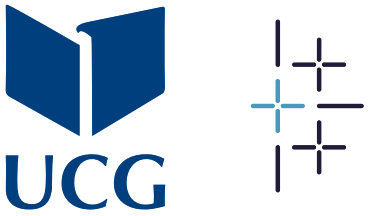


UNIVERZITET CRNE GORE
ELEKTROTEHNIČKI FAKULTET

Miljan Golubović

UPOREDNA ANALIZA ALGORITAMA ZA SIMULTANU LOKALIZACIJU I MAPIRANJE U REALNOM VREMENU

– MASTER RAD –

Podgorica, 2025.

PODACI I INFORMACIJE O STUDENTU

Ime i prezime:	Miljan Golubović
Datum i mjesto rođenja:	24.07.2001. Berane, Crna Gora.
Naziv završenog osnovnog studijskog programa i godina završetka studija:	Energetika i automatika, 2022.

INFORMACIJE O MASTER RADU

Naziv master studija:	Automatika i industrijska elektrotehnika
Naslov rada:	Uporedna analiza algoritama za simultanu lokalizaciju i mapiranje u realnom vremenu
Fakultet/Akademija na kojem je rad odbranjen:	Elektrotehnički fakultet, Univerzitet Crne Gore

UDK, OCJENA I ODBRANA MASTER RADA

Datum prijave master rada:	11.03.2025.
Datum sjednice Vijeća na kojoj je prihvaćena tema:	24.04.2025.
Mentor:	Prof. dr Žarko Zečević
Komisija za ocjenu/odbranu rada:	Prof. dr Božo Krstajić Prof. dr Žarko Zečević Prof. dr Milovan Radulović
Datum odbrane:	23.12.2025.

Izjava o autorstvu

Potpisani-a: Miljan Golubović

Broj indeksa/upisa: 1/22

Izjavljujem

da je master rad pod nazivom

"Uporedna analiza algoritama za simultanu lokalizaciju i mapiranje u realnom vremenu"

- rezultat sopstvenog istraživačkog rada,
- da predloženi master rad ni u cjelini ni u djelovima nije bio predložen za dobijanje bilo koje diplome prema studijskim programima drugih ustanova visokog obrazovanja,
- da su rezultati korektno navedeni, i
- da nijesam povrijedio/la autorska i druga prava intelektualne svojine koja pripadaju trećim licima.

U Podgorici, 23.12.2025. godine

Potpis magistranda

Miljan Golubović

Sažetak

Predmet ovog rada su algoritmi za simultanu lokalizaciju robota i mapiranje okruženja (SLAM) u realnom vremenu. U radu je sprovedena detaljna uporedna analiza dvije glavne kategorije SLAM pristupa: parametarskih metoda zasnovanih na Kalmanovim filtrima (EKF-SLAM i UKF-SLAM) i neparametarskih metoda zasnovanih na čestičnim filterima i Rao–Blackwellizaciji (FastSLAM 1.0, FastSLAM 2.0 i uFastSLAM). Ovi pristupi razlikuju se po načinu reprezentacije nesigurnosti, modelovanju stanja i ažuriranju informacija tokom kretanja robota. Eksperimentalni dio obuhvata opsežne simulacije u MATLAB okruženju sa različitim mapama, parametrima algoritama i testovima robusnosti, čime je omogućeno sagledavanje ponašanja algoritama u širokom spektru uslova. Evaluacija performansi izvršena je na osnovu korijena srednje kvadratne greške u estimaciji putanje robota i pozicije obilježja okruženja, dok su dodatno analizirani računski zahtjevi algoritama i sprovedena analiza konzistentnosti estimacije kod parametarskih pristupa. Na osnovu sprovedene analize formulisane su smjernice koje mogu poslužiti kao osnov za izbor odgovarajućeg SLAM pristupa u zavisnosti od tipa senzora, karakteristika radnog prostora i raspoloživih računarskih resursa.

Ključne riječi: SLAM – mobilni robot – procjena – Kalmanov filter – čestični filter – uporedna analiza

Abstract

The subject of this thesis is real-time algorithms for simultaneous robot localization and mapping (SLAM). The thesis presents a detailed comparative analysis of two main categories of SLAM approaches: parametric methods based on Kalman filters (EKF-SLAM and UKF-SLAM) and nonparametric methods based on particle filters and Rao–Blackwellization (FastSLAM 1.0, FastSLAM 2.0, and uFastSLAM). These approaches differ in the way they represent uncertainty, model the state, and update information during robot motion. The experimental part comprises extensive simulations in the MATLAB environment with different maps, algorithm parameters, and robustness tests, which makes it possible to observe the behavior of the algorithms over a wide range of conditions. Performance evaluation is carried out using the root mean square error of the estimated robot trajectory and landmark positions in the environment, while the computational requirements of the algorithms are additionally analyzed, together with a consistency analysis of the estimation for the parametric approaches. Based on the conducted analysis, guidelines are formulated that can serve as a basis for selecting an appropriate SLAM approach depending on the type of sensor, characteristics of the workspace, and available computational resources.

Keywords: SLAM – mobile robot – estimation – Kalman filter – particle filter – comparative analysis

Sadržaj

Spisak slika

Spisak tabela

Uvod

Autonomni mobilni roboti danas imaju sve širu primjenu u industriji, logistici, poljoprivredi i istraživačkim misijama, gdje je neophodno da se kreću samostalno i donose odluke u dinamičnim i često nepoznatim okruženjima. Da bi mogli samostalno da se kreću kroz nepoznato okruženje, autonomni mobilni roboti moraju imati mogućnost da pouzdano percepiraju okolinu i precizno odrede sopstveni položaj u prostoru. Međutim, tačno određivanje položaja otežano je zbog kumulativnih grešaka u odometriji, ograničenog dometa i preciznosti senzora, kao i promjenljivih uslova okruženja. Radi prevazilaženja ovih problema razvijen je koncept simultane lokalizacije i mapiranja (engl. *Simultaneous Localization and Mapping* – SLAM), koji omogućava robotima da u nepoznatom prostoru istovremeno procjenjuju vlastitu poziciju i formiraju mapu okruženja, čime se obezbjeđuje stabilno i precizno kretanje uprkos navedenim nesigurnostima [4–6].

Razvoj SLAM algoritama doveo je do poboljšanja tačnosti, efikasnosti i robusnosti sistema autonomne navigacije, čime je omogućena primjena autonomnih robota u širokom spektru zadataka – od industrijske automatizacije i logistike [7], preko vojne upotrebe [8], do podvodne i vazdušne robotike [9]. U literaturi su razvijeni brojni algoritmi koji problem simultane lokalizacije i mapiranja razmatraju kroz teorijske modele i praktične implementacije. Klasične metode zasnivaju se na filtracionim pristupima, poput Kalmanovih i čestičnih filtera [10, 11], dok savremeniji SLAM pristupi obuhvataju graf-optimizacione metode [12–15], tehnike globalne optimizacije [16, 17], algoritme zasnovane na *smoothing* pristupu [18, 19] i modele dubokog učenja (engl. *Deep Learning*) [20, 21]. Tokom razvoja ove oblasti nastale su brojne varijante SLAM-a, koje se razlikuju prema tipu senzora, dimenzijama prostora i načinu reprezentacije okruženja. Među najzastupljenijima su 2D i 3D SLAM [22], zatim vizuelni SLAM (engl. *Visual SLAM*), uključujući ORB-SLAM (*Oriented FAST and Rotated BRIEF SLAM*), LSD-SLAM (*Large-Scale Direct SLAM*) i DSO-SLAM (*Direct Sparse Odometry SLAM*) [23–25] koji koristi monokularne, stereo i RGB-D kamere, te volumetrijski SLAM koji generiše guste trodimenzionalne mape prostora [26]. Pored njih, razvijeni su i specijalizovani pristupi poput lidar-SLAM-a [27], inercijalnog SLAM-a [28] i multi-robot SLAM-a [29], koji omogućavaju integraciju podataka iz različitih senzora i simultano mapiranje većih okruženja.

U ovom radu razmatrani su algoritmi EKF-SLAM, UKF-SLAM, FastSLAM 1.0, FastSLAM 2.0 i uFastSLAM, koji predstavljaju najznačajnije predstavnike različitih probabilističkih pristupa rješavanju SLAM problema. EKF-SLAM (*Extended Kalman Filter SLAM*) i UKF-SLAM (*Unscented Kalman Filter SLAM*) zasnivaju se na parametarskim metodama procjene stanja, pri čemu se EKF-SLAM oslanja na linearizaciju nelinearnih modela, dok UKF-SLAM koristi *unscented* transformaciju sigma-tačaka za propagaciju srednje vrijednosti i kovarijanse. S druge strane, FastSLAM algoritmi zasnivaju se na neparametarskim metodama procjene stanja, pri čemu koriste čestične filtere (*Particle Filters*) kako bi razdvojili procjenu položaja robota od procjene položaja obilježja. Varijante FastSLAM 1.0, FastSLAM 2.0 i uFastSLAM razlikuju se prema načinu propagacije i ažuriranja kovarijanse robota, kao i poboljšanjima u efikasnosti procesa mapiranja. Izbor ovih algoritama omogućava sveobuhvatno poređenje parametarskih i neparametarskih pristupa u pogledu tačnosti, robusnosti i računske efikasnosti, što predstavlja osnovni cilj analize sprovedene u ovom radu.

Izbor optimalnog SLAM algoritma u praksi ostaje izazov jer njegove performanse zavise od više faktora, uključujući karakteristike senzora, karakteristike šuma, dinamiku okruženja, raspoložive računske resurse, kao i od zahtjeva za tačnošću i brzinom obrade podataka u realnom vremenu. Zbog toga je neophodno ispitati ponašanje različitih algoritama u uslovima koji odražavaju realne scenarije rada, kako bi se identifikovale njihove prednosti i ograničenja i formulisale preporuke za praktičnu primjenu u zavisnosti od uslova okruženja i tehničkih zahtjeva sistema.

Cilj ovog rada je da se eksperimentalnim pristupom kvantitativno uporede predstavnici parametarskih i neparametarskih SLAM pristupa, kako bi se identifikovali uslovi u kojima pojedinačni algoritmi pokazuju prednost. U tu svrhu sproveden je niz kontrolisanih eksperimenata u kojima su varirane kovarijanse procesnog i mjernog šuma, domet senzora i broj čestica, uz dodatne testove robusnosti zasnovane na uvođenju bias-a upravljačkog ugla i balističkog drifta. Evaluacija performansi izvršena je na osnovu korijena srednje kvadratne greške u estimaciji putanje robota i pozicije obilježja okruženja, dok su dodatno analizirani računski zahtjevi algoritama i sprovedena analiza konzistentnosti estimacije kod parametarskih pristupa. Na osnovu rezultata simulacija formulisane su praktične preporuke za izbor odgovarajućeg pristupa u zavisnosti od uslova primjene.

