Hiukassuodin- ja hiukassiloitinalgoritmit sekä niiden soveltaminen AoA-menetelmään perustuvassa Bluetooth-sisätilapaikannuksessa

Lasse Rintakumpu

Pro gradu -tutkielma
Maaliskuu 2024

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

MATEMATIIKAN JA TILASTOTIETEEN LAITOS

TURUN YLIOPISTO
Matematiikan ja tilastotieteen laitos

Lasse Rintakumpu: Hiukassuodin- ja hiukassiloitinalgoritmit sekä niiden soveltaminen AoA-menetelmään perustuvassa Bluetooth-sisätilapaikannuksessa
Pro gradu -tutkielma, X s.
Tilastotiede
Maaliskuu 2024

Tutkielmassa esitetään hiukassuodin- ja hiukassiloitinalgoritmien teoria Bayesilaisessa tilastotieteellisessä viitekehyksessä. Lisäksi tutkielmassa käsitellään hiukassuotimien varianssin estimointia.

Empiirisenä esimerkkinä tutkielmassa tarkastellaan hiukassuodin- ja hiukassiloitinalgoritmien käyttöä AoA-teknologiaan perustuvassa Bluetooth-sisätilapaikannusratkaisussa.

Asiasanat: SMC-menetelmät, Monte Carlo -menetelmät, sekventiaalinen Monte Carlo, suodinongelma, hiukassuodin, hiukassiloitin, SIR-algoritmi, sisätilapaikannus, BLE, AoA, triangulaatio, Bayesilainen päättely

Sisällys

Luku 1

Johdanto

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

Tämän tutkielman tavoitteena on esittää hiukassuotimien teoria sekä joitakin menetelmäperheeseen kuuluvia algoritmeja. Tutkielman ensimmäisessä luvussa kuvataan yleisellä tasolla sekä suodinongelma että sen ratkaisujen historiaa ja esitetään joitakin Monte Carlo -menetelmiin liittyviä yleisiä tuloksia sekä Bayesilainen viitekehys suodinongelmalle. Toisessa luvussa kuvataan kaksi hiukassuodinalgoritmia, saapasremmisuodin sekä SIR-algoritmi ja perehdytään hiukassuotimen varianssin estimointiin. Kolmannessa luvussa tarkastellaan suodinongelmaan läheisesti liittyvää siloitteluongelma ja esitetään hiukassiloitinalgoritmeja tämän ongelman ratkaisemiseksi. Neljäs luku keskittyy hiukassuotimen käyttöön empiirisessä AoA/Bluetooth-teknologiaan perustuvassa sisätilapaikannussovelluksessa. Tässä luvussa esitetään myös hiukassuodinalgoritmit radiosignaalin tulokulman arviointiin sekä radiovastaanottimen kalibrointiin. Lisäksi käsitellään lyhyesti sisätilapaikannuksessa hyödynnettävää karttasovitusalgoritmia.

Hiukassuodin- sekä hiukassiloitinalgoritmien osalta tutkielman esitykset seuraavat erityisesti Simo Särkän kirjaa *Bayesian Filtering and Smoothing* (2013) [24], Fredrik Gustafssonin artikkelia "Particle Filter Theory and Practice with Positioning Applications" (2010) [9] sekä Olivier Cappén, Simon J. Godsillin ja Eric Moulines'n artikkelia "An overview of existing methods and recent advances in sequential Monte Carlo" (2007) [22]. Hiukassuotimien varianssin estimointi seuraa artikkeleita TODO.

1.1 Notaatioista

Tässä tutkielmassa käytetään seuraavia notaatioita. Vektoreita merkitään pienellä kirjaimella, esimerkiksi z. Hiukassuotimen hiukkaset sisältäviä vektoreita merkitään x_k^i, missä alaindeksi viittaa ajanhetkeen $k, k = \{1, \ldots, T\}$ ja yläindeksi partikkeliin i, missä $i = \{1, \ldots, N\}$. Ajanhetkien $k, k = \{1, \ldots, T\}$ havainnot sisältäviä vektoreita merkitään $\{y_1, \ldots, y_k\}$. Lähtökohtaisesti kaikki tutkielmassa esitetyt muuttujat ovat ylä- ja alaindeksejä lukuunottamattavektoreita. Skalaareihin pyritään viittaamaan

Taulukko 1.1: Lyhenteet ja symbolit

Lyhenne tai symboli	Selitys
RTSS	*Rauch-Turn-Striebel smoother*, Rauch-Turn-Striebel-siloitin
SMC	*Sequential Monte Carlo*, sekventiaalinen Monte Carlo -menetelmä, synonyymi hiukassuotimelle
BS-PS	*Backwards simulation particle smoother*

isoilla kirjaimilla, esimerkiksi Z. Milloin tämä ei ole mahdollista, selviää muuttujan skalaariarvoisuus asiayhtedestä. Prosesseihin viitataan alaindeksoidulla isolla kirjaimella, esimerkiksi X_k. Matriiseja merkitään isolla lihavoidulla kirjaimella, esimerkiksi $\mathbf{X}$ ja funktiota TODO. Taulukossa 1.1 esitetään tutkielman keskeisimmät lyhenteet ja symbolit.

1.2 Suodinongelma

Stokastisten prosessien teoriassa suodinongelmaksi kutsutaan tilannetta, jossa halutaan muodostaa keskineliövirheen mielessä 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).

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

Tässä tutkielmassa keskitytään erityisesti epälineaarisen, ns. Markovin piilomallin posteriorijakauman Bayesilaiseen ratkaisuun. Ongelmassa tiedetään, miten havaitut muuttujat y_k kytkeytyvät "piilossa oleviin" tilamuuttujiin x_k sekä osataan sanoa jotain tilamuuttujien todennäköisyyksistä. Oletetaan myös, että piilossa oleville tiloille X_k pätee Markov-ominaisuus, jolloin kutakin hetkeä seuraava tila x_{k+1} riippuu menneistä tiloista $x_{1:k}$ 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 ajanhetkellä $k = \{1, \dots, t\}$ piilossa olevan prosessin tilaa merkitään x_k ja havaittua prosessia y_k, saadaan mallit

$$x_{k+1} = f(x_k, \nu_k), \tag{1.2}$$
$$y_k = h(x_k) + e_k. \tag{1.3}$$

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

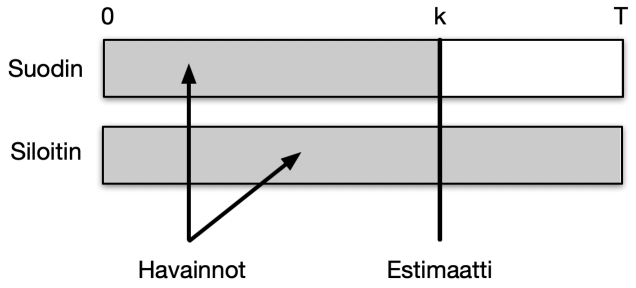
hiukassuodinalgoritmit pyrkivät ratkaisemaan juurikin epälineaarisen, ei-Gaussisen suodinongelman, voivat funktiot $f(\cdot)$ ja $h(\cdot)$ olla epälineaarisia eikä kohinan tarvitse olla normaalijakautunutta.

Mallit voidaan esittää myös yleisemmässä jakaumamuodossa

$$x_{k+1} \sim p(x_{k+1}|x_k), \tag{1.4}$$
$$y_k \sim p(y_k|x_k). \tag{1.5}$$

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 $p(x_k|y_k)$ jokaisena ajanhetkenä $\{1, \ldots, k\}$ ei ainoastaan haluttuna ajanhetkenä k. Hiukassuodinalgoritmit 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 1.1: Suodin- ja siloitteluongelma

1.3 Suodin- ja siloitteluongelmien historiaa

Tämä alaluku esittää pääpiirteittäin suodinongelmalle esitettyjen ratkaisujen historian. Lineaarisen suodinongelman osalta alaluku noudattaa Dan Crisanin artikkelia "The stochastic filtering problem: a brief historical account" (2014) [5] sekä Mohinder S. Grewalin ja Angus P. Andrewsin artikkelia "Applications of Kalman Filtering in Aerospace 1960 to the Present" (2010) [19]. Hiukassuotimien osalta lähteenä toimii Cappé &al (2007) [22].

Suodinongelma nousi esille insinööritieteiden 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 sallivan suotimen esitti matemaatikko, kybernetiikan kehittäjä Norbert Wiener. Wiener-suotimena tunnettua ratkaisuaan

varten Wiener muotoili seuraavat kolme ominaisuutta, jotka prosessin X estimaatin $\hat{X}_t$ pitää toteuttaa.

1. *Kausaliteetti*: X_t tulee estimoida käyttäen arvoja Y_s, missä $s \leq t$.
2. *Optimaalisuus*: X_t:n estimaatin $\hat{X}_t$ tulee minimoida keskineliövirhe $\mathbb{E}[(X - \hat{X}_t)^2]$.
3. *On-line -estimointi*: Estimaatin $\hat{X}_t$ tulee olla saatavissa minä hyvänsä ajanhetkenä t.

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

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 säätö- ja paikannussovelluksissa Kalman-suotimen käyttö on luontevampaa. Suotimien oletuksia ja oletusten välisiä eroja ei käsitellä tässä tutkielmassa, mutta alaluvussa 1.6. TODO:LINKKI käsitellään Kalman-suotimen formaalia yhteyttä hiukassuotimiin.

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 lentoratojen laskennassa Kalman-suotimen perusversiota. Laajennetun Kalman-suotimen toimintaperiaate perustuu epälineaaristen funktioiden linearisointiin Taylorin kehitelmän avulla kulloisenkin estimaatin ympärillä. Laajennettu Kalman-suodin on erityisesti paikannussovellusten *de facto* -suodinstandardi, mutta suodin ei kuitenkaan ole epälineaarisen ongelman optimaalinen estimaattori.

Kalman-suotimesta on lisäksi olemassa lukuisia muita epälineaarisiin ongelmiin soveltuvia 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 mallia ei tunneta tarkasti tai kohinaa ei voida olettaa normaalijakautuneeksi, ovat hiukassuotimet eli sekventiaaliset Monte Carlo -menetelmät Kalman-suotimen johdannaisia parempia ratkaisuja. Vaikka

tila-avaruuden dimensioiden kasvaessa kasvaa myös hiukassuotimien vaatima laskentateho, ovat hiukassuotimet aina sitä parempia mitä epälineaarisempia mallit ovat ja mitä kauempana normaalijakaumasta kohina on. Viimeisten vuosikymmenten aikana myös laskennan teho on kasvanut merkittävästi samalla kun laskennan hinta on vastaavasti romahtanut, mikä puoltaa Monte Carlo -menetelmien käyttöä entistä useammissa ongelmissa.

Joitakin suodinongelman rekursiivisia Monte Carlo -ratkaisuja löytyy jo 1950–70-luvuilta, erityisesti säätöteoriaan piiristä. Olennainen nykyalgoritmeihin periytynyt oivallus varhaisissa suodinalgoritmeissa oli tärkeytysotannan käyttö halutun jakaumaestimaatin laskennassa. Tärkeytysotanta-algoritmiin voidaan turvautua, kun emme pysty suoraan tekemään havaintoja jostakin jakaumasta p ja teemme sen sijaan havaintoja jakaumasta q, jota painotamme niin, että tuloksena saadaan jakauman p harhaton estimaatti. Algoritmi on kuvattu tarkemmin tutkielman alaluvussa 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 päätyy vain tietyille partikkeleille, jolloin vastaavasti suuri osa partikkeleista ei enää estimoi haluttua jakaumaa. Tähän ongelmaan palataan myöhemmin.

Merkittävän ratkaisun ehtymisongelmaan esittivät Gordon, Salmond ja Smith artikkelissaan "Novel approach to nonlinear/non-Gaussian Bayesian state estimation" (1993). [21] Artikkelin ratkaisu kulki nimellä *bootstrap filter*, saapasremmisuodin. Saapasremmisuodin vältti 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 myös partikkeleiksi ja menetelmiä siten hiukassuotimiksi. Erityisesti myöhemmin esitettävää SIR-algoritmia kutsutaan usein hiukkassuotimeksi. Termiä hiukkassuodin käytti ensimmäisen kerran Del Moral artikkelissa "Nonlinear Filtering: Interacting Particle Resolution" (1996) [20], SMC-menetelmät termiä Liu ja Chen artikkelissa "Sequential Monte Carlo Methods for Dynamic Systems" (1998) [11]. Tässä tutkielmassa käytetään yleisemmin käytettyä termiä hiukassuotimet.

1.4 Monte Carlo -menetelmistä

Tässä alaluvussa kuvataan lyhyesti hiukassuotimissa käytettävien Monte Carlo -menetelmien perusperiaate todennäköisyysjakauman estimoinnissa. Lisäksi esitetään tärkeytysotanta-algoritmi (*importance sampling*), jonka tarkoituksena 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)

[24].

