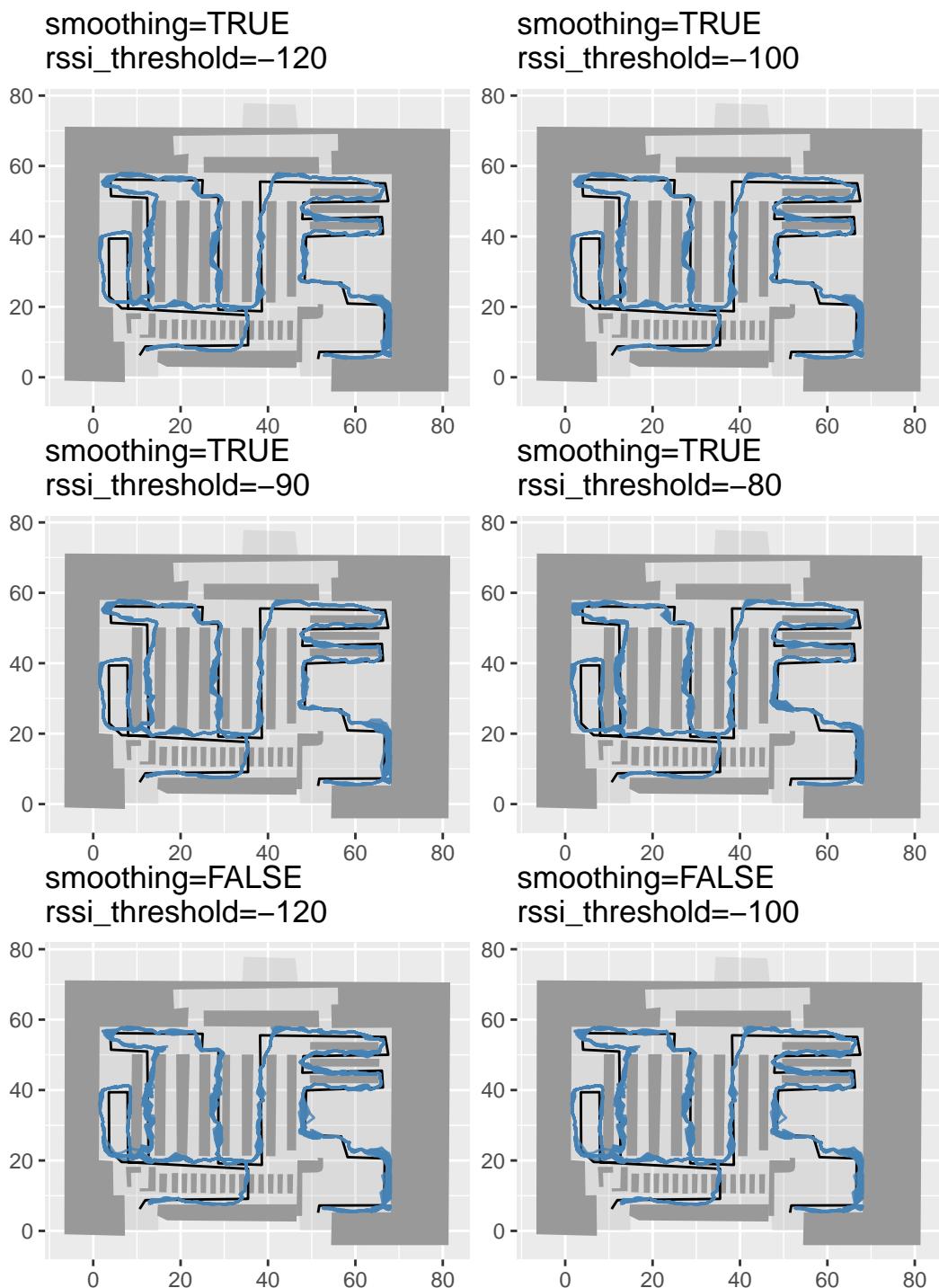
Vaihe 2



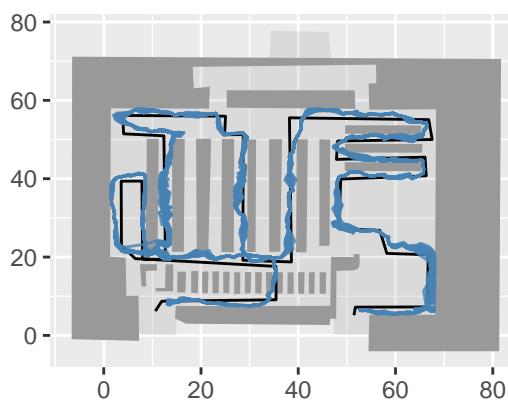




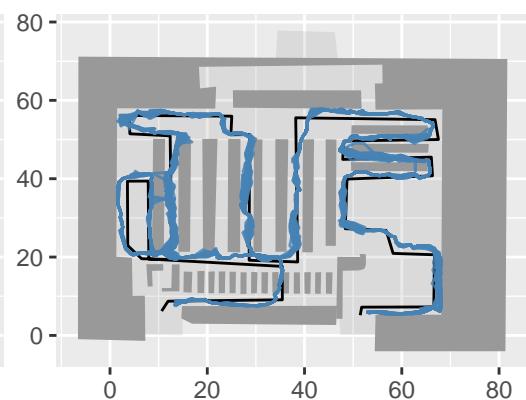
Vaihe 3



smoothing=FALSE
rssi_threshold=-90



smoothing=FALSE
rssi_threshold=-80



Lähteet

- [1] Johan Alenlöv and Jimmy Olsson. Particle-based adaptive-lag online marginal smoothing in general state-space models. *IEEE Transactions on Signal Processing*, 67(21), 2019.
- [2] Jayaprasad Bojja, Jussi Collin, Simo Särkkä, and Jarmo Takala. Pedestrian localization in moving platforms using dead reckoning, particle filtering and map matching. page 1116 –1120, 2015. doi: 10.1109/ICASSP.2015.7178143.
- [3] Olivier Cappé, Simon J. Godsill, and Eric Moulines. An Overview of Existing Methods and Recent Advances in Sequential Monte Carlo. *Proceedings of the IEEE*, 95(5):899–924, 2007.
- [4] Hock Peng Chan and Tze Leung Lai. A general theory of particle filters in hidden Markov models and some applications. *The Annals of Statistics*, 41(6):2877 –2904, 2013.
- [5] Yi Cheng, Wenbo Ren, Chunbo Xiu, and Yiyang Li. Improved Particle Filter Algorithm for Multi-Target Detection and Tracking. *Sensors*, 24(14), 2024.
- [6] Nicolas Chopin. Central limit theorem for sequential monte carlo methods and its application to bayesian inference. *The Annals of Statistics*, 32(6):2385–2411, 2004.
- [7] Dan Crisan. The stochastic filtering problem: A brief historical account. *Journal of Applied Probability*, 51A:13–22, 2014.
- [8] Dan Crisan and Arnaud Doucet. Convergence of sequential monte carlo methods. URL https://www.stats.ox.ac.uk/~doucet/crisain_doucet_convergenceofSMC2000.pdf.
- [9] Dan Crisan and Arnaud Doucet. A survey of convergence results on particle filtering methods for practitioners. *IEEE Transactions on Signal Processing*, 50(3):736–746, 2002.
- [10] Pavel Davidsonl, Jussi Collin, and Jarmo Takala. Application of particle filters for indoor positioning using floor plans. pages 1–4, 2010. doi: 10.1109/UPINLB.2010.5653830.
- [11] Arnaud Doucet, Simon Godsill, and Christophe Andrieu. On sequential monte carlo sampling methods for bayesian filtering. *Statistics and Computing*, 10(3):197–208, 2000.