Master rad se sastoji od uvoda, pet poglavlja, zaključka i liste referenci. U prvom poglavlju predstavljeni su osnovni probabilistički koncepti koji se koriste u robotici, izvori nesigurnosti i metode filtriranja stanja, uključujući Bayesov, Kalmanov i čestični pristup. Drugo poglavlje sadrži opis kinematičkih i senzorskih modela robota koji povezuju mjerene veličine sa stvarnim stanjem sistema. U trećem poglavlju detaljno su obrađeni algoritmi EKF-SLAM, UKF-SLAM, FastSLAM 1.0, FastSLAM

2.0 i uFastSLAM, uz objašnjenje njihove matematičke formulacije, principa rada i strukture filtera, te načina ažuriranja stanja i mape, kao i pristupa obradi nelinearnosti i modelovanju nesigurnosti. Četvrto poglavlje posvećeno je analizi računske složenosti i performansi posmatranih algoritama, sa fokusom na njihovu efikasnost i skalabilnost u uslovima nezavisnim od konkretne implementacije. U petom poglavlju prikazani su rezultati simulacija sprovedenih u različitim scenarijima, uz uporednu analizu performansi algoritama u pogledu tačnosti lokalizacije, konzistentnosti mape i robusnosti na šumove. Na kraju, u zaključku su sumirani glavni rezultati rada i razmotrene su mogućnosti daljih istraživanja u oblasti SLAM algoritama i njihove primjene u realnim uslovima.

Glava 1

Nesigurnost i varijabilnosti u robotici

Efikasno izvršavanje zadataka robota zavisi od tačnih i pouzdanih informacija o okruženju u kojem se robot nalazi. Ključna komponenta koja omogućava robotima navigaciju u prostoru je upotreba senzora. Međutim, mjerenja dobijena od senzora uvijek nose određeni stepen greške zbog prisutnosti šumova, smetnji i tehničkih ograničenja samih senzora. Ovakva nesigurnost u podacima negativno utiče na kvalitet navigacije robota i efikasnost izvršenja njegovih zadataka. U nastavku će biti opisani glavni uzroci ove nesigurnosti, kao i metode kojima se ona može umanjiti ili potpuno otkloniti.

1.1 Uvod

Nesigurnosti nastaju kada robot nema dovoljno informacija ili raspolaže netačnim podacima o svom okruženju. Njihovi uzroci mogu se podijeliti u pet kategorija [30]:

- **Okruženje:** Prirodna varijabilnost okruženja (npr. promjene u osvjetljenju, vremenskim uslovima i topografiji), kao i prisustvo dinamičkih prepreka (poput ljudi, vozila i životinja), dovode do povećanog šuma.
- **Senzori:** Zbog ograničenja u dometu i rezoluciji, kao i prisustva fizičkih i elektronskih šumova, senzorska mjerenja su često nepouzdana i varijabilna.
- **Aktuatori:** upravljački šumovi, mehaničko habanje i temperaturne promjene utiču na preciznost kretanja robota, dovodeći do odstupanja između stvarne i očekivane pozicije.
- **Modeli:** Modeli su pojednostavljene aproksimacije stvarnog svijeta. Njihove apstrakcije, ograničene reprezentacije i nemogućnost da prate vremenske promjene dodatno doprinose nesigurnosti.

- **Računarska obrada:** Ograničeni računarski resursi, kašnjenja u obradi i numeričke greške utiču na tačnost procjene stanja i brzinu reagovanja sistema, naročito u uslovima rada u realnom vremenu.

S obzirom na sve navedene izvore nesigurnosti, od suštinskog je značaja primjena metoda za obradu podataka, kao što su tehnike filtriranja, radi umanjenja njihovog uticaja na performanse sistema.

1.2 Princip maksimalne vjerovatnoće

U mnogim aplikacijama, posebno u oblasti robotike i obrade podataka, često se suočavamo sa zadatkom procjene nepoznatih parametara na osnovu ograničenih i često zašumljenih mjerenja. Kako bi procjene bile pouzdane, neophodno je koristiti matematičke metode koje omogućavaju izvlačenje najvjerovatnijih vrijednosti parametara na osnovu dostupnih podataka.

Jedan od najčešće korišćenih pristupa u ovom kontekstu je metoda maksimalne vjerovatnoće, koja pruža sistematski način za izbor vrijednosti parametra koja najbolje odgovara posmatranim podacima. U nastavku je dat osnovni pregled ovog pristupa.

Pretpostavimo da imamo skup podataka X, koji predstavlja slučajni uzorak iz distribucije zavisne od nepoznatog parametra α. Da bismo naglasili zavisnost distribucije od tog parametra, koristimo sledeće oznake:

- $p(X \mid \alpha)$ – kada je u pitanju diskretna distribucija (funkcija vjerovatnoće),
- $f(X \mid \alpha)$ – kada je u pitanju kontinuirana distribucija (funkcija gustine vjerovatnoće).

Bez obzira da li je riječ o diskretnoj ili kontinuiranoj distribuciji, princip maksimalne vjerovatnoće nalaže da procijenimo parametar α kao onu vrijednost koja maksimizira vjerovatnoću (ili gustinu vjerovatnoće) posmatranog uzorka. Ta vrijednost označava se kao procjena maksimalne vjerovatnoće $\hat{\alpha}$, i definiše se kao [31]:

$$\hat{\alpha} = \arg\max_{\alpha} L(\alpha \mid X), \tag{1.1}$$

gdje je $L(\alpha \mid X)$ funkcija vjerovatnoće – odnosno funkcija gustine u slučaju kontinuiranih distribucija.

1.3 Probabilistička reprezentacija znanja

U realnim robotskim sistemima, jedno od ograničenja je nemogućnost potpunog i direktnog posmatranja ili mjerenja stanja robota. Na primjer, precizna pozicija i orijentacija robota u prostoru obično nisu dostupne neposredno – čak ni sa najnaprednijim senzorima [10]. Umjesto toga, robot procjenjuje svoje stanje na osnovu prikupljenih podataka. Drugim riječima, stvarno stanje robota razlikuje se od njegovog procijenjenog stanja.

Zbog ove nesigurnosti, robotski sistemi ne mogu bazirati svoju unutrašnju reprezentaciju isključivo na pojedinačnim informacijama iz senzora, već moraju održavati cijelu distribuciju mogućih stanja, uz odgovarajuće vjerovatnoće svakog od tih stanja. Ova distribucija odražava sve što robot zna o svojoj lokaciji, orijentaciji ili bilo kojoj drugoj promjenljivoj.

Ovakvo unutrašnje znanje o stanju, koje se bazira na svim do sada dostupnim mjerenjima i upravljačkim signalima nazivamo vjerovanje (engl. *belief*). Drugim riječima, vjerovanje predstavlja procjenu (u probabilističkom smislu) koja stanja su vjerovatna, a koja manje vjerovatna, s obzirom na sve što je robot do sada detektovao i uradio.

Vjerovanje nad promjenjivom stanja x_t označavamo kao $bel(x_t)$, što predstavlja skraćenicu za posteriornu vjerovatnoću:

$$bel(x_t) = p(x_t \mid z_{1:t}, u_{1:t}), \tag{1.2}$$

ova distribucija opisuje vjerovatnoću stanja x_t u trenutku t, uslovljenu svim dosadašnjim mjerenjima $z_{1:t}$ i upravljačkim signalima $u_{1:t}$.

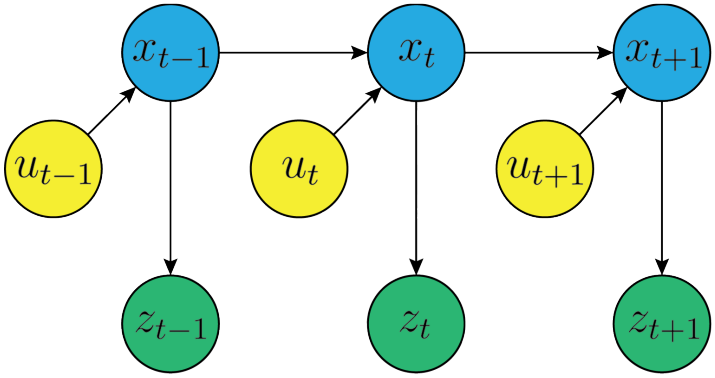
Važno je primijetiti da se ovdje vjerovanje odnosi na procjenu stanja nakon uključivanja najnovijeg mjerenja z_t. Međutim, u nekim slučajevima korisno je izračunati *posterior* i prije ažuriranja novim mjerenjem, tj. odmah nakon što je izvršena kontrola u_t, ali prije nego što je dobijeno mjerenje z_t. Takva distribucija zapisuje se kao:

$$\overline{bel}(x_t) = p(x_t \mid z_{1:t-1}, u_{1:t}). \tag{1.3}$$

Ova distribucija često se u kontekstu probabilističkog filtriranja naziva predikcija (engl. *prediction*). Terminologija ukazuje na to da $\overline{bel}(x_t)$ predviđa stanje u trenutku t na osnovu prethodnog posteriora, a prije ažuriranja sa najnovijim mjerenjem. Proračunavanje $bel(x_t)$ iz $\overline{bel}(x_t)$ naziva se korekcija (engl. *correction*) ili mjerno ažuriranje (engl. *measurement update*).

1.4 Bayesov filter

Trenutna pozicija robota označava se kao x_t. Tokom vremena, robot prelazi iz prethodnog stanja x_{t-1} u trenutno stanje x_t pod uticajem upravljačkog signala u_t, dok istovremeno generiše odgovarajuće mjerenje z_t. Veza između stanja robota, upravljačkih signala i mjerenja ilustrovana je na slici 1.1, i predstavlja osnovu za primjenu Bayesovog filtera. Ova veza omogućava ažuriranje distribucije vjerovanja trenutnog stanja na osnovu novih mjerenja i promjena u sistemu.



Slika 1.1: Dinamička Bayesova mreža koja karakteriše evoluciju upravljanja, stanja i mjerenja

Bayesov filter predstavlja najopštiji probabilistički okvir za rekurzivnu estimaciju trenutnog stanja robota x_t, zasnovan na kombinovanju prethodnog vjerovanja $bel(x_{t-1})$, upravljačkih signala u_t i mjerenja z_t. Na taj način, sistem u svakom vremenskom koraku ažurira svoje znanje o sopstvenoj poziciji i okruženju. Rekurzivni Bayesov algoritam prikazan je u Algoritmu 1.

Algoritam 1 BAYES_FILTER$(bel(x_{t-1}), u_t, z_t)$

1: **for all** x_t **do**
2: $\quad \overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1}) \cdot bel(x_{t-1})\, dx_{t-1}$
3: $\quad bel(x_t) = \eta \cdot p(z_t \mid x_t)^1 \cdot \overline{bel}(x_t)$
4: **end for**
5: **return** $bel(x_t)$

Bayesov filter se sastoji od dva osnovna koraka [30]:

1. Prvi korak, poznat kao predikcioni korak, prikazan je u drugoj liniji Algoritma 1. U njemu se procjenjuje vjerovanje za trenutno stanje x_t na osnovu pret-

[1] $p(z_t \mid x_t, z_{1:t-1}, u_{1:t}) \approx p(z_t \mid x_t)$ – Markovljeva pretpostavka, odnosi se na svojstvo stohastičkih procesa, prema kojoj je buduće stanje sistema nezavisno od prethodnih stanja, pod uslovom da je trenutno stanje poznato [30].

hodnog vjerovanja $bel(x_{t-1})$ i poznatog upravljanja u_t. Vjerovanje $bel(x_t)$ dobija se integrisanjem proizvoda dvije raspodjele: prethodnog vjerovanja $bel(x_{t-1})$ i vjerovatnoće da upravljanje u_t uzrokuje prelaz iz x_{t-1} u x_t.

2. Drugi korak, poznat kao korekcioni korak, prikazan je u trećoj liniji Algoritma 1. U njemu se trenutno vjerovanje $bel(x_t)$ množi sa vjerovatnoćom da bi posmatrano mjerenje z_t moglo biti dobijeno iz stanja x_t. Ova operacija se vrši za svako moguće stanje x_t. Pošto rezultat množenja ne mora biti pravilna vjerovatnoća (odnosno ne mora se integrisati na 1), koristi se normalizacioni faktor η. Time se dobija konačno vjerovanje $bel(x_t)$, koje se vraća u petoj liniji Algoritma 1.