1.4.1 Monte Carlo -approksimaatio

Bayesilaisessa päättelyssä ollaan yleisesti kiinnostuttu laskemaan johonkin posterioritiheysjakaumaan p liittyvää odotusarvoa

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

missä g on tila-avaruuden mielivaltainen funktio ja $p(x|y_{1:t})$ on havaintoihin $\{y_1, \ldots, y_k\}$ liittyvä x:n posterioritiheysjakauma. 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 perusperiaate on tehdä riippumattomia otoksia estimoitavasta jakaumasta ja laskea haluttu odotusarvo otosten avulla. Jos tehdään N otosta jakaumasta $x^i \sim p(x|y_{1:t})$, missä $i = \{1, \ldots, N\}$ saadaan näiden otosten avulla laskettua odotusarvon estimaatti

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

Monte Carlo -estimaatti konvergoi keskeisen raja-arvolauseen nojalla ja sen estimointivirheen voidaan osoittaa olevan luokkaa $O(\frac{1}{\sqrt{N}})$ riippumatta tilamuuttujan x dimensiosta. Hiukkassuotimet hyödyntävät Monte Carlo -estimointia sekventiaalisesti, jolloin estimaatti lasketaan rekursiivisesti kullekin ajanhetkelle $k = \{1, \ldots, t\}$. Tähän palataan luvuissa 3 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. tärkeytysotantaa. 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. Kirjoitetaan halutun posteriorijakauman odotusarvo integraalina

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

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

Muodostetaan näin odotusarvo

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

missä $g(x)$ on jokin estimoinnissa hyödyllinen, mielivaltainen funktio. Tutkielmassa käytetty notaatio x_k^i viittaa ajanhetken k partikkeliin i, missä $i = \{1, \ldots, N\}$. Tärkeytysotantaa kuvaa nyt algoritmi (1). Kun posteriorijakauman estimaatti muodostetaan kyseisellä algoritmilla voidaan tulos kirjoittaa

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

missä $\delta(x)$ on Diracin deltafunktio.

Algoritmi 1: Tärkeytysotanta

begin
  for $i = 1, 2, \ldots, N$ **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 analyyttisesti, pitää estimoinnissa käyttää numeerisia menetelmiä. SMC-menetelmissä tämä tarkoittaa jakauman sekventiaalista Monte Carlo -approksimointia, jonka toteutus esitetään alaluvun 4 algoritmissa. Molemmat esitykset noudattavat Gustafssonia (2010).

Bayesilainen ratkaisu tilavektorin posteriorijakauman estimaatille $\hat{p}(x_k|y_{1:k})$ saadaan seuraavalla rekursiolla (käydään läpi jokaiselle ajanhetkelle $k = \{1, \ldots, t\}$). Lasketaan ensin

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

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

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

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

Lopuksi lasketaan päivitysaskel ajalle, joka saadaan edelleen kokonaistodennäköisyydellä

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

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

1.6 Kalman-suotimen ja hiukassuotimen yhteydestä ja eroista

Tässä alaluvussa käsitellään lyhyesti Kalman-suotimen yhteyttä hiukassuotimeen edellä estitetyn teorian valossa. Esitys noudattaa Särkkää (2013). [24] 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

$$x_k = \mathbf{A}_{k-1}x_{k-1} + q_{k-1}, \tag{1.14}$$
$$y_k = \mathbf{H}_k x_k + r_k \tag{1.15}$$

missä $\mathbf{A}_{k-1}$ on dynaamisen mallin tilasiirtymään kuvaava matriisi ja $\mathbf{H}_k$ on havaintojen mallimatriisi. Normaalisuusoletuksesta puolestaan seuraa, että sekä mallin että prosessin kohinavektorit noudattavat normaalijakaumia $q_{k-1} \sim \mathbb{N}(0, \mathbf{Q}_{k-1})$ ja $r_k \sim \mathbb{N}(0, \mathbf{R}_k)$, missä $\mathbf{Q}_{k-1}$ ja $\mathbf{R}_k$ ovat kovarianssimatrsiiseja. Lisäksi oletetaan, että prosessin priorijakauma on normaali eli $x_0 \sim \mathbb{N}(m_0, \mathbf{P_0})$. Mallit voidaan nyt kirjoittaa tiheysfunktiomuodossa

$$p(x_k|x_{k-1}) = \mathbb{N}(x_k|\mathbf{A}_{k-1}x_{k-1}, \mathbf{Q}_{k-1}) \tag{1.16}$$
$$p(y_k|x_k) = \mathbb{N}(y_k|\mathbf{H}_k x_k, \mathbf{R}_k) \tag{1.17}$$

,

joista voidaan edelleen johtaa suodinongelman mallit

$$p(x_k|y_{1:k-1}) = \mathbb{N}(x_k|m_k^*, \mathbf{P}_k^*) \tag{1.18}$$
$$p(x_k|y_{1:k}) = \mathbb{N}(x_k|m_k, \mathbf{P}_k) \tag{1.19}$$
$$p(y_k|y_{1:k-1}) = \mathbb{N}(y_k|\mathbf{H}_k m_k^*, \mathbf{S}_k) \tag{1.20}$$

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 P_0.

begin
  for $k = \{1, 2, \ldots, t\}$ **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 tiettyjen ehtojen vallitessa. Tämän lyhyen alaluvun tarkoituksena oli esittää, että Kalman-suotimessa ongelma on samaa muotoa kuin hiukassuotimessa, joten linearisoituja Kalman-suotimia ei tässä käsitellä. Hiukassuodin myös ratkaisee ongelman mille hyvänsä epälineaariselle mallille.

Luku 2

Hiukassuotimet

2.1 SIR-algoritmi

Tässä alaluvussa esitetään SMC-menetelmiin kuuluva SIR-algoritmi, epälineaarisen suodinongelman ratkaisemiseksi. Algoritmi on numeerinen toteutus alaluvussa 3 kuvatusta Bayesilaisesta suotimesta. Esitetty algoritmi perustuu Gustafssoniin (2010). 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 $i = \{1, 2, \ldots, N\}$ jokaisella ajanhetkellä $k = \{1, 2, \ldots, t\}$.

Seuraava toistetaan jokaiselle ajanhetkelle $k = \{1, 2, \ldots, t\}$. Algoritmin ensimmäisessä vaiheessa päivitetään painot yhtälön (2.1) mukaan.

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

Tämä vastaa yllä 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 $\sum_{i=1}^N w_{k|k}^i = 1$.

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

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

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

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

kynnysarvoehdon $\hat{N}_{eff} < N_{th}$ täyttessä, jolloin uudelleenotantaa kutsutaan adaptiiviseksi uudelleenotannaksi. Tällaista uudelleenotantaa hyödynnetään esitetyssä algoritmissa (3) . Uudelleenotantaa tarkastellaan lähemmin alaluvussa 4.1.2. Lopuksi päivitetään aika (jos $k < t$), luodaan uudet ennusteet partikkeleille ehdotusjakaumasta (2.4)

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

ja päivitetään partikkelien painot tärkeytysotannalla (2.5), sen mukaan kuinka todennäköisiä partikkelien ennusteet ovat

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

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

Alla käsitellään algoritmiin liittyvän uudelleenotantamenetelmän, partikkelien määrän ja ehdotusjakauman valinta. Lopuksi esiteetään algoritmin konvergenssia, marginaalijakaumaa sekä aikakompleksisuutta koskevia tuloksia.

Algoritmi 3: SIR

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

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

begin

  for $k = \{1, 2, \ldots, t\}$ **do**

    for $i = \{1, 2, \ldots, N\}$ **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 partikkeleille ehdotusjakaumasta $x_{k+1}^i \sim q(x_{k+1}|x_k^i, y_{k+1})$,
        päivitetään partikkelien painot tärkeytysotannalla.

2.1.1 Parametrien valinta

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

2.1.1.1 Otoskoon N valinta

Yleispätevää sääntöä otoskoon/partikkelien lukumäärän N valinnalle on vaikeaa antaa, sillä vaadittava estimointitarkkuus riippuu usein käsillä olevasta ongelmasta. Gordon &al. (1993) esittävät kuitenkin kolme tekijää, jotka vaikuttavat partikkelien lukumäärän valintaan

a. tila-avaruuden ulottuvuuksien lukumäärä n_x,

b. tyypillinen päällekäisyys 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 ulottuvuus pystytään kattamaan. 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 käytettävissä olevan laskentatehon sekä vaadittavan laskentanopeuden kannalta järkevää. Tähän palataan tutkielman lopuksi empiirisessä paikannusesimerkissä.

2.1.1.2 Uudelleenotantamenetelmän valinta

Ilman uudelleenotantaa on mahdollista, että algoritmi alkaa kärsiä SIS-algoritmille ominaisesta 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 siten lisää 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

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

Näin laskettu 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 effektiiviselle otoskoolle ajanhetkellä k laskennallinen approksimaatio

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

Sekä määritelmälle (2.6) että (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) esittää uudelleenotannan kynnysarvoksi esimerkiksi $\hat{N}_{th} = 2N/3$.

Uudelleenotanta ei muuta approksimoitavan jakauma p odotusarvoa, mutta se lisää jakauman Monte Carlo -varianssia. On kuitenkin olemassa esimerkiksi osittamiseen perustuvia uudelleenotantamenetelmiä, 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 tarkoituksenmukaista, erityisesti jos kyseessä on reaaliaikainen sovellutus. Kirjoitetaan ehdotusjakauma muodossa

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

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 ajanhetkellä k (tasoitusongelmassa tarvitsisimme koko polun $x_{1:k}$). Alla tarkastellaan edelleen Gustafssonia (2010) 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

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

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

$$w^i_{k|k} \propto w^i_{k-1|k-1} \frac{p(y_k|x^i_k)p(x_k|x^{k-1})}{q(x_k|x^i_{k-1}, y_k)}. \tag{2.10}$$

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

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

Yhtälön (2.11) perusteella saadaan edelleen prioriotannan painoiksi

$$w^i_{k|k} = w^i_{k|k-1}p(y_k|x^i_k) = w^i_{k-1|k-1}p(y_k|x^i_k). \tag{2.12}$$

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

$$p(x_k|x^i_{k-1}, y_k) = p(y_k|x_k)\frac{p(x_k|x^i_{k-1})}{p(y_k|x^i_{k-1})}. \tag{2.13}$$

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 (2.14)

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

Yhtälön (2.14) perusteella saadaan edelleen uskottavuusotannan painoiksi (2.15).

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

2.1.2 Konvergenssituloksia

Alla esitetään kaksi SIR-algoritmiin liittyvää konvergenssitulosta. Se, kuinka hyvin esitetyllä algoritmilla arvioitu posterioritiheys $\hat{p}(x_{1:k}|y_{1:k})$ approksimoi todellista tiheysfunktiota $p(x_{1:k}|y_{1:k})$ sekä mikä on approksimaation keskineliövirhe. Tulokset 1–2 noudattavat Crisanin ja Doucet'n artikkeleita "Convergence of Sequential Monte Carlo Methods" (2000) [7] ja "A Survey of Convergence Results on Particle Filtering Methods for Practitioners" (2002) [8], tulos 3 Chopinin artikkelia "Central limit theorem for sequential Monte Carlo methods and its application to Bayesian inference" (2004) [4].

Konvergenssitulos 1: Kun $N \to \infty$ algoritmille pätee $\forall k$ tulos (2.16).

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

Konvergenssitulos 2: Keskineliövirheelle pätee asymptoottinen konvergenssi (2.17).

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

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 $p_k = p < \infty$.

Konvergenssitulos 3: Keskeinen raja-arvolause (2.18).

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

,

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 1 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 estimaattia

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

Toinen, tarkempi vaihtoehto on käyttää laskennassa tärkeytyspainoa

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

painon (2.5) sijaan. Tällöin jokaisella aikapäivitysaskeleella lasketaan painot kaikkien mahdollisten tila-aika-avaruuspolkujen yli. Samoin kuin uudelleenotanta tämä pienentää painojen varianssia.

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.

Tällaisia tilanteita varten algoritmista on olemassa $\mathcal{O}(N\log(N))$ -versioita, jotka perustuvat esimerkiksi N:n kappaleen oppimiseen (N-body learning). Näiden algoritmien käsittely jää tämän tutkielman ulkopuolelle, mutta katsauksen algoritmeista ovat esittäneet esimerkiksi Klaas &al. artikkelissa "Toward Practical N^2 Monte Carlo: the Marginal Particle Filter" (2012).

2.2 Saapasremmisuodin

Saapasremmisuodin 4 eli *bootsrtrap filter* on SIR-algoritmin muunnelma, jossa tärkeytysotannan (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 $i = \{1, \ldots, N\}$ 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 ~ 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
                 p̂(x_{1:k}|y_{1:k}) = Σ_{i=1}^N w_{k|k}^i δ(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 määrän hiukkasia. Saapasremmisuodin esitetään tässä sen historiallisen tärkeyden vuoksi, sillä kyseessä oli ensimmäinen uudelleenotataantaa hyödyntävä hiukassuodinalgoritmi. Suotimen käytännön toteutukseen palataan luvussa 4.