- [12] N.J. Gordon, D.H. Salmond, and A.F.M Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. *IEE Proceedings F (Radar Signal Process)*, 140(2):107–113, 1993.
- [13] Mohinder S. Grewal and Angus P. Andrews. Applications of kalman filtering in aerospace 1960 to the present. *IEEE Control Systems Magazine*, 30(3):69–78, 2010.
- [14] Marco Gunia, Adrian Zinke, and Niko Joram. Analysis and design of a music-based angle of arrival positioning system. *ACM Transactions on Sensor Networks*, 19(3):66, 1–41.
- [15] Fredrik Gustafsson. Particle filter theory and practice with positioning applications. *IEEE Aerospace and Electronic Systems Magazine*, 25(7):53–82, 2010.
- [16] Monson H. Hayes. *Statistical Digital Signal Processing and Modeling*. John Wiley & Sons, Inc., 1996.
- [17] Ngoc-Huynh Ho, Phuc Huu Truong, and Gu-Min Jeong. Step-detection and adaptive step-length estimation for pedestrian dead-reckoning at various walking speeds using a smartphone. *Sensors*, 16(9), 2016.
- [18] Genshiro Kitagawa. Monte carlo filter and smoother for non-gaussian nonlinear state space models. *Journal of Computational and Graphical Statistics*, 5(1):1–25, 1996.
- [19] Mike Klaas, Nando de Freitas, and Arnaud Doucet. Toward practical n2 monte carlo: The marginal particle filter. *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence*, page 308–315, 2005. URL <https://arxiv.org/pdf/1207.1396.pdf>.
- [20] Anthony Lee and Nick Whiteley. Variance estimation in the particle filter. *Biometrika*, 105(3):609–625, 2018.
- [21] J. Liu and R. Chen. Sequential monte carlo methods for dynamic systems. *Journal of the American Statistical Association*, 93(443):1032–1044, 1998.
- [22] Alessandro Mastrototaro and Jimmy Olsson. Adaptive online variance estimation in particle filters: the alvar estimator. *Statistics and Computing*, 33(77), 2022.
- [23] Pierre Del Moral. Nonlinear filtering: Interacting particle resolution. *Markov Processes and Related Fields*, 2(4):555–580.
- [24] D. Munoz, R. Enriquez-Caldera, and C. Vargas. *Position Location Techniques and Applications*. Elsevier, 2009.
- [25] Samuel Nyobe, Fabien F. Campillo, and Serge Moto. The one step fixed-lag particle smoother as a strategy to improve the prediction step of particle filtering. *hal-03464987*, 2021. URL <https://inria.hal.science/hal-03464987v2/file/papier.pdf>.
- [26] George Oguntala, Raed Abd-Alhameed, Stephen Jones, James Noras, Mohammad Patwary, and Jonathan Rodriguez. Indoor location identification

- technologies for real-time iot-based applications: An inclusive survey. *Computer Science Review*, 30:55–79, 2018. ISSN 1574 –0137. doi: <https://doi.org/10.1016/j.cosrev.2018.09.001>. URL <https://www.sciencedirect.com/science/article/pii/S1574013718301163>.
- [27] Jimmy Olsson and Randal Douc. Numerically stable online estimation of variance in particle filters. *Bernoulli*, 25(2):1504–1535, 2019.
 - [28] V. Pierlot, M. Urbin-Choffray, and M. Van Droogenbroeck. A new three object triangulation algorithm based on the power center of three circles. *Research and Education in Robotics - EUROBOT 2011*, page 248–262.
 - [29] Arno Solin, Simo Särkkä, Juho Kannala, and Esa Rahtu. Terrain navigation in the magnetic landscape: Particle filtering for indoor positioning. *2016 European Navigation Conference (ENC)*, page 1 –9, 2016. URL <https://api.semanticscholar.org/CorpusID:23211261>.
 - [30] Simo Särkkä. *Bayesian Filtering and Smoothing*. Cambridge University Press, 2013. URL https://users.aalto.fi/~ssarkka/pub/cup_book_online_20131111.pdf.