Za rekurzivnu estimaciju vjerovanja kroz vrijeme, algoritam zahtijeva početno vjerovanje $bel(x_0)$ u trenutku $t = 0$, koje služi kao granični uslov. Ako je početna vrijednost x_0 poznata sa sigurnošću, vjerovanje se inicijalizuje raspodjelom vjerovatnoće koncentrisanom u tački x_0:

$$bel(x_0) = \delta(x_0 - \hat{x}_0), \tag{1.4}$$

gdje $\delta(\cdot)$ predstavlja Dirakovu delta funkciju, a $\hat{x}_0$ je poznata vrijednost početnog stanja. U slučaju potpune neizvjesnosti o početnom stanju, koristi se uniformna raspodjela preko definisanog domena:

$$bel(x_0) = \text{const.}, \tag{1.5}$$

što označava da je vjerovatnoća jednako raspoređena preko svih mogućih vrijednosti x_0. Djelimično znanje može se predstaviti neuniformnim raspodjelama, ali se u praksi najčešće koriste ekstremi: potpuna sigurnost ili potpuna neizvjesnost.

Bayesov filter, u ovom obliku, primjenljiv je samo na jednostavne probleme. Da bi bio upotrebljiv u praksi, potrebno je da se izrazi u linijama 3 i 4 mogu izračunati direktno, ili da je broj stanja ograničen, tako da se integrali mogu zamijeniti sumama.

1.5 Rekurzivna estimacija

Kao ilustraciju rekurzivne estimacije, razmotrimo problem izračunavanja srednje vrijednosti slučajne promjenjive na osnovu niza zašumljenih mjerenja koja pristižu sukcesivno. Pretpostavimo da se procjena srednje vrijednosti ažurira pri svakom novom mjerenju. Najjednostavniji pristup je čuvanje svih prethodnih mjerenja i računanje aritmetičke sredine svih do tada dostupnih podataka u svakom koraku. Takav pristup, iako tačan, postaje neefikasan kako broj mjerenja raste – zbog sve većih zahtjeva za memorijom i obradom.

Međutim, moguće je izračunavati srednju vrijednost rekurzivno, bez potrebe za čuvanjem svih prethodnih mjerenja. Osnovna ideja je da se nova procjena gradi na osnovu prethodne procjene i posljednjeg mjerenja:

$$\bar{x}_k = \frac{k-1}{k}\bar{x}_{k-1} + \frac{1}{k}x_k, \tag{1.6}$$

gdje je $\bar{x}_k$ nova srednja vrijednost nakon k mjerenja, a x_k trenutno mjerenje. Ovaj postupak omogućava kontinuirano ažuriranje procjene koristeći samo prethodni rezultat i novo mjerenje, čime se značajno smanjuju memorijski i računarski zahtjevi.[2]

Ovakav način računanja predstavlja osnovni primjer rekurzivnog algoritma. Ključna karakteristika rekurzivnih metoda jeste korišćenje rezultata iz prethodnog koraka za izračunavanje nove vrijednosti, bez potrebe za ponovnim procesiranjem kompletnih podataka. Upravo se ova ideja nalazi u osnovi Kalmanovog filtera, koji se koristi za optimalnu estimaciju u prisustvu šuma. Time se Kalmanov pristup razlikuje od alternativnih metoda kao što je, na primjer, Wienerov filter [32].

1.6 Kalmanove metode filtriranja

Stohastički šumovi u realnim sistemima nastaju kao posljedica nepreciznosti senzora, spoljašnjih smetnji i tehničkih ograničenja samih sistema. Njihovo ponašanje je nasumično i teško predvidljivo, što otežava tačnu procjenu stanja sistema. U praksi se često pretpostavlja da se šum može modelovati pomoću normalne (Gaussove) raspodjele, što znatno pojednostavljuje problem modelovanja.

Ova pretpostavka se zasniva na centralnoj graničnoj teoremi, prema kojoj zbir velikog broja nezavisnih slučajnih promjenjivih teži normalnoj raspodjeli, bez obzira na oblik njihovih pojedinačnih raspodjela. Ovaj rezultat je prvi formalno predstavio De Moavre 1733. godine [33].

U ovim uslovima, moguće je koristiti specijalizovane varijante Bayesovog filtera koje omogućavaju efikasniju implementaciju. Najpoznatije među njima su: Kalmanov filter (KF), Prošireni Kalmanov filter (EKF) i Unscented Kalmanov filter (UKF).

1.6.1 Kalmanov filter

Kalmanov filter, kao najpoznatija implementacija Bayesovog filtera, omogućava procjenu vjerovanja nad realnim (kontinuiranim) stanjima. U svakom vremenskom

[2]Rekurzivno ažuriranje može biti podložno nakupljanju numeričkih grešaka usljed ograničene preciznosti računanja s realnim brojevima (tzv. *floating-point* greške). Ipak, u većini praktičnih slučajeva ove greške su zanemarljive, naročito ako se koristi stabilna numerička implementacija sa dovoljno visokom preciznošću.

trenutku t, filter računa srednju vrijednost μ_t i kovarijansu Σ_t vjerovanja za trenutno stanje. Vjerovanje $bel(x_t)$ prati Gaussovu raspodjelu, pod uslovom da su, uz važenje Markovljeve pretpostavke, ispunjena tri ključna uslova [30]:

1. **Model kretanja:** Model tranzicije stanja $p(x_t \mid u_t, x_{t-1})$, koji opisuje vjerovatnoću trenutnog stanja na osnovu prethodnog stanja i upravljačkog signala, mora biti linearan u svojim argumentima uz dodatak Gausovog šuma. Ovaj odnos je dat sljedećom formulom:

$$x_t = A_t x_{t-1} + B_t u_t + \varepsilon_t, \tag{1.7}$$

   gdje su:

   - x_t, x_{t-1} – vektori stanja dimenzije n,
   - u_t – vektor upravljačkog signala dimenzije m,
   - A_t – matrica prelaza stanja dimenzije $n \times n$,
   - B_t – matrica upravljanja dimenzije $n \times m$,
   - $\varepsilon_t \sim \mathcal{N}(0, R_t)$ – Gaussov šum u modelu kretanja sa kovarijansom R_t.

2. **Mjerni model:** Mjerni model $p(z_t \mid x_t)$, koji povezuje izlazni signal z_t za dato stanje x_t, takođe mora biti linearan u svojim argumentima uz dodati Gausov šum. Ovaj odnos je dat sljedećom formulom:

$$z_t = C_t x_t + \delta_t, \tag{1.8}$$

   gdje su:

   - z_t – vektor mjerenja dimenzije k,
   - C_t – matrica mjernog modela dimenzije $k \times n$,
   - $\delta_t \sim \mathcal{N}(0, Q_t)$ – Gaussov šum mjerenja sa kovarijansom Q_t.

3. **Početno vjerovanje** $bel(x_0)$ mora biti normalne raspodjele:

$$bel(x_0) = \mathcal{N}(\mu_0, \Sigma_0). \tag{1.9}$$

Pretpostavlja se da su matrice A_t, B_t i C_t konstantne ili unaprijed poznate, kao što to zahtijeva formulacija Kalmanovog filtera.

Kada su ovi uslovi ispunjeni, vjerovanje $bel(x_t)$ u svakom trenutku ostaje u obliku normalne raspodjele. Pseudokod za implementaciju Kalmanovog filtera prikazan je u Algoritmu 2.

Algoritam 2 Kalmanov Filter

1: **function** KALMAN_FILTER($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$)
2: $\quad \bar{\mu}_t = A_t\mu_{t-1} + B_t u_t$
3: $\quad \bar{\Sigma}_t = A_t\Sigma_{t-1}A_t^T + R_t$
4: $\quad K_t = \bar{\Sigma}_t C_t^T (C_t\bar{\Sigma}_t C_t^T + Q_t)^{-1}$
5: $\quad \mu_t = \bar{\mu}_t + K_t(z_t - C_t\bar{\mu}_t)$
6: $\quad \Sigma_t = (I - K_t C_t)\bar{\Sigma}_t$
7: $\quad$ **return** μ_t, Σ_t
8: **end function**

Linije 2 i 3 predstavljaju *predikcioni korak*, gdje se određuju očekivano stanje i njegova kovarijansa. Linija 4 računa *Kalmanovo pojačanje* K_t, koje određuje odnos povjerenja između predikcije i mjerenja, u skladu s njihovim nesigurnostima. Linije 5 i 6 predstavljaju *korekcioni korak*, u kojem se stanje ažurira na osnovu pristiglih mjerenja.

Važno je napomenuti da čak i u idealnom slučaju bez mjernog šuma, nesigurnost uzrokovana modelom kretanja ostaje prisutna. Kalmanov filter, u suštini, računa ponderisanu sredinu između predikcije zasnovane na modelu kretanja i informacija dobijenih od senzora, čime omogućava optimalnu procjenu stanja. Ilustracija rezultata ovog procesa prikazana je na slici 1.2.

1.6.2 Prošireni Kalmanov filter (EKF)

Prošireni Kalmanov filter (EKF) predstavlja ekstenziju klasičnog Kalmanovog filtera, prilagođenu za rad sa nelinearnim sistemima [30,34]. U stvarnim scenarijima, kao što je navigacija mobilnog robota, kretanje često uključuje promjenu pravca i orijentacije, što uvodi nelinearnosti u sistem – najčešće kroz funkcije poput $\sin(\cdot)$ i $\cos(\cdot)$.

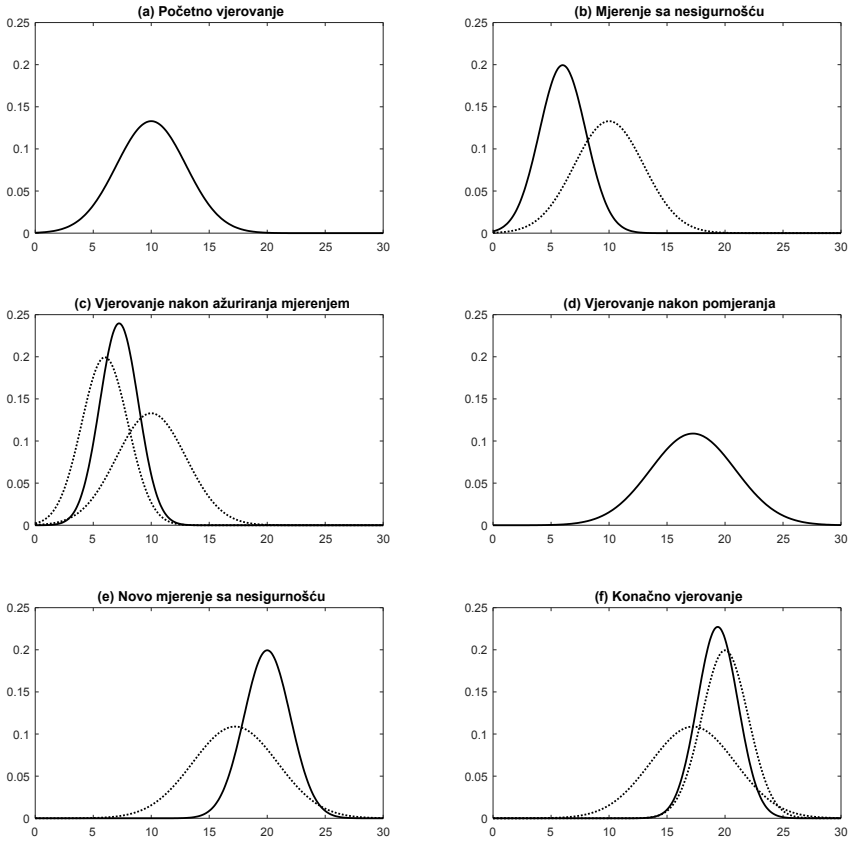
Kako bi se omogućila primjena principa Kalmanovog filtriranja u takvim uslovima, razvijen je prošireni Kalmanov filter (EKF), koji koristi lokalnu linearnu aproksimaciju nelinearnih funkcija pomoću Taylorevog reda, pri čemu se zadržava samo prvi (linearni) član. Na taj način, nelinearne funkcije tranzicije stanja $g(\cdot)$ i mjernog modela $h(\cdot)$ linearizuju se u okolini trenutne procjene stanja.