2.3 Varianssin estimoinnista

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

$$\hat{\sigma}^2_{MC} = \mathrm{Var}_{\mathrm{MC}}(\hat{p}(x_{1:k}|y_{1:k})) = \frac{1}{M-1}\sum_{i=1}^{M}(x_k^i - \bar{x_k})^2. \tag{2.21}$$

,

missä x_k^i on k:nen ajanhetken piste-estimaatti ajolle $i = 1, \ldots, M$ ja $\bar{x_k}$ piste-estimaattien aritmeettinen keskiarvo laskettuna kaikkien M ajojen yli. TODO: vektoroi. Tällaisen Monte Carlo -varianssin estimoiminen on kuitenkin laskennallisesti tehotonta. Monissa käytännön sovelluksissa jo yhden hiukassuodinalgoritmin ajaminen vaatii runsaasti laskentatehoa, jolloin Monte Carlo -varianssin laskeminen

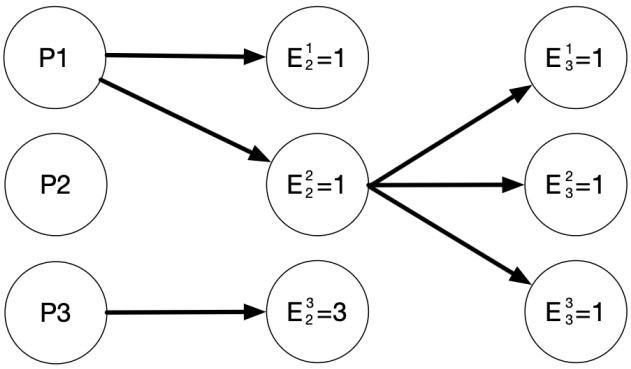
ei ole mahdollista. Varianssia ei voi myöskään laskea analyyttisesti, mutta koska keskeisen raja-arvolauseen 2.18 nojalla tiedetään, että asymptoottinen varianssi

$$\lim_{N\to\infty} N\mathrm{Var}(\hat{g}(x_k^i)) \tag{2.22}$$

,

on olemassa, on sen estimointiin on lähivuosina kehitetty joitakin menetelmiä. Alla käsitellään Leen ja Whitleyn (2018) ehdottamaa varianssin estimointitapaa. [2]

Ajetaan SIR-algoritmi kuten esitetty algoritmissa 3, mutta merkitään kullakin algoritmin suorituskerralla $k = 1, \ldots, T$ ja jokaisella partikkelilla $n = 1, \ldots, N$ indeksillä E_k^n kunkin hiukkasen kantaisää eli toisin sanoen sitä hiukkasta, josta kyseinen hiukkanen on uudelleenotantojen kaltta polveutunut ajanhetkestä $k = 1$ lähtien. Kaavio 2.1 havainnollistaa hiukkasten 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:

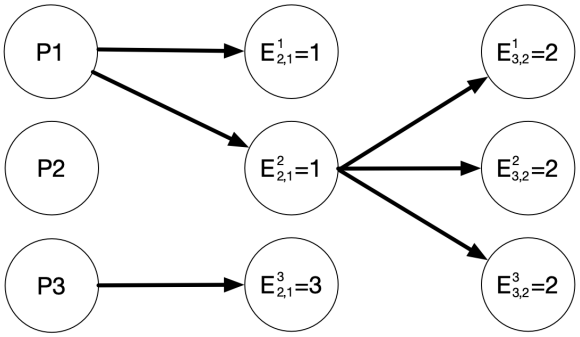
$$\hat{\sigma}^2_{CLT} = \frac{1}{N^2}\left[(\sum_{i=1}^{N}\gamma(x_k^i))^2 - (\frac{N}{N-1})^{n+1}(\sum_{i=1}^{N}\sum_{j:E_n^j=i}\gamma(x_k^i)\gamma(x_k^j))\right] \tag{2.23}$$

,

missä $\gamma : \mathbf{X}_n \to \mathbb{R}$ on rajoitettu, $\mathcal{X}_n$-mitallinen funktio. SIR-algoritmin kohdalla jakaumaestimaatti $\hat{p}(x_{1:k}|y_{1:k}))$. Kyseessä on harhaton ja konsistentti asymptoottisen varianssin estimaatti. Tarkemmin $N\hat{\sigma}^2_{CL}$ konvergoi asymptoottiseen varianssiin 2.22. Tätä ei tutkielman puitteissa todisteta.

Yllä esitetty varianssiestimaatti kärsii kuitenkin epätarkkuudesta, sillä kun $k \to \infty$ polveutuvat kaikki hiukkaset lopulta samasta kantaisästä eli indeksit $E_k^i, \ldots, E_k^N$ ovat kaikki yhtäsuuria. Tämän vuoksi on mielekästä johtaa indeksit ainoastaan tietystä aiemmasta ajanhetkestä alkaen. Olsson ja Douc (2019) [15] ehdottavat tähän tarkoitukseen Henok-indeksiä $E_{k,m}^n$, jossa m merkitsee ajanhetken $m < k$ hiukkasta,

josta kyseinen hiukkanen polveutuu. Hiukkasten kantaisien sukupolvi määritetään viipeellä λ niin, että $m = k - \lambda$. Kaavio 2.2 havainnollistaa hiukkasten polveutumista, kun $\lambda = 1$.



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

Nyt varianssi saadaan muotoon

$$\hat{\sigma}^2_{OD} = \frac{1}{N^2}\left[(\sum_{i=1}^{N}\gamma(x_k^i))^2 - (\frac{N}{N-1})^{n+1}(\sum_{i=1}^{N}\sum_{j:E^j_{n,k(\lambda)}=i}\gamma(x_k^i)\gamma(x_k^j))\right] \tag{2.24}$$

,

missä $k(\lambda) := k - \lambda$. 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. Tämän jälkeen estimaatti voi myös alkaa kärsiä samasta epätarkkuudesta kuin Eeva-indekseihin perustuva CLT-estimaatti.

Mastrototaro ja Olsson (2023) [1] laajentavat tätä estimaattia edelleen niin, että viive λ valitaan mukautuvasti. Mastrototaron ja Olssonin ALvar-estimaatti (*Adaptive-Lag variance*) lasketaan, kuten OD-varianssi edellä 2.24, mutta kunkin algoritmin ajokerran jälkeen asetetaan seuraavan ajanhetken λ seuraavasti:

$$\lambda_{k+1} \leftarrow \underset{\lambda\in[0,\lambda_k+1]}{\arg\max}\ \hat{\sigma}^2_{k+1,\lambda}(\gamma_{k+1}) \tag{2.25}$$

,

Tämä perustuu havaintoon, jonka mukaan ehtyneille Henok-indekseille on olemassa $\lambda' \in [0, \lambda - 1]$, joka täyttää ehdon $\hat{\sigma}^2_{k+1,\lambda}(\gamma_{k+1}) < \hat{\sigma}^2_{k+1,\lambda'}(\gamma_{k+1})$. Koska myös ehtyneiden indeksien kantaisät ovat ehtyneitä. Nyt indeksi λ_{n+1} voidaan valita rekursiivisesti niin, että se tuottaa suurimman varianssiestimaatin, joka on kuitenkin rajoitettu ylhäältä arvoon $\lambda_n + 1$. 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 hiukassiloittimen viipeen

valinnassa (kts. luku 4 LINKKI). Varianssiestimaattia hyödynnetään myös luvun 5 empiirisen esimerkin uskottavuusfunktioiden parametrien estimoinnissa.

Luku 3

Hiukassilottimet

Tässä luvussa käsitellään suodinongelmaan läheisesti liittyvän hiukassiloittimen ratkaisemista ns. hiukassiloitinalgoritmien avulla. Kuten hiukassuotimien 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 hiukassuodinalgoritmin ajon jälkeen sekä online-algoritmeihin, jotka suoritetaan yhdessä hiukassuodinalgoritmin kanssa. Siloitinongelman esittely seuraa Särkkää (2013) [24]. Algoritmien käsittely pohjautuu SIR-, BSPS- ja TODO-siloittimien osalta niin ikään Särkkään (2013) [24]. TODO MUUT.

3.1 Bayesilainen siloitin

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

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

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

,

missä $p(x_k|y_{1:k})$ on suodintiheys ajanhetkellä k ja $p(x_{k+1}|y_{1:k})$ prediktiivinen jakauma ajanhetkelle $k+1$. Kuten suodinongelman kohdalla, voidaan ongelma ratkaista suljetussa muodossa, kun mallit ovat LUVUNTODO tavoin 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 linearisoida. Näitä linearisoituja suodattimia ei käsitellä tässä

tutkielmassa. Hiukassuotimen tavoin hiukassiloitin ratkaisee ongelman mille hyvänsä epälineaariselle mallille.

3.2 Offline-algoritmit

Offline-siloittimet estimoivat siloitintiheyttä ajanhetkellä $k < T$, kun havaintodata on käytössä koko ajanjaksolta $1 \dots T$. Oheiset algoritmit siis olettavat, että kaikki mahdollinen tuleva data on jo niiden käytössä. Ohessa käsitellään lyhyesti muutamaa ehdottua offline-hiukassiloitinalgoritmia.

3.2.1 SIR-siloitin

SÄRKKÄ2013, KITAGAWA1996, DOUCET2000. Kuten aiemmin mainittua, näyttävät hiukassuodinalgoritmit, erityisesti SIR-algoritmi 3, ratkaisevan siloitteluongelman ilmaiseksi, kunhan tallennamme ajanhetkellä k koko otoshistorian $x^i_{0:k}$. Tällöin voimme estimoida täyttä siloitteluposteriorijakaumaa seuraavasti:

$$p(x_{0:T}|y_{1:T}) \approx \sum_{i=1}^{N} w^i_T \delta(x_{0:T} - x^i_{0:T}), \tag{3.3}$$

.

Nyt ajanhetken k siloitinjakauma saadaan laskettua

$$p(x_k|y_{1:T}) \approx \sum_{i=1}^{N} w^i_T \delta(x_k - x^i_k), \tag{3.4}$$

,

missä x^i_k on $x^i_{0:T}$:n k:s elementti. Koska uudelleenotanta hävittää otoshistorian, pitää uudelleenotanta suorittaa koko otoshistoriasta $x^i_{0:k} = (x^i_{0:k-1}, x^i_k)$ pelkän ajanhetken k otoksen x^i_k sijaan. Koska nyt koko otoshistoria pitää tallentaa, vaatii SIR-siloitin NkT muistia pelkän N sijaan. Vastaavasti myös uudelleenotannan aikakompleksisuus kasvaa. [17]

SIR-siloittimen suurin ongelma on kuitenkin sen tuottamien estimaattien hyvyys. Kun ajanhetkien määrä kasvaa, johtaa koko otoshistorian uudelleenotanta kaiken painon kasautumiseen historian tietyille otoksille, jolloin SIR-siloittimen tuottamat estimaatit eivät enää estimoi haluttua (siloittelu)posteriorijakaumaa. [17]

3.2.2 BS-PS-siloitin

Backward-simulation particle smoother (BS-PS) eli taaksepäin simuloiva hiukassiloitin estimoi paremmin hiukassuotimen tulosten perusteella siloitinjakaumaa. Tässä algoritmissa hiukkasten historia simuloidaan ajanhetkestä T taaksepäin ajanhetkeen 0:

Algoritmi 5: Taaksepäin simuloiva hiukassiloitin

Result: Posteriorisiloitinjakauman $p(x_k|y_{1:T})$ estimaatti.
Data: Suodinjakaumia edustavat hiukkaset ja näihin liittyvät painot w_k^i, x_k^i, missä $i = 1, \ldots, N$ ja $k = 1, \ldots, T$

begin
  begin
    Valitaan $\tilde{x}_T = x_T^i$
  for $k = \{T-1, \ldots, 0\}$ **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:

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

,

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

3.2.3 Uudelleenpainottava hiukassiloitin

Uudelleenpainottavassa hiukassiloittimessa (tunnetaan myös nimellä marginaalihiukassiloitin, kts. mm. Doucet, Godsill & ja Andrieu [3]) siloitinjakaumaa estimoidaan käyttämällä SIR-hiukassuodattmista (3) saatuja hiukkasia, mutta ne painotetaan uudelleen käyttäen dataa ajanhetkestä T alkaen, edeten ajassa taaksepäin.

Algoritmi 6: Uudelleenpainottava hiukassiloitin

Result: Posteriorisiloitinjakauman $p(x_k|y_{1:T})$ estimaatti.
Data: Suodinjakaumia edustavat hiukkaset ja näihin liittyvät painot w_k^i, x_k^i, missä $i = 1, \ldots, N$ ja $k = 1, \ldots, T$

begin
  begin
    Asetetaan $w_{T|T}^i = w_T^i$, jokaiselle $i = 1, \ldots, N$;
  for $k = \{T-1, \ldots, 0\}$ **do**
    begin
      Lasketaan uudet painot
      $w_{k|T}^i = \sum_j w_{k+1|T}^j \frac{w_k^i p(x_{k+1}^j|x_k^i)}{\sum_l w_k^l p(x_{k+1}^j|x_k^l)}$

,

jolloin halutun siloitinjakauma estimaatti ajanhetkellä k saadaan painotettuna keskiarvona $p(x_k|y_{1:T}) \approx \sum_i w^i_{k|t}\delta(x_k - x^i_k)$. Algoritmin aikakompleksisuus on $\mathcal{O}(N^2)$.

3.3 Online-algoritmit

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

3.3.1 Kiinteän viipeen siloitin

Yksinkertaisin tapa toteuttaa kiinteän viipeen siloitin on yksinkertaisesti käyttää SIR-siloitinta niin, että maksimiajanhetki T korvataan valitulla viipeellä $k + L \leq T$. [17]. Nyt yhtälön 3.3 jakauma saadaan muotoon

$$p(x_{0:(k+L)}|y_{1:(k+L)}) \approx \sum_{i=1}^{N} w^i_{k+L}\delta(x_{0:(k+L)} - x^i_{0:(k+L)}), \tag{3.6}$$

ja nykyisen ajanhatken k siloitinjakauma lasketaan tästä jakaumasta kuten SIR-siloittimessa (kts. yhtälö 3.4). Kiinteän viipeen siloitin myös välttää SIR-siloittimen approksimaatio-ongelmat. Kun viipeelle L pätee $k + L \ll T$ parantaa viipeen pidentäminen tiettyyn pisteeseen asti jakauman approksimaatiota. Kitagawa (1996) suosittelee 10–20 aika-askeleen viivettä ja esittää 50 aika-askelta viipeen ylärajaksi. [17]. Paremman estimaatin vastapainona pidemmän viipeen valinta lisää myös viivettä, joka dataa tuottavaan järjestelmään pitää lisätä. Siloittimien tulokset ovat saatavilla vasta L ajanhetken 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 prediktiivisenä siloittimena, jossa siloittelujakaumaa $p(x_{0:(k+1)}|y_{1:(k+1)}$ käytetään suodinjakauman $p(x_{1:(k)}|y_{1:k})$ laskennassa. [23] Ydinajatuksena on muokata SIR-algoritmia 3 niin, että ajanhetken k painoja w^i_k painotetaan edelleen seuraavan ajanhetkestä $k+1$ lasketuilla painoilla ja näin painottaa jo nykyhetkessä niitä hiukkasia, joiden uskottavuus on seuraavalla ajanhetkellä suurempi. Tämä prediktiivinen siloitin voidaan toteuttaa lisäämällä SIR-algoritmiin painotusvaiheen jälkeen seuraava ala-algoritmi:

Algoritmi 7: Prediktiivinen siloiti (viive=1)

Result: Prediktiivisellä 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 $i = \{1, 2, \ldots, N\}$ **do**
    begin
      Luodaan simuloidut hiukkaset $\tilde{x}_{k+1}^i$ ehdotusjakaumsta $q(\tilde{x}_{k+1}|x_k^i, y_{k+1})$
    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 prediktiivinen siloitin suodinjakauman laskemisen tarkkuutta. Vastaavasti prediktiivinen siloitin mahdollistaa saman suodinjakauman estimaatin tarkkuuden kuin SIR-algoritmi pienemmällä määrällä hiukkasia, kuitenkin vainb tuplaten uskottavuusfunktiota laskettaessa vaadittavan laskentatehon ja muistitarpeen.

3.3.2 Mukautuvan viipeen siloitin

Yllä esitetyssä kiinteän viipeen siloittimessa on valittu viive L suunnitteluparametri. Valittu viive on aina kompromissi: liian suuri viive kasvattaa siloitinjakauman estimoinnin 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 yrittävät ratkaista tämän ongelman mukauttamalla kunakin ajanhetkenä valittua viivettä johonkin kriteeriin perustuen. Erään version mukautuvan viipeen siloittimesta esittävät Johan Alenlöv ja Jimmy Olsson artikkelissa "Particle-Based Adaptive-Lag Online Marginal Smoothing in General State-Space Models" (2019) [16]. Siloitin hyödyntää hiukassuotimen varianssiestimaattia viipeen valinnassa.

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

$$\sigma_{s|t}^2 = \sum_{i=1}^{N} \frac{w_t^i}{\Omega_t} \left\{ \tilde{x}_{s|t} - \sum_{j=1}^{N} \frac{w_t^j}{\Omega_t} \tilde{x}_{s|t} \right\}^2, \tag{3.7}$$

pysyy tietyn valitun rajan ϵ yläpuolella, missä $\tilde{x}_{s|t}$ on kyseiselle viipeellä laskettu marginaalisiloitinjakauman painovektori (kts. algoritmi 6). Kun tämä ehto ei enää täyty, käytetään suurimmalle kriteerin $\sigma^2 s|t < \epsilon$ täyttämälle viipeelle laskettuja

painoja siloitinjakauman estimointiin kaikilla $t' \geq t$. Varianssin estimoinnista katso alaluku TODOLINKKI.

Algoritmi 8: Mukautuvan viipeen siloitin

Result: Siloittelujakauman estimaatti viipeellä s, tarkemmin $\sum_i^N w_t^i \tilde{x}_{s|t}^i \Omega_t$.

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

begin

  for $t = \{1, 2, \ldots, T\}$ **do**

    begin

      Ajetaan SIR-algoritmi 3 ajanhetkenä t

    begin

      Jokaiselle $s \in S$ lasketaan painovektori kuten algoritmissa 6.

    begin

      $S \leftarrow S \cup \{s\}$Jokaiselle $s \in S$ lasketaan painovektori kuten algoritmissa 6.

    begin

      Jokaiselle s lasketaan varianssi $\sigma_{s|t}^2$ kuten yhtälössä 3.7. Jos $\sigma_{s|t}^2 < \epsilon$ poistetaan s joukosta S ja käytetään siloitinjaukauman estimaattia $\sum_i^N w_t^i \tilde{x}_{s|t}^i \Omega_t$ kaikille ajanhetkille $t' \geq t$.

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

Luku 4

Hiukassuodin 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. Yleinen valinta ovat erilaiset Bluetooth-standardiin tai muuhun radioteknologiaan perustuvat lähetin-vastaanotinratkaisut.

Tässä luvussa TODO kuvaus.

4.1 Teknologian kuvaus

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

Walkbase on kehittänyt sisätilapaikannukseen oman laitteisto- ja ohjelmistoratkaisunsa, jonka tavoitteena on tarjota kaikissa ympäristöissä 95% varmuudella alle metrin paikannustarkkuus. Merkitään tätä paikannusvirhettä:

$$E_{\text{pos}} = \tag{4.1}$$

,

jolloin yhden testi. Paikannusvirheeseen palataan luvun tulososassa. Walkbasen paikannusratkaisu koostuu kolmesta eri laitteistokomponentista, AT-2-Bluetooth-lähetin-vastaanottomista, jotka kiinnitetään ostoskärryihin, XR-2-Bluetooth-lähetin-vastaanottomista, jotka kiinnitetään tilan kattoripustuksiin sekä OSCU-laskentayksiköstä, joka luo paikkadataa XR-2.1-vastaanotinten perusteella ja lähettää paikkadatan edelleen palvelinkeskukseen.

AT-2 on Bluetooth 5.1 (BLE) -strandardin mukaan toimiva lähetin-vastaanotin, joka toimii 2.4Ghz taajuusalueella. Walkbasen suunnitteleman laitteen PCB-

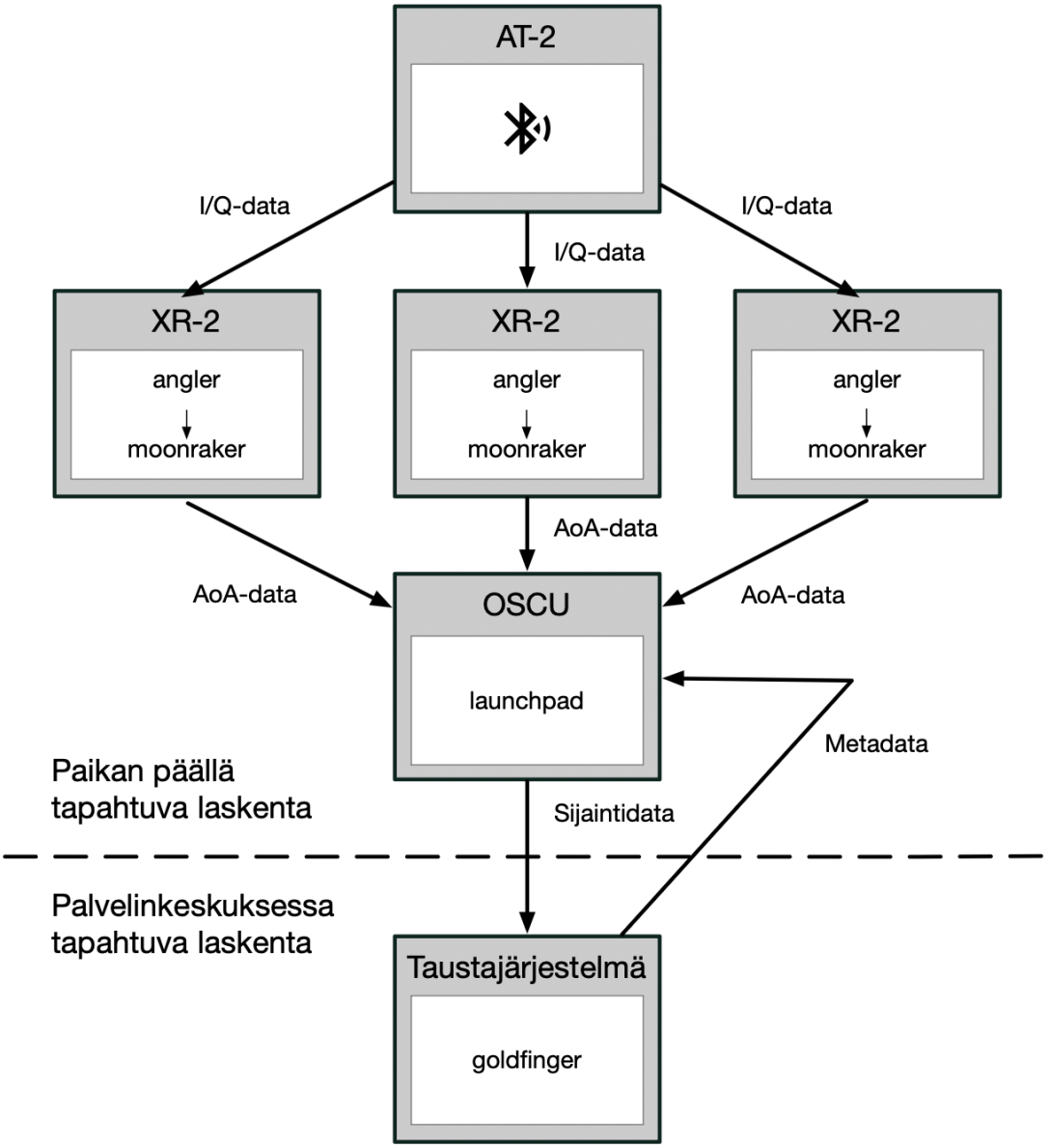
kehäantenni kykenee lähettämään GFSK-moduloitua dataa 2Mbps nopeudella. Laite saa virtansa yhdestä CR-2477-paristosta.

XR-2.1 on Bluetooth 5.1 (BLE) -strandardin mukaan toimiva lähetin-vastaanotin, joka toimii 2.4Ghz taajuusalueella. Walkbasen suunnitteleman laitteenPCB-kehäantenni kykenee lähettämään 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 TODO. Koska OSCU-laskentayksikön laskentateho on rajallista, on paikannusalgoritmin aikakompleksisuus yksi käytettävän algoritmin ydinkriteereistä. Tähän palataan myöhemmin koeasetelman kuvauksessa TODO linkki.

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

Taustajärjestelmässä Go-ohjelmointikielellä *goldfinger* prosessoi sijaintidatan Walkbasen analytiikka-alustan käyttämään muotoon. *Goldfinger* myös vastaa siitä, että kaikki *launchpad*-sovelluksen vaatima metadata on sen käytössä. Kaavio 4.1 kuvaa järjestelmän laitteisto- ja ohjelmistoarkkitehtuurit.



Kuva 4.1: Järjestelmäarkkitehtuuri

Tässä luvussa keskitytään *launchpad*-sovelluksen hyödyntämään paikannusalgoritmiin, mutta sitä ennen käsitellään lyhyesti tulokulman laskentamenetelmät sekä *angler*-sovelluksen toiminta. Tekijänoikeussyistä tutkielmassa ei hyödynnetä Go-ohjelmointikielellä toteutettua ohjelmakoodia. Sen sijaan algoritmi on toteutettu R-ohjelmointikielellä.

4.1.1 AoA-menetelmistä

vaihe-eron havaitsemiseen.

MUSIC-algoritmi esitys seuraa Monson H. Hayesin kirjaa *Statistical Digital Signal Processing and Modeling* (1996) [10].

Gunia [18]

Lisäksi *angler*-sovellus hyödyntää niin ikään omaa hiukassuodinalgoritmiaan tulokulmien laskentaan. Tätä hiukassuodinalgoritmia ei käsitellä tämän tutkielman puitteissa.

4.1.2 Kalibraatioalgoritmi

Koska *angler*-sovellus laskee suuntimakulman aina antennielementin määrätystä reunasta nähden, on tärkeää, että laitteen asemointi karttapohjoiseen nähden on tiedossa. Koska laitteen tarkan kulman arvioiminen asennusympäristössä on haastavaa eikä laite sisällä kompassia, on alla esitetty SIR-hiukassuodinta hyödyntävä kalibraatioalgoritmi, joka ottaa huomioon myös laitteen intertiamittausyksiköstä saatavat kallistumis- ja nyökkäämiskulmat. Kalibraatio-ongelma on esitetty kaaviossa TODO.