Za funkciju tranzicije stanja (model kretanja) $g(x_{t-1}, u_t)$, linearizacija oko procjene μ_{t-1} daje:

$$x_t \approx g(\mu_{t-1}, u_t) + G_t(x_{t-1} - \mu_{t-1}), \tag{1.10}$$

gdje je G_t Jakobijeva matrica funkcije g, izračunata kao:

$$G_t = \left.\frac{\partial g}{\partial x}\right|_{x=\mu_{t-1}, u=u_t}. \tag{1.11}$$



Slika 1.2: Vizuelna interpretacija Kalmanovog filtera kroz korake predikcije i korekcije. Grafici prikazuju promjenu vjerovanja usljed mjerenja i kretanja, sa nesigurnošću modelovanom Gaussovim raspodjelama.

Analogno, mjerna funkcija (mjerni model) $h(x_t)$ linearizuje se oko predikcije $\bar{\mu}_t$ kao:

$$z_t \approx h(\bar{\mu}_t) + H_t(x_t - \bar{\mu}_t), \tag{1.12}$$

pri čemu je H_t Jakobijan mjernog modela:

$$H_t = \left.\frac{\partial h}{\partial x}\right|_{x=\bar{\mu}_t}. \tag{1.13}$$

Prošireni Kalmanov filter koristi Jakobijane G_t i H_t umjesto matrica A_t, B_t i C_t iz klasične formulacije Kalmanovog filtera. Konkretno:

- G_t zamjenjuje matrice tranzicije i upravljanja (A_t i B_t),
- H_t zamjenjuje matricu mjernog modela (C_t).

Ostatak algoritma primjenjuje se na isti način kao kod linearnog Kalmanovog filtera. Pseudo-kod za EKF dat je u Algoritmu 3.

Algoritam 3 Prošireni Kalmanov filter

function EXTENDED_KALMAN_FILTER($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$)
  $\bar{\mu}_t = g(u_t, \mu_{t-1})$
  $\bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R_t$
  $K_t = \bar{\Sigma}_t H_t^T \left(H_t \bar{\Sigma}_t H_t^T + Q_t\right)^{-1}$
  $\mu_t = \bar{\mu}_t + K_t \left(z_t - h(\bar{\mu}_t)\right)$
  $\Sigma_t = (I - K_t H_t)\bar{\Sigma}_t$
  return μ_t, Σ_t
end function

1.6.3 Unscented Kalmanov filter (UKF)

Prošireni Kalmanov filter (EKF) aproksimira raspodjelu stanja Gaussovom slučajnom promjenjivom, koju propagira kroz nelinearni sistem koristeći linearnu aproksimaciju prvog reda oko trenutne procjene. U tu svrhu koristi se Jakobijan za proračun srednje vrijednosti i kovarijanse. Ipak, budući da se linearizacija vrši samo u jednoj tački (trenutno procijenjenom stanju) EKF može značajno odstupati od stvarne raspodjele, naročito u sistemima sa izraženim nelinearnostima.

Alternativu ovakvom pristupu nudi *Unscented transformacija* (UT), koja zaobilazi potrebu za derivacijama. Umjesto da linearizuje funkciju, UT koristi skup *sigma tačaka* sa ciljem pokrivanja ključnih aspekata raspodjele stanja. Ove tačke se direktno propagiraju kroz nelinearnu funkciju, a zatim se na osnovu rezultata računa nova srednja vrijednost i kovarijansa. Takav pristup omogućava preciznije aproksimacije bez računanja izvoda [10, 35].

Pretpostavimo da imamo slučajnu promjenjivu $x \in \mathbb{R}^n$ sa srednjom vrijednošću μ i kovarijansom Σ, koju želimo propagirati kroz nelinearnu funkciju $y = g(x)$. UT koristi $2n + 1$ sigma tačaka, koje se određuju kao:

$$\chi^{[0]} = \mu, \tag{1.14}$$

$$\chi^{[i]} = \mu + \left(\sqrt{(n+\lambda)\Sigma}\right)_i, \qquad i = 1, \ldots, n, \tag{1.15}$$

$$\chi^{[i]} = \mu - \left(\sqrt{(n+\lambda)\Sigma}\right)_{i-n}, \qquad i = n+1, \ldots, 2n. \tag{1.16}$$

Svakoj sigma tački se zatim dodjeljuje odgovarajući *težinski koeficijent*, koji se koristi za izračunavanje srednje vrijednosti i kovarijanse rezultujuće raspodjele:

$$W_m^{[0]} = \frac{\lambda}{n+\lambda}, \tag{1.17}$$

$$W_c^{[0]} = W_m^{[0]} + \left(1 - \alpha^2 + \beta\right), \tag{1.18}$$

$$W_m^{[i]} = W_c^{[i]} = \frac{1}{2(n+\lambda)}, \qquad i = 1, \ldots, 2n. \tag{1.19}$$

Nakon propagacije sigma tačaka kroz funkciju g,

$$y_i = g(\chi_i), \quad i = 0, \ldots, 2n, \tag{1.20}$$

nova srednja vrijednost i kovarijansa se dobijaju kao:

$$\mu' = \sum_{i=0}^{2n} W_m^{[i]} y_i, \tag{1.21}$$

$$\Sigma' = \sum_{i=0}^{2n} W_c^{[i]} (y_i - \mu')(y_i - \mu')^T. \tag{1.22}$$

Parametri su definisani na sljedeći način:

- $\lambda = \alpha^2(n + \kappa) - n$ – ukupni parametar skaliranja sigma tačaka,
- α – kontroliše širinu rasporeda sigma tačaka (tipično 10^{-3}),
- κ – dodatni slobodni parametar (često 0),
- β – unosi informaciju o obliku raspodjele (npr. $\beta = 2$ za Gaussovu distribuciju).

Unscented Kalmanov filter (UKF) koristi ovu transformaciju u fazama predikcije i korekcije, bez potrebe za linearnom aproksimacijom. Njegova asimptotska složenost jednaka je kao kod EKF-a[3], ali u mnogim slučajevima daje tačnije procjene, posebno pri nelinearnim transformacijama stanja i mjerenja. UKF, za razliku od EKF-a, ne zahtijeva izračunavanje Jakobijana, što ga čini pogodnim za sisteme u kojima je teško definisati ili izvesti analitičke izvode. Zbog toga se često opisuje kao *filter bez derivacije* [10, 35].

Algoritam 4 Unscented Kalman Filter (UKF)

function UNSCENTED_KALMAN_FILTER($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$)

$\chi_{t-1}^{[i]} = \left(\mu_{t-1},\ \mu_{t-1} + \sqrt{(n+\lambda)\Sigma_{t-1}},\ \mu_{t-1} - \sqrt{(n+\lambda)\Sigma_{t-1}}\right)$

$\chi_t^{*[i]} = g(u_t, \chi_{t-1}^{[i]})$

$\bar{\mu}_t = \sum_{i=0}^{2n} W_m^{[i]} \chi_t^{*[i]}$

$\bar{\Sigma}_t = \sum_{i=0}^{2n} W_c^{[i]} (\chi_t^{*[i]} - \bar{\mu}_t)(\chi_t^{*[i]} - \bar{\mu}_t)^T + R_t$

$\bar{\chi}_t^{[i]} = \left(\bar{\mu}_t,\ \bar{\mu}_t + \sqrt{(n+\lambda)\bar{\Sigma}_t},\ \bar{\mu}_t - \sqrt{(n+\lambda)\bar{\Sigma}_t}\right)$

$Z_t^{[i]} = h(\bar{\chi}_t^{[i]})$

$\hat{z}_t = \sum_{i=0}^{2n} W_m^{[i]} Z_t^{[i]}$

$S_t = \sum_{i=0}^{2n} W_c^{[i]} (Z_t^{[i]} - \hat{z}_t)(Z_t^{[i]} - \hat{z}_t)^T + Q_t$

[3]Ipak, u praksi se EKF često pokazuje nešto bržim, budući da UKF zahtijeva dodatne proračune vezane za generisanje i propagaciju sigma tačaka. Uprkos tom konstantnom faktoru usporenja, UKF ostaje veoma efikasan.

10: $\Sigma_{xz} = \sum_{i=0}^{2n} W_c^{[i]}(\bar{\chi}_t^{[i]} - \bar{\mu}_t)(Z_t^{[i]} - \hat{z}_t)^T$
11: $K_t = \Sigma_{xz} S_t^{-1}$
12: $\mu_t = \bar{\mu}_t + K_t(z_t - \hat{z}_t)$
13: $\Sigma_t = \bar{\Sigma}_t - K_t S_t K_t^T$
14: **return** μ_t, Σ_t
15: **end function**

1.7 Particle Filter

Kalmanovi filteri se koriste za procjenu vjerovatnoće stanja sistema pod pretpostavkom da su raspodjele Gaussove prirode i da su modeli blago nelinearni. U takvim slučajevima, Kalman filteri pružaju efikasno rješenje. Međutim, kada ove pretpostavke nijesu ispunjene, neophodno je koristiti alternativne metode. Jedna od njih je *Particle filter* (PF), neparametarska implementacija Bayesovog filtera. Za razliku od Kalmanovih metoda koje koriste parametarsku reprezentaciju (npr. Gaussovu raspodjelu), particle filteri aproksimiraju vjerovatnoću konačnim skupom uzoraka (*čestica*). Ovakav pristup omogućava modelovanje širokog spektra raspodjela i precizno opisivanje nelinearnih transformacija slučajnih promjenjivih [10].

U okviru particle filtera, svaka čestica $x_t^{[m]}$ sa pridruženim težinskim koeficijentom $w_t^{[m]}$ predstavlja hipotezu o stvarnom stanju sistema u trenutku t. Skup od M čestica $\langle x_t^{[m]}, w_t^{[m]} \rangle_{m=1}^{M}$ definiše aproksimaciju vjerovatnoće stanja. Ova se vjerovatnoća može zapisati kao suma Dirakovih delta funkcija centriranih u svakoj čestici:

$$bel(x_t) \approx \sum_{m=1}^{M} w_t^{[m]} \delta(x_t - x_t^{[m]}). \tag{1.23}$$

Ideja je da distribucija $bel(x_t)$ bude aproksimirana tako da su oblasti sa većom vjerovatnoćom reprezentovane većim brojem čestica, dok oblasti sa manjom vjerovatnoćom imaju manje čestica. U idealnom slučaju, vjerovatnoća da neka hipoteza x_t bude uključena u skup čestica treba da bude proporcionalna izlazu Bayesovog filtera:

$$x_t^{[m]} \sim p(x_t \mid z_{1:t}, u_{1:t}). \tag{1.24}$$

Poput svih Bayesovih metoda, particle filter rekurzivno procjenjuje vjerovatnoću $p(x_t \mid z_{1:t})$ koristeći prethodnu procjenu $p(x_{t-1} \mid z_{1:t-1})$. Ovo se postiže izgradnjom novog skupa čestica $\langle x_t^{[m]}, w_t^{[m]} \rangle$ iz prethodnog skupa $\langle x_{t-1}^{[m]}, w_{t-1}^{[m]} \rangle$, čime se aproksimira vjerovanje $bel(x_t)$.