Algoritmi 9: Kalibraatioalgoritmi

Result: Posteriorisiloitinjakauman $p(x_k|y_{1:T})$ estimaatti.
Data: Suodinjakaumia edustavat hiukkaset ja näihin liittyvät painot w_k^i, x_k^i, missä $i = 1, \ldots, N$ ja $k = 1, \ldots, T$

begin
  begin
    Asetetaan $w_{T|T}^i = w_T^i$, jokaiselle $i = 1, \ldots, N$;
  for $k = \{T-1, \ldots, 0\}$ **do**
    begin
      Lasketaan uudet painot
      $$w_{k|T}^i = \sum_j w_{k+1|T}^j \frac{w_k^i p(x_{k+1}^j|x_k^i)}{\sum_l w_k^l p(x_{k+1}^j|x_k^l)}$$

,

4.2 Datan kuvaus

Koeasetelmaa varten AT-2-tagi on asetettu lähettämään IQ-dataotoksia 20hz taajuudella. AZIMUTH AND ELEVATION AND THEIR DISTRIBUTIONS. Nämä on kuvattu tarkemmin osiossa TODO. Jokaisen XR-2-laitteen *angler*-sovellus laskee näiden perusteella sekvenssinumeron, jonka perusteella samasta AT-2-tagin lähettämästä IQ-dataoksesta lasketut usean eri vastaanottimen laskemat tulokulmat voidaan yhdistää samaan IQ-dataotokseen. XR-2-laitteen lähettämä tulokulmadata on kuvattu alla.

Muuttuja	Kuvaus	Esimerkkiarvo
id	havainnon yksilöivä tunniste	317,092
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 IQ-dataotokseen yhdistävä juokseva numerointi	2,066
azimuth_location	atsimuuttikulman θ jaukauman sijaintiparametri (radiaania)	0.39
azimuth_scale	atsimuuttikulman θ jaukauman skaalaparametri ($\frac{1}{\text{radiaania}^2}$)	80.98
elevation_location	korkeuskulman γ jaukauman sijaintiparametri (radiaania)	0.13
elevation_scale	korkeuskulman γ jaukauman skaalaparametri (radiaania^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. [6]

launchpad-sovelluksessa tulokulmadataan yhdistetään XR-2-laitteen MAC-osoitteen perusteella lisäksi tarvittavaa, XR-2-laitteita koskevaa metadataa. 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

Atsimuuttikulma ϕ 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 mittaamisen

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

$$\Phi = (\theta + \eta \times \frac{\pi}{180^\circ}) \mod 2\pi. \tag{4.2}$$

.

Suuntimakulma Φ kertoo vastaanottimen ja lähettimen välisen kulman. Lisäksi saatavilla on PostGIS-muotoon tallennettua polygonidataa, joka vastaa koeympäristön pohjapiirrustusta sekä koeympäristössä esiintyviä liikkumisen estäviä kohteita, kuten hyllyjä tai pöytiä. Näitä hyödynnetään sekä sijaintialgoritmin alustuksessa että karttasovitusalgoritmissa (kts. TODO)

Havaintomuuttujien ohella koetilanteesta on tallennettu testipolku, jota pitkin AT-2-tagia liikutetaan koetilanteessa. Testipolkudata pitää sisällään karttaan piirretyn janan pääte- 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

Testimuuttujia käytetään paikannusalgoritmien paikannusvirheen laskemisessa.

4.2.1 Karttaprojektioista

Kaikki yllä esitetyssä datassa esiintyvät sijaintikoordinaatit on tallennettu tietokantaan WGS 84 -tasokoordinaattijärjestelmässä. Koska hiukassuotimiin perustuvaa paikannusalgoritmia sovellettaessa on monin paikoin tarve syöttää parametereja 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 interpolaatiolla. Määritellään ensin kerrospolygonin rajausalue. Koska kaikki käytetyt koordinaatit ovat tämän rajausalueen sisällä, voidaan tätä rajausaluetta käyttää konversiossa metreiksi. Rajausalue koostuu neljästä kulmapisteestä. Poimitaan näistä pisteistä minimit ja maksimit sekä pituus- että leveyskoordinaateille. Näin saadaan neljä arvoa $B_{\text{lonlat}} = \{\text{lon}_{\text{min}}, \text{lon}_{\text{max}}, \text{lat}_{\text{min}}, \text{lat}_{\text{max}}\}$. Määritellään rajausalueen sivujen pituus metreinä geodeettisen etäisyyden avulla, jolloin saadaan kaksi metreissä laskettua etäisyyttä $D_{\text{m}} = d_{\text{lon}}, d_{\text{lat}}$. Käytetään näitä kulmapisteitä sekä metreinä laskettuja etäisyyksiä interpoloimaan koordinaatit

WGS 84 -koordinaattijärjestelmästä metreissä esitetyille arvoalueille $[0, \max(x_m)]$ ja $[0, \max(y_m)]$ seuraavasti:

$$x_m = f(x_{\text{lon}}; B_{\text{lonlat}}, D_m) = \frac{x_{\text{lon}} - \text{lon}_{\text{min}}}{\text{lon}_{\text{max}} - \text{lon}_{\text{min}}} \times d_{\text{lon}} \tag{4.3}$$

$$y_m = f(y_{\text{lat}}; B_{\text{lonlat}}, D_m) = \frac{y_{\text{lat}} - \text{lat}_{\text{min}}}{\text{lat}_{\text{max}} - \text{lat}_{\text{min}}} \times d_{\text{lat}} \tag{4.4}$$

,

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

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

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

.

4.2.2 Muunnettu data

Kun yllä esitetyt konversiot on. Käytetään datassa seuraavia. TÄHÄN NOTAATIOT.

4.3 Sisätilapaikannusalgoritmi

4.3.1 Ongelman kuvaus

Tarkoituksena on estimoida liikkuvan AT-2-tagin sijaintia. Merkitään tätä estimoitavaa tilasarjaa $x_{1:k} = \{x_1, \ldots, x_k\}$. 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 malli (4.7)

$$x_{k+1} = f(x_k, \nu_k) = x_k + D_k \begin{bmatrix} \cos\psi_k \\ \sin\psi_k \end{bmatrix} + \nu_k, \tag{4.7}$$

missä D_k on AT-2-tagin ajanhetkenä k kulkema matka ja ψ_k AT-2-tagin suuntimakulma kyseisenä ajanhetkenä. 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 $\text{id}(\cdot)$ on identiteettifunktio.

Vastaavasti $y_{1:k} = \{y_1, \ldots, y_k\}$ kuvaa AT-2-tagin ja XR-2-laitteiden välillä laskettuja kulmahavaintoja. Näin ollen jokainen havainto koostuu (maksimissaan) paikantimien määrää vastaavasta määrästä kulmia. Havainnot lasketaan sekunnin tarkkuudella, mutta todellinen havaintotarkkuus on tiheämpi.

Lisäksi tunnetaan sensoreihin $\{s^1, \ldots, s^4\}$ liittyvät pituus- ja leveyskoordinaatit (λ, ϕ), jotka on muutettu TODO mukaan metreiksi.

$$u = \begin{bmatrix} \lambda^1 & \phi^1 \\ \vdots & \vdots \\ \lambda^4 & \phi^4 \end{bmatrix}. \tag{4.8}$$

Määritellään havainnoille malli

$$y_k = h(x_k, u) + e_k = \text{atan2}(\begin{bmatrix} \phi^1 - x_k^y \\ \vdots \\ \phi^4 - x_k^y \end{bmatrix}, \begin{bmatrix} \lambda^1 - x_k^x \\ \vdots \\ \lambda^4 - x_k^x \end{bmatrix}) + e_k, \tag{4.9}$$

missä

$$\text{atan2}(y, x) = \begin{cases} \arctan(\frac{y}{x}) & \text{jos } \ > 0, \\ \arctan(\frac{y}{x}) + \pi & \text{jos } \ < 0 \text{ ja } y \geq 0, \\ \arctan(\frac{y}{x}) - \pi & \text{jos } \ > 0 \text{ ja } \ < 0, \\ +\frac{\pi}{2} & \text{jos } x = 0 \text{ ja } \ > 0, \\ -\frac{\pi}{2} & \text{jos } x = 0 \text{ ja } \ < 0, \\ \text{ei määritelty} & \text{jos } x = 0 \text{ ja } y = 0 \end{cases} \tag{4.10}$$

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

Kovarianssimatriisin estimaattina käytetään kunakin ajanhetkenä k antennikohtaisista havainnoista estimoituja otosvariansseja $\text{diag}(\hat{\sigma}_k^1, \ldots, \hat{\sigma}_k^4)^2 = \text{diag}(\frac{1}{n-1}\sum_{i=1}^n (s_i^1 - \bar{s})^2, \ldots, \sum_{i=1}^n (s_i^4 - \bar{s})^2)$. Määrittelemätön atan2-tapaus, jossa $x = 0$ ja $y = 0$ on käytetyllä mittaustarkkuudella käytännössä mahdoton. Jos tapaus halutaan välttää, voidaan nolla-arvot tarpeen vaatiessa korvata joillakin hyvin lähellä nollaa olevalla arvolla. Saadaan uskottavuusfunktioksi

$$p(y_k|x_k) \propto \prod_{j=1}^{4} \exp\left\{ -\frac{\left\| h(x_k^j, u) - y_k^j \right\|^2}{2(\hat{\sigma}_i^j)^2} \right\}, \tag{4.11}$$

missä $j = \{1, l \ldots, n\}$ vastaa nyt kutakin XR-2-laitetta. Kumpikaan funktiosta $h(\cdot)$ ja $f(\cdot)$ ei ole lineaarinen, joten SIR-algoritmi on sopiva valinta ongelman ratkaisemiseksi. Koetuloksia arvioidaan ensisijaisesti paikannusvirheen avulla. Paikannusvirhe e_k lasketaan jokaisen ajanhetken k posteriorijakaumaestimaatista $\hat{p}_k$ painotettuna keskiarvona

$$\epsilon_k = \sum_{i=1}^{N} w_i^k d(x_k^i, y_k), \tag{4.12}$$

.

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

4.3.2 Uskottavuusmallit

Jokainen yhtä tulokulmaa vastaava *angler*-sovelluksen tuottama havaintodatarivi pitää sisällään neljä parametrimuuttujaa, `azimuth_location`, `azimuth_scale`, `elevation_location` ja `elevation_scale`. Koska *angler*-sovellus on kirjoitettu varta vasten tuottamaan dataa hiukassuodinpaikannusalgoritmia varten, ovat nämä suoraan hiukassuotimen uskottavuusmallin parametreja.

Angler-sovelluksessa XR-2-laitteen ja AT-2-tagin välinen atsimuuttikulma simuloidaan von Mises -jakaumasta, jolloin muuttujat `azimuth_location` ja `azimuth_scale` vastaavat tämän jakauman sijainti- ja skaalaparametreja μ ja κ ja näistä edellistä voidaan pitää itse atsimuuttikulman estimaattina. Määritellään siis jokaiselle hiukkassuotimen aikahetkelle k sekä XR-2-laitteelle l seuraava atsimuuttikulman uskottavuusmalli:

$$L_{\theta_{k,l}}(y_{k,l}|x_{k,l}; \mu_{k,l}, \kappa_{k,l}) = \frac{e^{\kappa_{k,l}\cos(x_{k,l}-\mu_{k,l})}}{2\pi I_0(\kappa_{k,l})} \tag{4.13}$$

,

missä I_0 on 0:s ensimmäisen lajin Bessel-funktio ja $x_{k,l}$ on jokaisen hiukkasen $n = 1, \ldots, N$ 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 $b = 2\pi$. 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 hiukkassuotimen aikahetkelle k sekä XR-2-laitteelle l seuraava korkeuskulman uskottavuusmalli:

$$L_{\gamma_{k,l}}(y_{k,l}|x_{k,l}; \mu_{k,l}, \sigma_{k,l}^2, a = 0, b = 2\pi) = \frac{1}{\sqrt{2\pi}}\exp(-\frac{(x_{k,l} - \mu_{k,l})}{2\sigma^2})/\sqrt{\sigma^2}Z_{k,l} \tag{4.14}$$

,

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

Uskottavuusmallit 4.13 ja 4.14 kertomalla saadaan yhdistetty uskottavuusmalli hiukkassuotimen aikahetkelle k sekä XR-2-laitteelle l

$$L_{k,l} = L_{\theta_{k,l}} \times L_{\gamma_{k,l}} \tag{4.15}$$

,

josta voidaan edelleen laskea jokaisen hiukkasen uskottavuus ajanhetkenä k

$$L_k = \prod_i^L L_{k,i} \tag{4.16}$$

.

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

4.3.3 Datan valinta

In the positioning particle filter, convex hull method is used in angle selection. The basic premise of this method is simple. Before evaluating particle likelihoods an angle/locator selection is performed. As one step of this process, a convex hull is drawn around the particle cloud and only angles that intersect this polygon get included in the likelihood evaluation. The following illustration explains the method.

Here the angles from the locators #2 and #1 get included in the likelihood evaluation as they intersect the convex hull of the particle cloud whereas the angle from locator #3 is discarded. As the basic interpretation of the particle cloud is that the probability of the tag being within it is one, we can safely assume that the correct position of the tag is also within the convex hull enclosing the particle cloud. Thus this method works well for getting rid of poor angles and reflections.

The Problem However, the above method also leads to a problem: the smaller the particle cloud gets, the more precise the angles need to be to get included in the likelihood evaluation. When running the positioning particle filter at the frequency of one this is rarely a problem, as this frequency necessitates a noise q value of 1.5m thus leading to particle clouds averaging at least 3m in width. However if we want to move to a higher frequency to be able to reduce the noise value into centimeter range, this will become a problem.

The Solution The simplest solution to the above problem is to add padding to the convex hull. An example of this is pictured below.

In the R positioning particle filter this padding is achieved by introducing a new design parameter called ch_q_adjustment. Before the application of the movement model this value is added to the q noise value and the particles are moved using this adjusted q_new = q + ch_q_adjustment value. The resulting moved particle cloud is then used to create the adjusted convex hull after which these particles are discarded and the original particles are moved again using the plain q value. This method of achieving the convex hull padding is a bit involved and probably computationally more costly than just applying the movement model once and adjusting the size of the resulting convex hull using some other transformation. However this approach has the upside of retaining the original logic of convex hull creation and allowing to "decompose" the used noise. I.e. using a q value of 0.9 and ch_q_adjustment value of 0.52 leads to a similar sized convex hull as using the q value of 1.42.

4.3.4 Dynaaminen malli

Dynaamisen mallin tehtävänä on liikuttaa hiukkasia ajanhetkien k ja $k+1$ välillä. Dynaaminen malli perustuu vektorisuunnistusmalliin 4.7. Vastaavaa mallia hyödyntää muun muassa Solin (2016) [14]. Koska AT-2-tagi ei sisällä luotettavaa IMU-yksikköä, ei tagi kuitenkaan tuota nopeus- tai suuntadataa. Tästä syystä vektorisuunnistusmallin nopeutta ilmaiseva skalaarimuuttuja $D_k = 0$, jolloin malli yksinkertaistuu satunnaiskävelymalliksi

$$x_{k+1} = x_k + v_k \tag{4.17}$$

,

missä v_k on kohinaa. Kohina luodaan jokaiselle ajanhetkelle seuraavasti. Ensin luodaan etäisyysvektori d_k katkaistusta normaalijakaumasta, jonka sijaintiparametri on 0 ja missä keskihajonta q ja katkaisukohdat q_min sekä q_max ovat paikannusalgoritmin suunnitteluparametreja:

$$d_k \sim \mathcal{N}_{\text{katkaistu}}(\mu = 0, \sigma = q, a = q_{\min}, b = q_{\max}), \tag{4.18}$$

.

Tämän jälkeen luodaan suuntavektori p tasajakaumasta

$$p_k \sim \mathcal{U}(0, 2\pi) \tag{4.19}$$

,

ja lopulta kohinavektori

$$v_k = (\cos(p_k) \times d_k, \sin(p_k) \times d_k), \tag{4.20}$$

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

4.3.4.1 Paikallaanolon havaitseminen

Paristonsäästyösyistä AT-2-tagi lähettää dataa ainoastaan liikkuessaan. Jos laite ei ole 10 sekunnin aikana havainnut IMU-yksikön perusteella kiihtyvyyttä, lopettaa laite datan lähettämisen. Laite on tällöin valmiusmoodissa. Tätä tietoa voidaan hyödyntää poistamalla dynaamisesta mallista kohina, kun laitteen tiedetään olevan paikallaan. Koska hiukkassilottimen (kts. kohta TODO) vuoksi tarvitsemme paikannusalgoritmiin viipeen, voimme hyödyntää tätä viivettä myös paikallaolon tehokkaampaan havaitsemiseen.

Jos yksikään XR-2-laite ei ole havainnut tagia yhdenkään sekuntiin kuuluvan 20 aika-askeleen aikana, voimme olettaa laitteen olevan valmiusmoodissa ja siten myös

paikoillaan. Huomattavaa on kuitenkin, että päinvastainen ei päde. Paikoillaan oleva laite ei välttämättä ole valmisumoodissa, sillä valmiusmoodiin siirtyminen kestää 10 sekuntia. Tähän oletukseen perustuen voidaan havaita paikallaolon 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 moneksi aikahetkeksi dynaamista mallia tulee vaimentaa. Jos $m = 0$ mallia ei vaimenneta.;

Data: Tagin tulokulmadata $n + 1$ aika-askeleelle (nykyinen aika-askel + n aika-askelta tulevaisuuteen). n 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, . . . , k + n} do
        begin
            Merkitään o_l kulmahavaintojen määrää ajanhetkenä 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 ttä aika-askelta. Jos algoritmi löytää useita peräkkäisiä aika-askeleita, joiden aikana ei ole tallennettu lainkaan kulmahavaintoija, valitsee sen näistä suurimman. Koska koe-esimerkissä AT-2-laite lähettää dataa 20hz taajuudella ja valmiustila aktivoituu 10 sekunnin kohdalla, valitaan alla $n = 200$.

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

4.3.4.2 Karttasovitusalgoritmi

Dynaamista mallia sovellettaessa, voimme myös höydyntää saatavilla olevaa sisätilan karttadataa. Määritellään kaksi polygonityyppiä, lattiapolygoni F sekä estepolygonit $E = \{E_1, \ldots, E_n\}$. Lattiapolygonit määrittävät alueen, jonka sisällä hiukassuodinalgoritmien hiukkasten pitää pysyä. Lattiapolygoneja on vain yksi. Vastaavasti estepolygonien joukko määrittää lattiapolygonien sisällä alueet, joiden sisälle hiukassuodinalgoritmin hiukkaset eivät voi siirtyä ja joiden läpi tagi ei voi kulkea. Käytännössä estepolygonit kuvaavat tilan seiniä sekä tiedossa olevia esteitä, kuten hyllyjä, pöytiä ja niin edelleen.

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

$$w_{k+1}^i = \begin{cases} w_{k+1}^i & \text{jos } x_{k+1}^i \in F, \\ 0 & \text{jos } x_{k+1}^i \notin F \vee x_{k+1}^i \in E \end{cases} \tag{4.21}$$

,

jonka 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:

$$w_{k+1}^i = \begin{cases} P \times w_{k+1}^i & \text{jos } x_{k+1}^i \text{ylittää estepolygonin}, \\ w_{k+1}^i & \text{muulloin} \end{cases} \tag{4.22}$$

,

jossa rangaistus P on algoritmin suunnitteluparametri. Vastaavaa toteutusta ovat hyödyntäneet muun muassa Davidson &al (2010) [12], jotka ehdottavat rangaistusarvoa $\frac{1}{1000}$. Rangaistuksen valintaan palataan luvussa parametrien valinta TODO LINKKI.

Koska erityisesti 4.21 asettaa hiukkasten painoja nollaan, eivät karttasovitetut hiukkaset välttämättä enää estimoi suodinjakaumaa tehokkaasti. Bojja &al [13] ehdottavat suoritettavaksi uudelleenotantaa karttasovituksen jälkeen. Uudelleenotannalla korvataan nollapainoiset hiukkaset hiukkasilla, joiden paino ei ole nolla:

$$\text{atan2}(y, x) = \begin{cases} \arctan(\frac{y}{x}) & \text{jos } \ > 0, \\ \arctan(\frac{y}{x}) + \pi & \text{jos } \ < 0 \text{ ja } y \geq 0, \\ \arctan(\frac{y}{x}) - \pi & \text{jos } \ > 0 \text{ ja } \ < 0, \\ +\frac{\pi}{2} & \text{jos } x = 0 \text{ ja } \ > 0, \\ -\frac{\pi}{2} & \text{jos } x = 0 \text{ ja } \ < 0, \\ \text{ei määritelty} & \text{jos } x = 0 \text{ ja } y = 0 \end{cases} \tag{4.23}$$

.

4.3.4.3 Reitinhakualgoritmi

- Create separate grids for raw positions and Kalman filtered positions with separate and adjustable grid cell widths.
- After positioning estimates and their Kalman filtered counterparts are created, snap these to the grid. Note! By default particles are not snapped to the grid, only the position estimates.
- See if the previous position is within speed limit / Kalman filter speed limit (these are separate arguments) distance. These are set to 3m/s by default.
- If the speed limit is not exceeded, create a new grid-snapped position. Also create a position to all the grid cells between the newly created position and the previous position. This path is found using A* algorithm.
- If the speed limit is exceeded, wait for new raw / Kalman filtered positions to come in until the speed limit is no longer exceeded, in which case proceed as in the previous step.
- Note! Pathfinding positions are not used anywhere else in the R positioning algorithm. Thus both the pathfinding positions and their Kalman filtered counterpresents create an entirely independent layer of positions on top of raw / Kalman filtered positions.
- Noteˆ2! When doing the A* pathfinding the time between positions is split equally so that the interpolated positions get a fractional time step between the start and stop positions of the pathfinding. Thus the positioning frequency of 1 is not respected.

4.3.5 Siloittelualgoritmit

Hyödynnetään alaluvussa TODO esitetty prediktiivistä siloitinta. Satunnaiskulkumalli ei ole optimaalinen, mutta ...

4.3.6 WB-sisätilapaikannusalgoritmi

Alla esitetään SIR-algoritmiin ja yllä esitettyihin algoritmeihin perustuva sisätilapaikannusalgoritmi kokonaisuudessaan.

Algoritmi 11: Uudelleenpainottava hiukassiloitin

Result: Posteriorisiloitinjakauman $p(x_k|y_{1:T})$ estimaatti.
Data: Suodinjakaumia edustavat hiukkaset ja näihin liittyvät painot w_k^i, x_k^i, missä $i = 1, \ldots, N$ ja $k = 1, \ldots, T$

begin
  begin
    Asetetaan $w_{T|T}^i = w_T^i$, jokaiselle $i = 1, \ldots, N$;
  for $k = \{T-1, \ldots, 0\}$ **do**
    begin
      Lasketaan uudet painot
      $$w_{k|T}^i = \sum_j w_{k+1|T}^j \frac{w_k^i p(x_{k+1}^j | x_k^i)}{\sum_l w_k^l p(x_{k+1}^j | x_k^l)}$$

,
algoritmin ominaisuuksista.

Koeasetelmassa käytetty SMC-algoritmi on toteuttu R-kielellä. Algoritmin toteutus on pääosin vektorisoitu ja tehokas. For-silmukkaa on käytetty ainoastaan ajanhetkien läpikäyntiin. Koska tämän silmukan muuntaminen vektorisoituun muotoon ei ole mahdollista, voidaan toteutusta pitää näiltä osin hyvin optimoituna. Algoritmin datan käsittely on toteutettu täysin suorituskyvyltään erinomaisella `data.table`-kirjastolla. Koodin profilointi XXX.

Koska algoritmin uskottavuudet sekä painot ovat hyvin pieniä, on laskentatarkkuusongelmien välttämiseksi toteutuksessa käytetty logaritmoituja painoja ja uskottavuusfunktiota. Tämä ei vaikuta lainkaan itse algoritmin toimintaan, mutta estää numeeristen ongelmien syntymisen. Tilanteessa, jossa tietyn paikantimen ja kaikkien partikkelien välinen uskottavuus on nolla, päätyvät kaikki painot nolliksi eikä algoritmi enää toimi. Tällaisessa tilanteessa uskottavuusfunktion R-toteutus kutsuu itseään rekursiivisesti uudelleen niin, että kyseinen paikannin on tiputettu havainnoista.

Paikannusvirheen laskemisessa on etäisyysfunktiona $d(\cdot)$ käytetty `raster`-kirjaston `pointDistance()`-funktiota. Koodissa on korostettu tiiviyden sijaan luettavuutta ja koodi on kommentoitu kattavasti. SMC-algoritmifunktion sekä uskottavuusfunktion R-koodit löytyvät osoitteesta https://github.com/rintakumpu/luk, mistä löytyy myös asetelmassa käytetty data csv-muodossa sekä kokeen muun koodin sisältävä R Markdown -notebook.

4.4 Empiirinen esimerkki

4.4.1 Koeasetelma

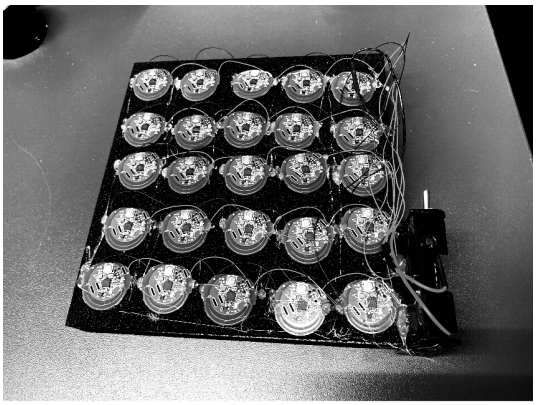
Koetta varten käytettiin yhden minuutin aikana kertyneitä havaintoja. Havaintoja on datassa yhteensä $N_{obs} = 15018$ kappaletta. Havaintojen aikaleimat on tallennettu sekunnin tuhannesosan tarkkuudella.