U Algoritmu 5 prikazan je pseudokod osnovne verzije *particle filter* algoritma. Operacije *sampling* i *resampling* predstavljene su apstraktno, dok će njihova detaljna

formulacija biti razmotrena u nastavku.

Algoritam 5 Particle Filter

function PARTICLE_FILTER(χ_{t-1}, u_t, z_t)
  $\bar{\chi}_t = \chi_t = 0$
  Sampling:
  for $m = 1$ to M **do**
    sample $x_t^{[m]} \sim p(x_t \mid u_t, x_{t-1}^{[m]})$
    $w_t^{[m]} = p(z_t \mid x_t^{[m]})$
    $\bar{\chi}_t = \bar{\chi}_t \cup \langle x_t^{[m]}, w_t^{[m]} \rangle$
  end for
  Resampling:
  for $m = 1$ to M **do**
    draw i with probability $\propto w_t^{[i]}$
    add $x_t^{[i]}$ to χ_t
  end for
  return χ_t
end function

Osnovni koraci Particle filtera:

- **Ulazni parametri:** prethodna raspodjela χ_{t-1}, upravljački signal u_t i mjerenje z_t.
- **Inicijalizacija:** kreiramo $\bar{\chi}_t$ i χ_t, prazne skupove čestica i njihovih težinskih koeficijenata za trenutni korak.
- **Predikcija:** za svaku česticu $x_{t-1}^{[m]} \in \chi_{t-1}$, uz pomoć modela kretanja robota $p(x_t \mid u_t, x_{t-1}^{[m]})$ generišemo novu česticu $x_t^{[m]}$. Skup svih novih čestica čini predikciju vjerovanja $\bar{\chi}_t$.
- **Ažuriranje težinskih faktora:** za svaku česticu računamo težinski faktor

$$w_t^{[m]} = p(z_t \mid x_t^{[m]}), \tag{1.25}$$

  koji označava vjerovatnoću da je mjerenje z_t dobijeno iz stanja koje čestica predstavlja.
- **Ažuriranje skupa čestica:** skup vjerovanja se ažurira tako što se svakoj čestici $x_t^{[m]}$ dodjeljuje odgovarajući težinski faktor $w_t^{[m]}$, čime se formira težinski skup:

$$\bar{\chi}_t = \{\langle x_t^{[m]}, w_t^{[m]} \rangle\}_{m=1}^{M}. \tag{1.26}$$

- **Resampling:** na osnovu težinskih faktora $w_t^{[j]}$ vrši se *resampling* – proces

kojim se čestice sa većom vjerovatnoćom češće odabiraju za novi skup, dok se čestice sa manjom vjerovatnoćom zanemaruju. Rezultat je novi skup χ_t sa ponovljenim (dupliranim) česticama u regionima veće vjerovatnoće. Ovaj proces dovodi do grupisanja čestica u regionima sa visokom vjerovatnoćom, što rezultira robusnijom i preciznijom procjenom stanja.

1.7.1 Importance Sampling

S ciljem da izračunamo očekivanje u odnosu na neku ciljnu distribuciju f, često se suočavamo s ograničenjem da uzorke možemo generisati samo iz neke druge, jednostavnije distribucije g. U takvim slučajevima, očekivanje se može izraziti preko distribucije g, korišćenjem metode *važnosnog uzorkovanja*:

$$\begin{aligned} E_f\left[I(x \in A)\right] &= \int f(x) I(x \in A)\, dx \\ &= \int \frac{f(x)}{g(x)} g(x) I(x \in A)\, dx \\ &= E_g\left[w(x) I(x \in A)\right]. \end{aligned} \tag{1.27}$$

U ovom izrazu, I označava *indikatorsku funkciju* koja ima vrijednost 1 ako je uslov ispunjen, a 0 u suprotnom. Ključnu ulogu ima *težinski faktor* $w(x)$, koji koriguje razliku između distribucija f i g, omogućavajući da se uzorci iz g koriste za procjenu očekivanja nad f.

Funkcija f se naziva *ciljna distribucija* i predstavlja željenu raspodjelu, dok je g *predložena distribucija* – obično jednostavnija za generisanje uzoraka, ali različita od f.

Kumulativna vjerovatnoća bilo kog podskupa A distribucije g, koja odgovara površini ispod krive g u opsegu A, može se aproksimirati udjelom čestica koje padaju unutar tog opsega:

$$\frac{1}{M} \sum_{m=1}^{M} I(x^{[m]} \in A) \xrightarrow{M \to \infty} \int_A g(x)\, dx. \tag{1.28}$$

Da bi se nadoknadila razlika između $f(x)$ i $g(x)$, česticama generisanim iz $g(x)$ dodjeljuju se težinski faktori na osnovu odnosa njihovih funkcija gustina vjerovatnoće:

$$w^{[m]} = \frac{f(x^{[m]})}{g(x^{[m]})}. \tag{1.29}$$

Težinski faktori $w^{[m]}$ predstavljaju nenormalizovane vjerovatnoće svake čestice. Da bi se dobila korektna procjena, potrebno ih je normalizovati. Time dobijamo konačnu

aproksimaciju integrala preko ciljne distribucije:

$$\frac{\sum_{m=1}^{M} I(x^{[m]} \in A) w^{[m]}}{\sum_{m=1}^{M} w^{[m]}} \xrightarrow{M \to \infty} \int_A f(x)\, dx. \tag{1.30}$$

Ova relacija pokazuje da normalizovana ponderisana suma uzoraka konvergira ka očekivanoj vrijednosti u odnosu na ciljnu raspodjelu $f(x)$. U ovom izrazu, imenilac služi kao *normalizator* težinskih faktora, osiguravajući da njihova suma iznosi 1. Na taj način se dobija ispravna aproksimacija očekivanja pod *ciljnom* distribucijom $f(x)$.

U kontekstu *Particle filtera*, funkcija f predstavlja ciljnu distribuciju vjerovanja $\text{bel}(x_t)$, dok funkcija g odgovara distribuciji proizvoda:

$$p(x_t \mid u_t, x_{t-1}) \cdot \text{bel}(x_{t-1}), \tag{1.31}$$

koja objedinjuje model kretanja sa prethodnim vjerovanjem.

1.7.2 Resampling

Resampling se koristi za odbacivanje čestica sa malim težinskim faktorima i fokusiranje računarskih resursa na čestice sa većim težinama, koje bolje reprezentuju vjerovatnu poziciju sistema. Međutim, ovaj proces uvodi dodatnu varijansu, poznatu kao *varijansa uzorkovanja*, koja se povećava ponovljenim resamplovanjem, dovodeći do gubitka raznolikosti u skupu čestica.

Razmotrimo ekstremni slučaj u kojem robot ostaje statičan, bez senzorskih informacija. U takvoj situaciji, idealna procjena njegove pozicije trebalo bi da ostane nepromijenjena kroz vrijeme. Ipak, standardni particle filter, koji u svakom koraku vrši resampling, ne garantuje ovakav ishod. Tokom resamplovanja, neke čestice se nasumično odbacuju, a nove se ne uvode jer je prelaz stanja deterministički. Vremenom, to dovodi do situacije u kojoj sve čestice konvergiraju ka jednoj – što lažno sugeriše da je robot sa sigurnošću odredio svoju poziciju, iako nema relevantna mjerenja. Ovaj fenomen poznat je kao *kolaps čestica* i predstavlja jedno od ključnih ograničenja osnovne implementacije particle filtera.

Jedan način da se smanji uticaj ovog problema jeste kontrola učestalosti resamplovanja. Ako je poznato da je stanje statičko – kao u slučaju mirovanja robota – resamplovanje se može privremeno suspendovati kako bi se sačuvala raznolikost čestica. Čak i kada se robot kreće, nije nužno da se resamplovanje vrši u svakom vremenskom koraku. Umjesto toga, može se akumulirati više mjerenja i izvršiti jedno resamplovanje sa ažuriranim težinama koje odražavaju kumulativnu vjerovatnoću:

$$w_t^{[m]} \leftarrow w_{t-1}^{[m]} \cdot p(z_t \mid x_t^{[m]}). \tag{1.32}$$

Balans između prečestog i prerijetkog resamplovanja je ključan. Uobičajen pristup uključuje procjenu efikasnog broja čestica (N_{eff}) pomoću varijanse težina:

$$N_{\text{eff}} = \frac{1}{\sum_{m=1}^{M} (w_t^{[m]})^2}. \tag{1.33}$$

Resamplovanje se vrši samo ako je N_{eff} ispod unaprijed definisanog praga, čime se izbjegava nepotrebno gubljenje raznolikosti.

Resamplovanje niske varijanse

Umjesto standardnog (multinomialnog) resamplovanja, može se koristiti metoda niske varijanse, koja smanjuje vjerovatnoću ponovnog izbora istih čestica. Ova metoda sekvencijalno raspoređuje čestice proporcionalno njihovim težinama i znatno smanjuje statističku varijansu. Pseudokod ove metode dat je u Algoritmu 6.

Algoritam 6 Low Variance Resampler

function LOW_VARIANCE_RESAMPLER(χ_t, W_t)
  $\bar{\chi}_t = 0$
  $r = \text{rand}(0, M^{-1})$
  $c = w_t^{[1]}$
  $i = 1$
  for $m = 1$ to M **do**
    $U = r + (m-1) \cdot M^{-1}$
    while $U > c$ **do**
      $i = i + 1$
      $c = c + w_t^{[i]}$
    end while
    $\bar{\chi}_t = \bar{\chi}_t \cup x_t^{[i]}$
  end for
  return $\bar{\chi}_t$
end function

Glava 2

Mjerni i model kretanja

Implementacija SLAM algoritma zahtijeva precizno definisanje načina na koji se robot kreće i prikuplja informacije o okruženju. Ovo poglavlje posvećeno je modelima kretanja (engl. *motion models*) i mjernim modelima (engl. *measurement models*), koji zajedno omogućavaju integraciju podataka neophodnih za pouzdano funkcionisanje SLAM sistema. U nastavku su predstavljeni osnovni principi, matematičke formulacije, kao i praktične prednosti i ograničenja različitih pristupa modelovanju kretanja i mjerenja.

2.1 Kinematički modeli robota

Kinematički model vozila opisuje kako rotacija točkova utiče na njegovo kretanje u prostoru, odnosno na promjenu njegove pozicije i orijentacije. Pored kinematičkog, postoji i dinamički model koji povezuje obrtni moment točkova sa ubrzanjem vozila, ali on neće biti razmatran u ovom radu.

Razmatranja se zasnivaju na pretpostavci da se vozilo kreće po čvrstoj, ravnoj i neklizajućoj površini. Pozicija robota definiše se pomoću tri koordinate:

$$\chi = g(x, y, \theta), \tag{2.1}$$

gdje su x i y položaj robota u ravni, a θ njegova orijentacija u odnosu na globalni koordinatni sistem. Brzina robota opisana je vremenskim izvodima ovih koordinata:

$$v = g(\dot{x}, \dot{y}, \dot{\theta}), \tag{2.2}$$

pri čemu $\dot{x}$ i $\dot{y}$ predstavljaju linearne brzine duž osa x i y, dok $\dot{\theta}$ označava ugaonu brzinu.

Za analizu kretanja robota posebno je važna transformacija između referentnog (globalnog) i lokalnog (vezanog za vozilo) koordinatnog sistema robota. Ova trans-

formacija omogućava da se brzine izražene u lokalnom okviru robota (tzv. *body frame*) transformišu u globalne koordinate. U tu svrhu koristi se tzv. *planarna rotacija šasije*, koja povezuje linearne i ugaone brzine u oba referentna sistema [1].

Brzina izražena u globalnom koordinatnom sistemu dobija se transformacijom iz lokalnog koordinatnog sistema korišćenjem trenutne orijentacije robota θ prema sledećoj transformaciji:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v_{bx} \\ v_{by} \\ \omega \end{bmatrix}. \tag{2.3}$$

Obrnuto, brzine izražene u lokalnom koordinatnom sistemu robota dobijaju se iz globalnih prema:

$$\begin{bmatrix} v_{bx} \\ v_{by} \\ \omega \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix}. \tag{2.4}$$

Gdje su:

- $\dot{x}, \dot{y}$ – linearne komponente brzine u globalnom koordinatnom sistemu,
- $\dot{\theta}$ – ugaona brzina robota oko globalne ose z,
- v_{bx}, v_{by} – linearne komponente brzine u lokalnom (tjelesnom) koordinatnom sistemu,
- ω – ugaona brzina izražena u lokalnom sistemu (ekvivalentna $\dot{\theta}$),
- θ – orijentacija robota u odnosu na globalni koordinatni sistem.

Ova transformacija omogućava jedinstven prikaz kretanja robota u oba referentna sistema[1].

U kinematičkom modelovanju, svaki tip vozila, u skladu sa svojom konstrukcijom i kinematičkim ograničenjima, ima odgovarajući model kretanja. Jedna važna klasifikacija zasniva se na načinu na koji točkovi vozila utiču na njegovo kretanje i mogućnosti manevrisanja. Prema tom kriterijumu, vozila sa točkovima mogu se svrstati u dve osnovne kategorije: *omnidirekciona* i *neholonomska* [1]:

- **Omnidirekciona vozila**[2] nemaju kinematička ograničenja i sposobna su za nesmetano kretanje u svim pravcima. Njihova šasija ne sadrži Pfaffova ograničenja brzine koja bi ograničavala pravac kretanja [36].

[1] Ova transformacija je validna samo kada je rotacija lokalnog koordinatnog sistema definisana oko centra vozila (tj. kada se lokalni sistem poklapa sa tačkom rotacije vozila) [36].

[2] Omni (latinski): prefiks koji znači „svi“ ili „svaki“.

- **Neholonomska vozila**, nasuprot tome, podliježu najmanje jednom Pfaffovom ograničenju brzine, što se matematički izražava formulom:

$$A(\chi)\dot{\chi} = 0. \tag{2.5}$$

U ovoj jednačini:

- $\chi \in \mathbb{R}^n$ predstavlja *vektor konfiguracije* sistema, tj. skup veličina koje definišu položaj i orijentaciju vozila (npr. $\chi = [x, y, \theta]^T$ za vozilo u ravni),
- $\dot{\chi} \in \mathbb{R}^n$ je *vektor brzina*, odnosno vremenske promjene konfiguracionih koordinata,
- $A(\chi) \in \mathbb{R}^{k \times n}$ je *matrica ograničenja* koja može zavisiti od konfiguracije χ, a svaki njen red predstavlja jedno linearno neholonomsko ograničenje.

Kod neholonomskih vozila, kao što su roboti sa diferencijalnim pogonom, ovakva ograničenja onemogućavaju kretanje u određenim pravcima – konkretno, bočno kretanje. I pored toga, ovi sistemi su često *upravljački dostižni*[3], odnosno sposobni da postignu proizvoljnu konfiguraciju u prostoru bez prepreka, uz odgovarajuću sekvencu upravljačkih signala.

U nastavku rada fokusiraćemo se isključivo na neholonomska vozila, dok omnidirekciona neće biti predmet dalje analize.

2.1.1 Modelovanje neholonomskih robota

Svi kinematički modeli neholonomskih robota mogu se zapisati u opštem obliku:

$$\dot{\chi} = f(\chi, u), \tag{2.6}$$

gdje je $\chi \in \mathbb{R}^n$ vektor konfiguracije, a $u \in \mathbb{R}^m$ vektor upravljačkih signala. Tri ključne osobine ovakvih modela su [1]:

- **Bez proklizavanja:** Kada su upravljački signali jednaki nuli, tada su i sve brzine nulte – sistem se ne kreće bez aktivnih ulaza.
- **Vektorska polja zavise od konfiguracije:** Kretanje robota u datom trenutku zavisi od njegove pozicije i orijentacije, odnosno od konfiguracije χ.
- **Linearni po upravljačkom signalu:** Svaki upravljački signal utiče na kretanje robota tako što određuje brzinu duž jednog od osnovnih pravaca kretanja.

[3]Controllably reachable (engleski): pojam iz teorije upravljanja koji označava mogućnost da se sistem iz bilo koje početne konfiguracije dovede u bilo koju ciljnu konfiguraciju korišćenjem validnih upravljačkih komandi.

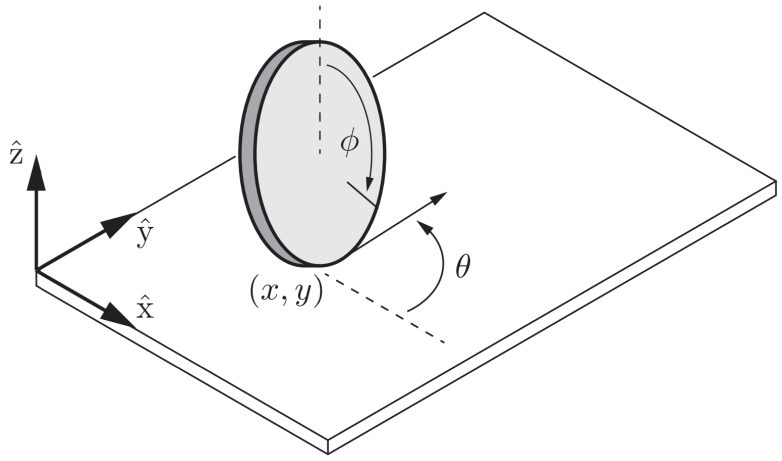
Monocikl

Najjednostavniji robot sa točkovima je robot sa jednim točkom, tzv. *monocikl*. Za poluprečnik točka r, koordinate vozila (x, y), ugao orijentacije θ u odnosu na referentni koordinatni sistem i ugao kotrljanja točka φ (slika 2.1), konfiguracija šasije definiše se sledećom kinematičkom jednačinom:

$$\dot{\chi} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\varphi} \end{bmatrix} = \begin{bmatrix} r\cos(\theta) & 0 \\ r\sin(\theta) & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = G(\chi)u = g_1(\chi)u_1 + g_2(\chi)u_2, \tag{2.7}$$

gdje je u_1 brzina kotrljanja pogonskog točka, a u_2 ugaona brzina vozila oko ose normalne na ravan. Ugao rotacije točka φ nije relevantan za kretanje robota u ravni, pa se može izostaviti iz modela.

$$\dot{\chi} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} r\cos(\theta) & 0 \\ r\sin(\theta) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}. \tag{2.8}$$



Slika 2.1: Točak koji se kotrlja u prostoru [1]

Robot sa diferencijalnim pogonom

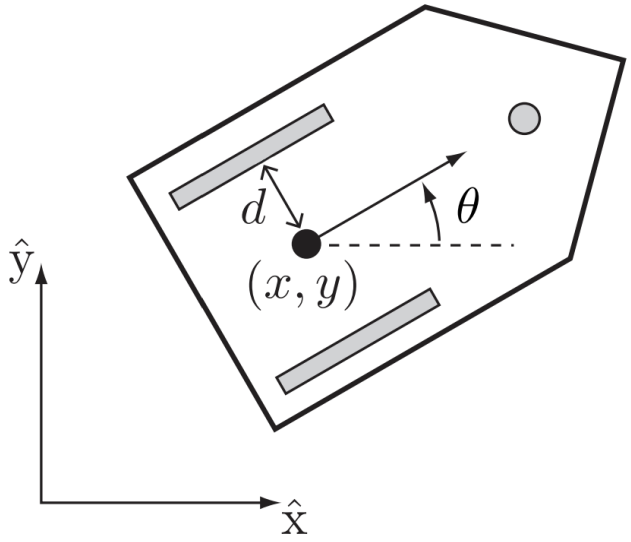
Robot sa diferencijalnim pogonom predstavlja najjednostavniji, ali praktično primjenjiv mobilni robotski sistem. Pogonski mehanizam čine dva točka poluprečnika r, postavljena simetrično sa lijeve i desne strane robota, čije se brzine u_L i u_R mogu nezavisno kontrolisati (slika 2.2).

Kretanje robota ostvaruje se kombinovanjem rotacija lijevog i desnog točka, što omogućava translaciju, rotaciju ili njihovu kombinaciju. Održavanje uspravnog sta-

nja robota obezbjeđuje se statičkom stabilnošću šasije, koja se najčešće postiže dodatnim pomoćnim točkom, poput loptastog oslonca. Alternativno, u nekim robotima koristi se dinamički mehanizam za održavanje ravnoteže.

Neka je razmak između točkova označen kao $2d$, a konfiguracija robota data vektorom $\chi = [x, y, \theta]^T$, kinematički model bez uglova kotrljanja točkova (koji nisu relevantni za kretanje u ravni) glasi:

$$\dot{\chi} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \frac{r}{2}\cos\theta & \frac{r}{2}\cos\theta \\ \frac{r}{2}\sin\theta & \frac{r}{2}\sin\theta \\ \frac{-r}{2d} & \frac{r}{2d} \end{bmatrix} \begin{bmatrix} u_L \\ u_R \end{bmatrix}. \tag{2.9}$$



Slika 2.2: Robot sa diferencijalnim pogonom koji se sastoji od dva pogonska točka i jednog pomoćnog točka (osjenčenog sivom bojom) [1].

Robot sa automobilskom kinematikom

Mobilni roboti koji imitiraju kretanje automobila koriste se u situacijama gde je potrebno modelovanje bliže stvarnom ponašanju vozila, za razliku od idealizovanih modela poput monocikla ili diferencijalnog pogona. Tipično se sastoje od dvije osovine: zadnje (fiksne), i prednje (upravljive), sa po dva točka na krajevima. Njihovo kretanje opisuje se tzv. Ackermann-ovim modelom (slika 2.3), kod koga se centar zaokretanja vozila nalazi na presjeku prave koja prolazi kroz zadnju osovinu i pravih koje su normalne na pravac prednjih točkova.