Esimerkissä käyteteään SMC-algoritmia Bluetooth-paikannussovelluksessa lähettimen sijainnin laskemiseen. Paikannukseen käytettävä data kerättiin toimistoympäristössä Bluetooth Low Energy (BLE) -lähettimen sekä kattoon sijoitettujen vastaanottimien avulla. Havainnot koostuvat vastaanottimien lähettimien signaalien perusteella laskemista, BLE5.1-standardin mukaisista signaalin tulokulmista eli AoA-havainnoista (angle of arrival). Lopuksi esimerkissä analysoidaan ja vertaillaan algoritmin eri versioiden suorituskykyä sekä suorituskyvyn että paikannustarkkuuden näkökulmasta. Vertailuarvona käytetään perinteistä triangulaatio-algoritmia.

Paikaunnusesimerkissä lähettimenä toimi 25 Bluetooth-paikannustagista koostuva Walkbase Foculator -testilaite (kuva 4.2), vastaanottimena toimistoympäristöön asennetut neljä Walkbase XR-2 -vastaanotinta (kuva 4.3). Jokainen vastaanotin sisältää kuusitoista antennia, joiden vastaanottamien lähetinsignaalien perusteella vastaanottimet laskevat signaalin tulokulman suhteessa vastaanottimeen. Tarkka tulokulmien laskemiseen käytetty algoritmi on paikantimen antennit toimittaneen Silicon Laboratories, Inc. -yrityksen liikesalaisuus, mutta perusperiaate on arvioida

tulokulma mittaamalla eri antenneiden välistä vaihe-eroa ns. IQ-signaalin avulla.

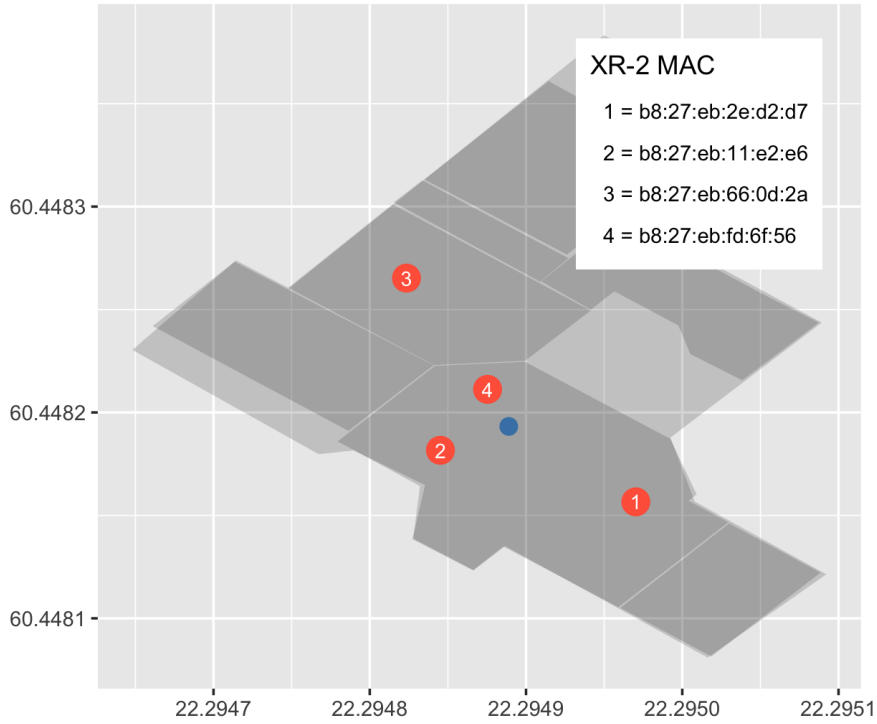
Esteettömässä ympäristössä koeasetelmassa käytetyn järjestelmän kulmavirhe on hyvin pieni ja paikannusongelma voidaan ratkaista riittävällä tarkkuudella suoraan triangulaatio-algoritmilla. Tässäkin tilanteessa voi paikannusta parantaa suodattimen käytöllä, mutta jo triangulaatio-algoritmin perusversio tuottaa halutun tarkkuuden. Toimistoympäristö on kuitenkin haastava, sillä erityisesti näyttöruudut sekä heijastavat että estävät radiosignaaleja. Silicon Labs lupaa omalle AoA-järjestelmälleen vastaavassa toimisympäristössä (seitsemällä paikantimella) kulmavirheen välillä $3.7^{\circ} - 5.7^{\circ}$. Tämä ei kuitenkaan riitä johdonmukaisesti haluttuun alle metrin paikannustarkkuuteen, joten AoA-paikannus toimistoympäristössä tarjoaa hyvän motivaation SMC-menetelmien käytölle.



Kuva 4.2: Walkbase Foculator

Kuva 4.3: Walkbase XR-2

Data on kuvattu tarkemmin alaluvussa 5.2. Koeympäristön pohjapiirustus on esitetty kuvassa 4.4. Piirustuksessa XR-2-paikantimet on kuvattu punaisilla numeroiduilla ympyröillä. Foculator-testilaitteen sijainti on kuvattu sinisellä ympyrällä. Toimistoympäristön pohjapiirustus on kuvattu harmaalla niin, että piirustuksesta erottuvat toimiston väliseinät.



Kuva 4.4: Koeasetelman pohjapiirustus

Toimisto valittiin testiympäristöksi, koska siellä testaaminen on helpompaa ja halvempaa kuin aidossa kauppaympäristössä. Lisäksi toimistossa hyvin toimiva järjestelmä todennäköisesti toimii sellaisenaan vähemmän esteisessä kauppaympäristössä.

Dataa kerättiin sekä liikkuvalla että paikallaan olevalla testilaitteella. Paikallaan oleva testilaite asetettiin pöydälle 1.5 metrin korkeudelle. Liikkuvassa tapauksessa testilaite kiinnitettiin robottiin, joka oli ohjelmoitu seuraamaan haluttua liikerataa vastaavia lattiamerkintöjä ennalta määrätyllä vakionopeudella. Yksinkertaisuuden vuoksi tässä tutkielmassa tarkastellaan ainoastaan ongelmaa, jossa testilaite on paikoillaan. Esitetyt mallit soveltuvat kuitenkin myös liikkuvalle laitteelle. Data kerättiin yöaikaan, jolloin toimiston käyttöaste oli alhainen. Tällä minimoitiin radiosignaalien tielle osuvien ihmisten vaikutus signaaleihin. Maan pinnan kaarevuus on otettu huomioon koeasetelmassa pisteiden välisiä etäisyyksiä laskettaessa.

4.4.2 Parametrien valinta

Priorijakaumana p_{x_0} käytettiin kahta toisistaan riippumatonta otosta tasajakaumista, joista toinen vastasi leveys- ja toinen pituusasteita. Jakaumien alkupisteet valittiin niin, että ne vastasivat pienimpiä paikantimien leveys- ja pituusasteista. Vastaavasti päätepisteet valittiin niin, että ne vastasivat suurimpia paikantimien leveys- ja pituusasteita.

$$p_{x_{0_{\text{lon}}}} \sim \mathcal{U}(\min \lambda, \max \lambda), \tag{4.24}$$
$$p_{x_{0_{\text{lat}}}} \sim \mathcal{U}(\min \phi, \max \phi). \tag{4.25}$$

Koska järjestelmän on tarkoitus toimia ainoastaan paikantimien muodostaman suorakaiteen sisäpuolella, ovat valitut jakaumien päätepisteet riittävät. Kummastakin jakaumasta otettiin $\sqrt{N}$ otosta, jolloin N partikkelia x_0^i saatiin näiden otosten permutaatioina.

Paikannus suoritettiin yhteensä 17 kertaa. Ensimmäisessä vaiheessa tarkasteltiin partikkelien määrän N vaikutusta paikannuskeskivirheeseen ilman uudelleenotantaa sekä priori- että uskottavuusotannalla. Seuraavassa vaiheessa valittiin ehdotusjakauma edellisen vaiheen tulosten perusteella ja tarkasteltiin tilannetta sekä adaptiivisella että jokaisella iteraatiolla suoritettavalla uudelleenotannalla.

Kaikkien tulosten vertailukohtana käytettiin Pierlot &al. artikkelissa "A New Three Object Triangulation Algorithm Based on the Power Center of Three Circles" (2011) esittämää ToTal-triangulaatioalgoritmia. [25] Triangulaatio-algoritmia ei käsitellä tässä tarkemmin, mutta se on esitetty algoritmissa 12. Algoritmia varten valittiin kunakin ajanhetkenä k ne kolme paikanninta ja kulmahavaintoa, joiden SNR-arvo oli korkein.

Algoritmit ajettiin RStudion versiossa 1.4.1106 R-ohjelmointikielen versiolla 4.0.4. Tietokoneena käytettiin vuoden 2017 mallia olevaa MacBook Pro -kannettavaa, jossa oli 3.1 Ghz Quad-Core i7 -prosessori sekä 16 gigatavua 2133 Mhz LPDDR3 -muistia. Suoritusnopeuden mittaamiseen käytettiin `microbenchmark`-kirjastoa. Jokainen algoritmi toistettiin kymmenen kertaa ja suoritusnopeus laskettiin näiden toistojen aritmeettisena keskiarvona.

Algoritmi 12: ToTal (Three object Triangulation algorithm)

Result: Testilaitteen sijaintiestimaatti (x_R, y_R).
Data: Kolmen paikantimen 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})
$x'_{12} = x'_1 + T_{12}y'_1, \quad y'_{12} = y'_1 - T_{12}x'_1$
$x'_{23} = x'_3 - T_{23}y'_3, \quad y'_{23} = y'_3 + T_{23}x'_3$
$x'_{31} = (x'_3 + x'_1) + T_{31}(y'_3 - y'_1), \quad y'_{31} = (y'_3 + y'_1) - T_{31}(x'_3 - x'_1).$

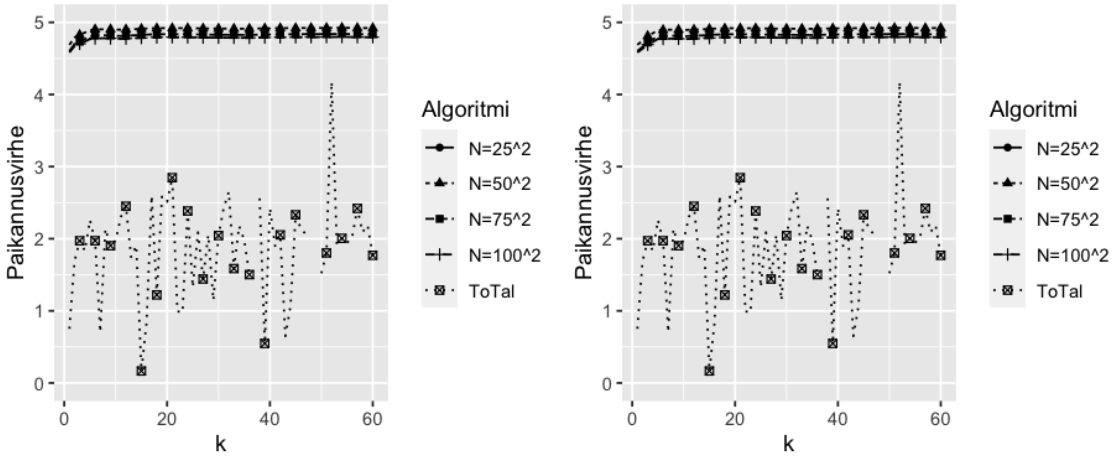
begin
Lasketaan $k'_{31} = x'_1x'_3 + y'_1y'_3 + T_{31}(x'_1y'_3 - x'_3y'_1)$.

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

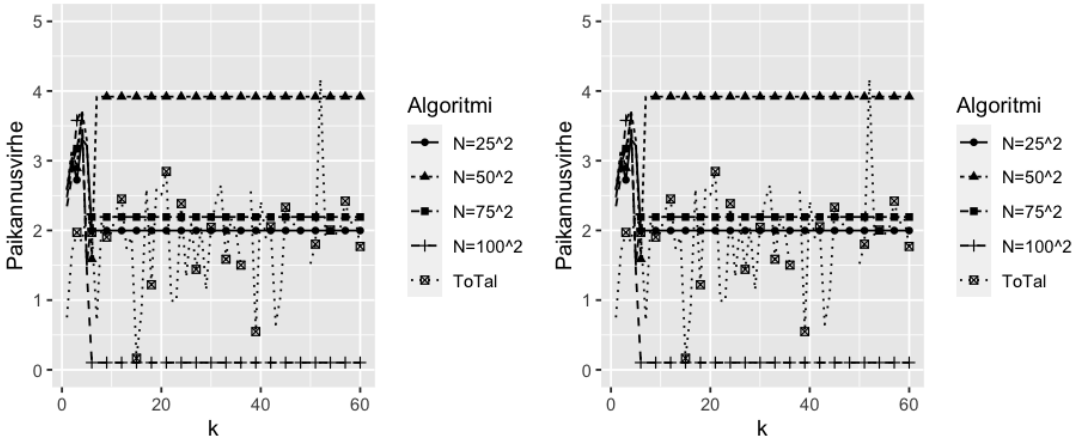
4.4.3 Tulokset

Ensimmäisessä vaiheessa paikannus suoritettiin ilman uudelleenotantaa sekä prioriettä uskottavuusotannalla (SIS) partikkelien määrän ollessa $N = \{25^2, 50^2, 75^2, 100^2\}$. Tulokset on esitetty kuvassa 4.5 sekä osana taulukkoa 4.4. Kuten tuloksista huomataan, ei ehdotusjakauman valinnalla ole juurikaan vaikutusta algoritmin toimivuuteen paikannusvirheen suhteen. Partikkelien määrän kasvattaminen ei myöskään automaattisesti paranna paikannustarkkuuta. Tämä viittaa siihen, että koeasetelma on herkkä priorijakauman valinnalle. Tuloksista huomataan lisäksi, että algoritmin aikakompleksisuus on luokkaa $\mathcal{O}(N)$, kuten tukielman teoriaosassa todettiin.