Konfiguracija vozila opisana je vektorom $\chi = [x, y, \theta, \psi]^T$, gde su:

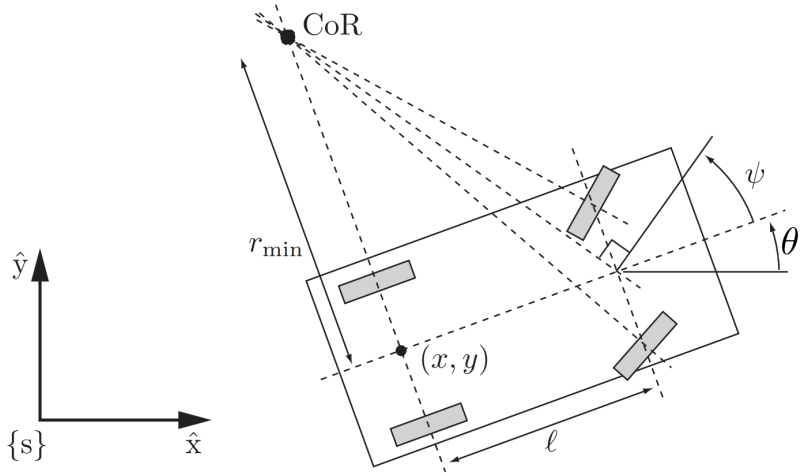
- (x, y) – koordinate centra zadnje osovine,
- θ – orijentacija (pravac kretanja vozila),

- ψ – ugao skretanja prednjih točkova u odnosu na šasiju.

Kinematički model tada ima oblik:

$$\dot{\chi} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} \cos(\phi) & 0 \\ \sin(\phi) & 0 \\ \frac{\tan(\psi)}{\ell} & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}. \tag{2.10}$$

gde je ℓ razdaljina između prednje i zadnje osovine.



Slika 2.3: Ackermann-ovo upravljanje kod robota nalik automobilu [1].

Normalizacija upravljačkih signala

Radi konzistentnog prikaza različitih modela kretanja, upravljački signali se izražavaju kroz standardizovani par: linearna brzina v i ugaona brzina ω, pri čemu $\omega = \dot{\theta}$ označava brzinu promjene orijentacije robota. Na osnovu ovih veličina, upravljački signali u_1 i u_2 za različite modele mogu se transformisati na sljedeći način:

- **Monocikl:** Upravljački signali izraženi su kao:

$$u_1 = \frac{v}{r}, \quad u_2 = \omega. \tag{2.11}$$

- **Diferencijalni pogon:** Upravljački signali za lijevi i desni točak izraženi su u funkciji linearne i ugaone brzine kao:

$$u_L = \frac{v - \omega d}{r}, \quad u_R = \frac{v + \omega d}{r}. \tag{2.12}$$

Obrnuto, ako su poznate brzine točkova, linearna i ugaona brzina robota do-

bijaju se kao:

$$\begin{aligned} v &= \frac{r}{2}(u_L + u_R), \\ \omega &= \frac{r}{2d}(u_R - u_L). \end{aligned} \tag{2.13}$$

- **Robot sa automobilskom kinematikom:** Upravljački signali se direktno izražavaju kao linearna i ugaona brzina:

$$u_1 = v, \quad u_2 = \omega. \tag{2.14}$$

Ograničenja upravljačkih signala u_1 i u_2 razlikuju se u zavisnosti od tipa robota – na primer, kod robota sa automobilskom kinematikom, često postoji ograničenje maksimalnog ugla skretanja točkova.

2.1.2 Diskretni kinematički model konstantne brzine

U praktičnim SLAM algoritmima, kinematički modeli konstantne brzine se često koriste zbog jednostavnosti implementacije i dobrih performansi u uslovima gdje se upravljački signali ne mijenjaju drastično između uzastopnih vremenskih koraka. Ovakvi modeli pružaju razuman kompromis između preciznosti i složenosti izračunavanja, omogućavajući efikasnu estimaciju stanja robota uz minimalne računske zahtjeve.

U daljoj analizi koristiće se diskretni model kretanja sa konstantnom linearnom brzinom v i promenljivom ugaonom brzinom ω. Iako robot koristi diferencijalni pogon, njegovo kretanje može se dobro aproksimirati idealizovanim monocikličkim modelom u kojem su v i ω upravljački ulazi. [10, 36]

Položaj i orijentacija monocila opisuju se sljedećim jednačinama:

$$\dot{x} = v\cos\theta, \qquad \dot{y} = v\sin\theta, \qquad \dot{\theta} = \omega. \tag{2.15}$$

Pretpostavljamo da su upravljački ulazi v i ω konstantni tokom malog vremenskog intervala Δt. Najprije rješavamo jednačinu za ugao:

$$\dot{\theta} = \omega \implies \theta(t) = \theta_0 + \omega t, \tag{2.16}$$

što znači da nakon vremena Δt nova orijentacija robota iznosi:

$$\theta_{t+\Delta t} = \theta_t + \omega\Delta t. \tag{2.17}$$

Za određivanje nove pozicije, potrebno je integrisati $\dot{x}$ i $\dot{y}$ tokom intervala Δt, uzi-

majući u obzir da se θ tokom kretanja mijenja:

$$x_{t+\Delta t} = x_t + \int_0^{\Delta t} v \cos\left(\theta_t + \omega\tau\right) d\tau, \tag{2.18}$$

$$y_{t+\Delta t} = y_t + \int_0^{\Delta t} v \sin\left(\theta_t + \omega\tau\right) d\tau, \tag{2.19}$$

gdje je τ promjenljiva integracije.

Rješavanjem ovih integrala dobijamo:

$$\int_0^{\Delta t} \cos\left(\theta_t + \omega\tau\right) d\tau = \frac{1}{\omega}\left[\sin\left(\theta_t + \omega\tau\right)\right]_0^{\Delta t} = \frac{1}{\omega}\left(\sin(\theta_t + \omega\Delta t) - \sin\theta_t\right), \tag{2.20}$$

$$\int_0^{\Delta t} \sin\left(\theta_t + \omega\tau\right) d\tau = -\frac{1}{\omega}\left[\cos\left(\theta_t + \omega\tau\right)\right]_0^{\Delta t} = -\frac{1}{\omega}\left(\cos(\theta_t + \omega\Delta t) - \cos\theta_t\right). \tag{2.21}$$

Uvrštavanjem rezultata integracije dobijamo diskretne formule za novu poziciju:

$$x_{t+\Delta t} = x_t + \frac{v}{\omega}\left(\sin(\theta_t + \omega\Delta t) - \sin\theta_t\right), \tag{2.22}$$

$$y_{t+\Delta t} = y_t - \frac{v}{\omega}\left(\cos(\theta_t + \omega\Delta t) - \cos\theta_t\right). \tag{2.23}$$

Na ovaj način, dobijamo egzaktne diskretne formule koje povezuju početno stanje robota i upravljačke ulaze tokom malog vremenskog koraka. Razlaganjem ovih izraza dobijamo:

$$x_{t+\Delta t} = x_t - \frac{v}{\omega}\sin\theta_t + \frac{v}{\omega}\sin(\theta_t + \omega\Delta t), \tag{2.24}$$

$$y_{t+\Delta t} = y_t + \frac{v}{\omega}\cos\theta_t - \frac{v}{\omega}\cos(\theta_t + \omega\Delta t), \tag{2.25}$$

$$\theta_{t+\Delta t} = \theta_t + \omega\Delta t, \tag{2.26}$$

ili, ekvivalentno, u matričnoj formi direktno kao funkcija početne pozicije (x, y, θ)[4]:

$$\begin{bmatrix} x \\ y \\ \theta \end{bmatrix}_{t+\Delta t} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} -\frac{v}{\omega}\sin\theta + \frac{v}{\omega}\sin(\theta + \omega\Delta t) \\ \frac{v}{\omega}\cos\theta - \frac{v}{\omega}\cos(\theta + \omega\Delta t) \\ \omega\Delta t \end{bmatrix}. \tag{2.27}$$

U praksi, roboti ne mogu trenutno prelaziti sa jedne brzine na drugu, niti mogu održavati savršeno konstantnu brzinu tokom cijelog vremena. Zbog toga se kretanje u realnim uslovima najčešće modeluje tako što se vremenska osa dijeli na male intervale Δt, unutar kojih se brzine aproksimiraju kao konstantne [10].

[4]Isti oblik diskretnih jednačina dobija se i za diferencijalni pogon kada se centar kretanja uzima na sredini osovine, odnosno kada se koriste v i ω kao upravljački ulazi.

2.2 Mjerni model

Za formiranje mape okruženja neophodna su relativna mjerenja u odnosu na objekte u prostoru. Senzori koji omogućavaju ovu funkcionalnost klasifikuju se prema načinu na koji prikupljaju informacije o okolini, a to uključuje: senzore udaljenosti i pravca, laserske senzore (LIDAR), ultrazvučne senzore (sonar), stereo kamere, RGB-D kamere i druge.

Mjerni modeli definišu kako senzor generiše podatke iz okruženja. Iako se svi prethodno navedeni senzori koriste u SLAM-u, u ovom radu fokusiraćemo se na senzore udaljenosti i pravca (engl. *range and bearing* senzore), koji lociraju tačkasta obilježja (engl. *landmarks*) u prostoru u odnosu na poziciju robota.

Budući da se ovakva mjerenja odnose na tačke u prostoru, pretpostavlja se da su objekti iz okruženja već redukovani na tačkasta obilježja. Ta redukcija može se izvesti različitim tehnikama, uključujući detekciju ivica, segmentaciju oblaka tačaka, RANSAC detekciju geometrijskih oblika (npr. linije, ravni), kao i ekstrakciju ključnih tačaka pomoću SIFT ili SURF algoritama. Ipak, detaljno razmatranje ovih metoda prevazilazi okvire ovog rada; u nastavku se pretpostavlja da su obilježja već dostupna u odgovarajućem obliku.

Ako označimo udaljenost do obilježja kao r, a ugao posmatranja kao ϕ, tada se svako pojedinačno mjerenje f_t^i u trenutku t zapisuje kao:

$$f(z_t) = \left\{ f_t^1, f_t^2, \dots \right\} = \left\{ \begin{bmatrix} r_t^1 \\ \phi_t^1 \end{bmatrix}, \begin{bmatrix} r_t^2 \\ \phi_t^2 \end{bmatrix}, \dots \right\}, \tag{2.28}$$

gdje je:

- $z_t = \{z_t^1, z_t^2, \dots, z_t^{n_t}\}$, skup svih pojedinačnih mjerenja u trenutku t, gdje je n_t broj mjerenja za dato t,
- $f_t^i = \begin{bmatrix} r_t^i \\ \phi_t^i \end{bmatrix}$ vektor mjerenja za i-to obilježje, gdje r_t^i predstavlja izmjerenu udaljenost, a ϕ_t^i ugao mjerenja u odnosu na pravac robota.

Broj opažanja u svakom vremenskom koraku može da varira, u zavisnosti od broja detektovanih obilježja. Pod pretpostavkom da su poznati položaj robota x_t i mapa m, opažanja se smatraju međusobno nezavisnim [10]. Ovo se može zapisati kao:

$$p(f(z_t) \mid x_t, m) = \prod_i p(r_t^i, \phi_t^i \mid x_t, m). \tag{2.29}$$

Ova pretpostavka vrijedi samo ako su šumovi pojedinačnih mjerenja međusobno nezavisni, što se u ovom radu uzima kao važeće. Dodatno, pretpostavlja se da ti

šumovi prate Gaussovu (normalnu) raspodjelu.