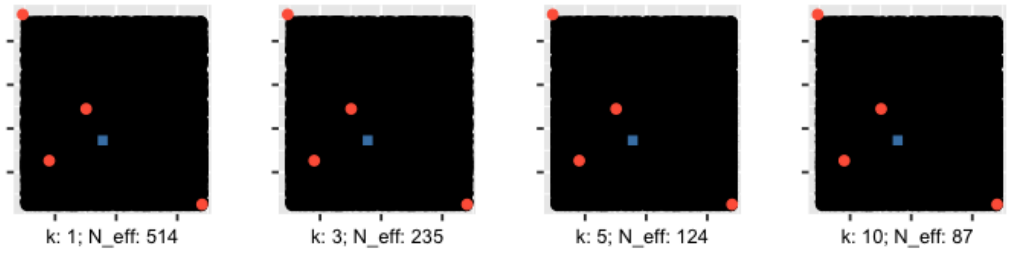
Kuva 4.5: Priori- ja uskottavuusotanta. Ei uudelleenotantaa.

Koska ehdotusjakaumalla ei juurikaan ollut vaikutusta tuloksiin, valittiin seuraavaan vaiheeseen muodoltaan yksinkertaisempi uskottavuusotanta. Uskottavuusotantaa käyttäen vertailtiin nyt joka iteraatiolla suoritettavaa uudelleenotantaa (bootstrap) sekä adaptiivista uudelleenotantaa, joka suoritettiin aina, kun effektiivinen otoskoko $N_{eff} < 2N/3$. Tulokset on esitetty kuvassa 4.6 sekä osana taulukkoa 4.4. Uudeeleenotantamenetelmän valinta ei vaikuta tuloksiin paikannusvirheen suhteen, mutta kuten oletettua, on adaptiivinen uudelleenotanta nopeampi. Nyt myös partikkelien määrän lisääminen parantaa selkeämmin paikkannustarkkuutta ja $N = 100^2$ partikkelilla saavutetaan jo haluttu alle metrin paikannustarkkuus.

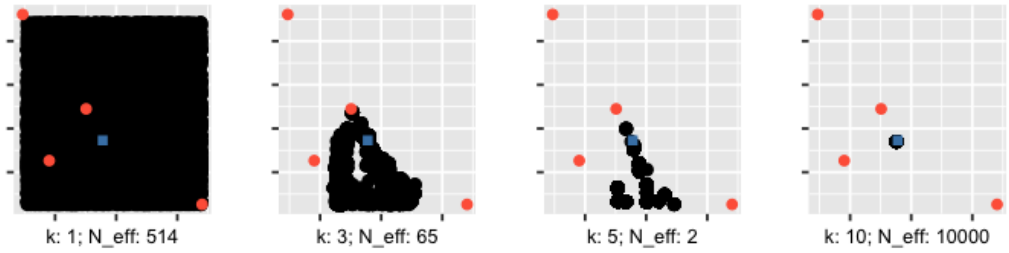


Kuva 4.6: Jokaisen iteraation uudelleenotanta / adaptiivinen uudellenotanta.

Kuvissa 4.7 ja 4.8 esitetään partikkelien jakauma sijainnin suhteen kartalla ajanhetkinä $k = \{1, 3, 5, 10\}$. Punaiset ympyrät kuvaavat vastaanottimia, sininen neliö testilaitteen todellista sijaintia ja mustat ympyrät partikkelien sijainteja. Molemmat kuvat esittävät uskottaavusotannalla saatuja tuloksia, kun $N = 100^2$. Edellisessä kuvassa 4.7 ei ole käytetty uudelleenotantaa. Vaikka partikkelien painoja ei kuvassa olekaan otettu huomioon, on kuvasta helppo huomata otoskoon ehtymisen ongelma. Jälkimmäisessä kuvassa 4.8 on käytetty adaptiivista uudelleenotantaa, joka on suoritettu aika-askelilla 1–6. Tämä ratkaisee otoskoon ehtymisen ja aika-askeleella 10 kaikki partikkelit approksimoivat haluttua sijaintia.



Kuva 4.7: Partikkelikartta. Ei uudelleenotantaa.



Kuva 4.8: Partikkelikartta. Adaptiivinen uudelleenotanta.

Algoritmi	N	Paikannusvirheen ka. (m)	Keston ka. (s)
ToTal	-	2.15	0.001
SIS/Priori	25^2	5.31	7.91
SIS/Priori	50^2	4.91	19.81
SIS/Priori	75^2	4.82	46.02
SIS/Priori	100^2	4.78	81.83
SIS/Uskottavuus	25^2	5.31	7.42
SIS/Uskottavuus	50^2	4.90	19.85
SIS/Uskottavuus	75^2	4.82	45.94
SIS/Uskottavuus	100^2	4.78	82.02
Bootstrap/Uskottavuus	25^2	2.08	8.64
Bootstrap/Uskottavuus	50^2	3.81	21.55
Bootstrap/Uskottavuus	75^2	2.24	45.88
Bootstrap/Uskottavuus	100^2	0.33	82.24
SIR/Uskottavuus	25^2	2.08	7.30
SIR/Uskottavuus	50^2	3.81	18.59
SIR/Uskottavuus	75^2	2.24	42.11
SIR/Uskottavuus	100^2	0.33	79.64

Taulukko 4.4: Paikannusvirheet sekä suoritusajat

Tuloksista huomataan, että uskottavuusotannalla ja uudelleenotannalla SMC-menetelmät tuottavat esitetyssä koeasetelmassa halutun tarkkuuden. Tarvittavalla määrällä partikkeleita tässä käytetyt SMC-algoritmit ovat kuitenkin aivan liian hitaita tuotantokäyttöön. Vaikka algoritmit pystyvät tuottamaan hitaimmillaankin noin sijainnin sekunnissa, pitää käytännön toteutuksessa sijainteja laskea joka sekunti sadoista paikantimista. Todennäköisesti yksinkertaisin tapa saavuttaa haluttu alle metrin paikannustarkkuus nopeammalla laskennalla on lisätä triangulaatiopaikannukseen jokin laskennallisesti kevyempi suodin, esimerkiksi EKF-suodin.

Koeasetelma toimi kuitenkin hyvänä konseptin todennuksena, sillä käytetyissä SMC-toteutuksissa on useita parannuskohteita. Ensinnäkin algoritmi voidaan toteuttaa täysin `data.table`-kirjaston avulla tai kokonaan R-ohjelmointikieltä tehokkaammilla ohjelmointikielillä. Toiseksi partikkelien määrää saattaa olla mahdollista laskea paikannustarkkuutta menettämättä käyttämällä parempia malleja. Koeasetelmassa käytetty havaintomalli (4.9) on erityisesti kohinan osalta hyvin *ad hoc* -tyyppinen malli. Mallia on mahdollista parantaa esimerkiksi lisämittauksista kerätyn vastaanotindatan avulla. Lisäksi tilamallia (4.7) pitää testata liikkuvalla lähettimellä, jotta eri paikannusalgoritmien erot tulevat näkyviin myös paremmin todellista käyttötilannetta vastaavassa koeasetelmassa.

Luku 5

Lopuksi

Tässä tutkielmassa on esitetty pääpiirteittäin SMC-menetelmien teoria Bayesilaisessa tilastotieteellisessä viitekehyksessä. Lisäksi tutkielmassa on käyty läpi uudelleenotantaa efektiivisen otoskoon perusteella hyödyntävä SIR-suodinalgoritmi. Lopuksi tutkielmassa on tarkasteltu SIR-algoritmin parametrien valintaan, suorituskykyyn sekä konvergenssiin liittyviä tuloksia.

Tutkielmassa on lisäksi tarkasteltu miten eri valinnat vaikuttavat algoritmin suorituskykyyn yksinkertaisen mutta kattavan ja todelliseen ongelmaan sekä dataan perustuvan paikannusesimerkin avulla.

Luku 6

Liitteet(`followed by`# A chapter‘)

Konvergenssituloksen **??** nojalla kyseinen varianssi $\sigma^2 \to \lim 0$, kun $N \to \inf$.

Lähteet

[1] Jimmy Olsson Alessandro Mastrototaro. Adaptive online variance estimation in particle filters: the alvar estimator. *Statistics and Computing*, 33(77), 2022.

[2] Nick Whiteley Anthony Lee. Variance estimation in the particle filter. *Biometrika*, 105(3):609–625, 2018.

[3] Christophe Andrieu Arnaud Doucet, Simon Godsill. On sequential monte carlo sampling methods for bayesian filtering. *Statistics and Computing*, 10(3):197–208, 2020.

[4] 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.

[5] Dan Crisan. The stochastic filtering problem: A brief historical account. *Journal of Applied Probability*, 51A:13–22, 2014.

[6] C. Vargas D. Munoz, R. Enriquez-Caldera. *Position Location Techniques and Applications*. Elsevier, 2009.

[7] Arnaud Doucet Dan Crisan. Convergence of sequential monte carlo methods. URL https://www.stats.ox.ac.uk/~doucet/crisain_doucet_convergenceofSMC2000.pdf.

[8] Arnaud Doucet Dan Crisan. A survey of convergence results on particle filtering methods for practitioners. *IEEE Transactions on Signal Processing*, 50(3): 736–746, 2002.

[9] Fredrik Gustafsson. Particle filter theory and practice with positioning applications. *IEEE Aerospace and Electronic Systems Magazine*, 25(7):53–82, 2010.

[10] Monson H. Hayes. *Statistical Digital Signal Processing and Modeling*. John Wiley & Sons, Inc., 1996.

[11] R. Chen J. Liu. Sequential monte carlo methods for dynamic systems. *Journal of the American Statistical Association*, 93(443):1032–1044, 1998.

[12] Pavel Davidsonl ja Jussi Collin ja Jarmo Takala. Application of particle filters for indoor positioning using floor plans. pages 1–4, 2010. doi: 10.1109/UPINLBS.2010.5653830.

[13] Jayaprasad Bojja Ja Jussi Collin ja Simo Särkkä ja Jarmo Takala. Pedestrian

localization in moving platforms using dead reckoning, particle filtering and map matching. pages 1116–1120, 2015. doi: 10.1109/ICASSP.2015.7178143.

[14] Arno Solin ja Simo Särkkä ja Juho Kannala ja Esa Rahtu. Terrain navigation in the magnetic landscape: Particle filtering for indoor positioning. *2016 European Navigation Conference (ENC)*, pages 1–9, 2016. URL https://api.semanticscholar.org/CorpusID:23211261.

[15] Randal Douc Jimmy Olsson. Numerically stable online estimation of variance in particle filters. *Bernoulli*, 25(2):1504–1535, 2019.

[16] Jimmy Olsson Johan Alenlöv. Particle-based adaptive-lag online marginal smoothing in general state-space models. *IEEE Transactions on Signal Processing*, 67 (21), 2019.

[17] 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.

[18] Niko Joram Marco Gunia, Adrian Zinke. Analysis and design of a music-based angle of arrival positioning system. *ACM Transactions on Sensor Networks*, 19 (3):66, 1–41.

[19] Angus P. Andrews Mohinder S. Grewal. Applications of kalman filtering in aerospace 1960 to the present. *IEEE Control Systems Magazine*, 30(3):69–78, 2010.

[20] Pierre Del Moral. Nonlinear filtering: Interacting particle resolution. *Markov Processes and Related Fields*, 2(4):555–580.

[21] A.F.M Smith N.J. Gordon, D.H. Salmond. Novel approach to nonlinear/non-gaussian bayesian state estimation. *IEE Proceedings F (Radar Signal Process)*, 140(2):107–113, 1993.

[22] Eric Moulines Olivier Cappé, Simon J. Godsill. An overview of existing methods and recent advances in sequential monte carlo. *Proceedings of the IEEE*, 95(5): 899–924, 2007.

[23] Serge Moto Samuel Nyobe, Fabien F. Campillo. 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.

[24] Simo Särkkä. *Bayesian Filtering and Smoothing*. Cambridge University Press, 2013. URL https://users.aalto.fi/~ssarkka/pub/cup_book_online_20131111.pdf.

[25] M. Van Droogenbroeck V. Pierlot, M. Urbin-Choffray. A new three object triangulation algorithm based on the power center of three circles. *Research and Education in Robotics - EUROBOT 2011*, page 248–262.