Mjerenje i-tog obilježja u trenutku t, koje odgovara j-tom obilježju u mapi, modeluje se geometrijski kao:

$$\begin{bmatrix} r_t^i \\ \phi_t^i \end{bmatrix} = \begin{bmatrix} \sqrt{(m_{j,x} - x)^2 + (m_{j,y} - y)^2} \\ \text{atan2}(m_{j,y} - y, m_{j,x} - x) - \theta \end{bmatrix} + \begin{bmatrix} \varepsilon_r \\ \varepsilon_\phi \end{bmatrix}, \tag{2.30}$$

gdje su $m_{j,x}$ i $m_{j,y}$ globalne koordinate j-tog obilježja u mapi, dok su ε_r i ε_ϕ Gaussove greške nulte srednje vrijednosti i standardnih devijacija σ_r i σ_ϕ, redom. Ove greške predstavljaju mjerni šum senzora i uzimaju se kao nezavisni aditivni šumovi.

Glava 3

Simultana lokalizacija i mapiranje

3.1 Uvod u SLAM

Simultana lokalizacija i mapiranje (engl. *Simultaneous Localization and Mapping* – SLAM) opisuje problem u kojem robot istovremeno gradi mapu svog okruženja i procjenjuje svoju poziciju unutar te mape. Za razliku od klasičnih problema lokalizacije ili mapiranja, u kojima je dostupna neka referenca (npr. poznata mapa ili precizna pozicija robota), SLAM se suočava s dodatnim izazovom: i okruženje i pozicija robota u početku su nepoznati. Zbog toga greška u procjeni jedne varijable direktno utiče na drugu, što ovaj proces čini međuzavisnim [10, 37].

Problem SLAM-a formalno se definiše ulazima:

- $z_{1:t}$ – sekvencom mjerenja senzora,
- $u_{1:t}$ – nizom upravljačkih signala (komandi kretanja),

i izlazima:

- m – mapom okruženja,
- $x_{1:t}$ – pozicijama robota u toj mapi.

Mapa okruženja može biti predstavljena na različite načine: kao metrička mapa, kao skup obilježja (*landmarks*), kroz mrežasti prikaz (engl. *occupancy grid*) ili kao hibridna kombinacija navedenih. U okviru ovog rada razmatra se samo mapa zasnovana na obilježjima.

Algoritam za prepoznavanje već posjećenih lokacija, poznat kao *loop closure*[1], neće biti analiziran, budući da se fokus stavlja na tzv. *online* varijantu SLAM-a. Vrijedno je napomenuti da se efekti zatvaranja kruga kod algoritama sa poznatom asocijacijom obilježja često javljaju implicitno, ali nisu dio eksplicitnog modelovanja u ovom radu [10].

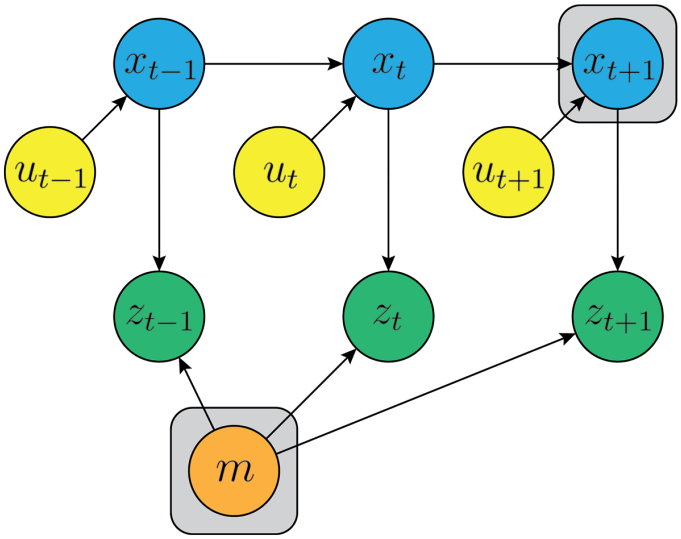
[1] Loop closure (engleski): zatvaranje kruga

3.1.1 Online SLAM i Full (Offline) SLAM

Iz perspektive izlaznih informacija, SLAM se može podijeliti na dva osnovna tipa. Prvi je *online SLAM*, koji se fokusira na procjenu trenutne pozicije robota i istovremeno mapiranje okruženja. Ovaj pristup se implementira inkrementalno – nakon svake obrade mjerenja i upravljačkih signala, prethodni podaci se odbacuju, čime se smanjuju memorijski zahtjevi i omogućava efikasna obrada u realnom vremenu. Ova ideja inkrementalne obrade odražena je u dinamičkom Bayesovom modelu, gdje se u svakoj iteraciji koriste samo trenutno mjerenje i upravljački signal, dok se prethodni podaci zanemaruju. Matematički, ovaj oblik se zapisuje:

$$p(x_t, m \mid z_{1:t}, u_{1:t}), \tag{3.1}$$

gdje x_t označava trenutnu poziciju robota u vremenskom trenutku t, $u_{1:t}$ i $z_{1:t}$ predstavljaju niz upravljačkih signala i senzorskih mjerenja do tog trenutka, dok m označava mapu okruženja koju robot paralelno procjenjuje [10]. Struktura zavisnosti između ovih varijabli prikazana je na slici 3.1 u obliku dinamičke Bayesove mreže.



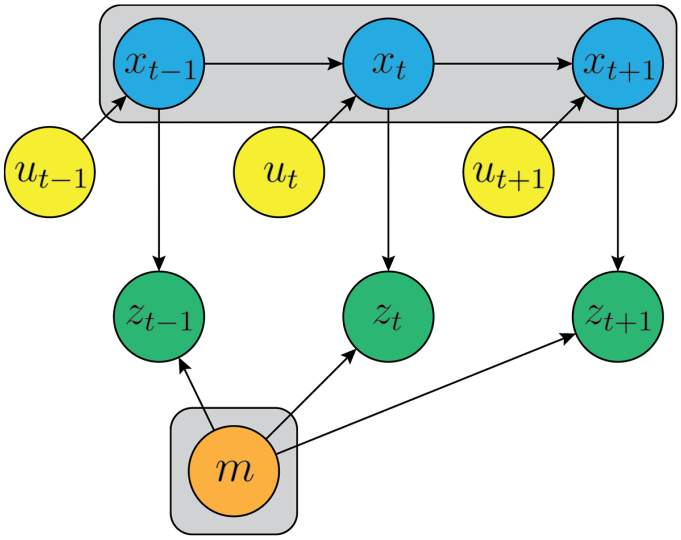
Slika 3.1: Grafički model online SLAM problema

Drugi oblik je *full SLAM* (ili *offline SLAM*), koji procjenjuje mapu zajedno sa cijelom trajektorijom robota. Za razliku od online SLAM-a, koji iz svih prethodnih mjerenja i upravljačkih komandi procjenjuje samo trenutnu poziciju x_t i zatim prethodne podatke odbacuje, full SLAM zadržava sve podatke do trenutka t i koristi ih za simultanu procjenu svih pozicija $x_{0:t}$ od početka kretanja do trenutka t. Matematički, zapisuje se kao:

$$p(x_{0:t}, m \mid z_{1:t}, u_{1:t}), \tag{3.2}$$

gdje $x_{0:t}$ označava sve pozicije robota od početne pozicije x_0 do trenutka t, dok m

predstavlja mapu okruženja. Uključivanje kompletne sekvence pozicija omogućava formulisanje jedinstvenog (jednoznačnog) rješenja problema SLAM-a (slika 3.2) [10].



Slika 3.2: Grafički model offline SLAM problema

U suštini, online SLAM se može posmatrati kao poseban slučaj full SLAM-a, pri čemu se umjesto procjene cijele trajektorije robota $x_{0:t}$ procjenjuje samo njegova trenutna pozicija x_t, zajedno s mapom m (tj. prethodne pozicije se implicitno uklanjaju – *marginalizuju*). Ova veza formalno je prikazana izrazom:

$$p(x_t, m \mid z_{1:t}, u_{1:t}) = \int \cdots \int p(x_{1:t}, m \mid z_{1:t}, u_{1:t})\, dx_1 \cdots dx_{t-1}. \tag{3.3}$$

Za razliku od full SLAM-a, koji zadržava sve prethodne informacije i omogućava naknadnu korekciju procjena – čime se problem može formulirati kao dobro postavljen (*well-posed*) sistem sa stabilnijim ishodom – online SLAM odbacuje prethodne podatke nakon obrade. Time žrtvuje dio preciznosti u korist efikasnosti i nižih računskih zahtjeva.

3.1.2 Poznata i nepoznata asocijacija

Kada robot detektuje obilježje u svom okruženju, SLAM algoritam mora utvrditi da li ono odgovara nekom prethodno viđenom obilježju ili se radi o novom obilježju. Ovo utvrđivanje je obično binarno: obilježje ili jeste ili nije isto kao neko ranije viđeno. Proces određivanja asocijacije između novog mjerenja i postojećih obilježja na mapi izražava se diskretnom promjenljivom. Ukoliko ovu promjenljivu označimo kao c_t, posteriorna raspodjela koju online SLAM procjenjuje može se zapisati kao:

$$p(x_t, m, c_t \mid z_{1:t}, u_{1:t}), \tag{3.4}$$

dok full SLAM procjenjuje:

$$p(x_{0:t}, m, c_{1:t} \mid z_{1:t}, u_{1:t}). \tag{3.5}$$

Rješenje online SLAM-a dobija se *marginalizacijom* prethodnih pozicija (integracijom preko svih njihovih mogućih vrijednosti) i sumiranjem prethodnih asocijacija iz zajedničke raspodjele:

$$p(x_t, m, c_t \mid z_{1:t}, u_{1:t}) = \int \cdots \int \sum_{c_1} \cdots \sum_{c_{t-1}} p(x_{1:t}, m, c_{1:t} \mid z_{1:t}, u_{1:t})\, dx_1 \ldots dx_{t-1}. \tag{3.6}$$

U praksi je, međutim, potpuna obrada svih pozicija i asocijacija često neizvodljiva zbog velike dimenzionalnosti prostora stanja i brzog rasta broja mogućih kombinacija asocijacija obilježja tokom vremena. Stoga se praktični SLAM algoritmi oslanjaju na aproksimativne metode određivanja asocijacije [10]. Metode asocijacije podataka (*data association*) uključuju Nearest-Neighbor (NN), Gated Nearest-Neighbor (GNN), Joint Compatibility Branch and Bound (JCBB), Maximum Likelihood (ML), Incremental Maximum Likelihood (IML), Probabilistic Data Association (PDA), Multi-Hypothesis Tracking (MHT) itd. U ovom radu osvrnućemo se na Gated Nearest-Neighbor (GNN) metodu, koja se odlikuje jednostavnošću i agnostičnošću u odnosu na konkretan SLAM algoritam.

GNN identifikuje najvjerovatniju asocijaciju tako što računa Mahalanobisovu udaljenost između mjerenja senzora i predviđenih mjerenja postojećih obilježja [38]. Ta udaljenost između stvarnog mjerenja z i njegovog predviđenog ekvivalenta $\hat{z}$ izračunava se formulom:

$$d = \sqrt{(z - \hat{z})^T S^{-1} (z - \hat{z})}, \tag{3.7}$$

gdje je S matrica kovarijanse inovacije koja uključuje nesigurnosti u poziciji robota i obilježja.

Mjerenja unutar definisanog praga (validacione kapije) smatraju se potencijalnim asocijacijama. GNN bira asocijaciju koja minimizira Mahalanobisovu udaljenost ili, ekvivalentno, maksimizira vjerovatnoću asocijacije:

$$\Lambda_i = \frac{1}{(2\pi)^{\frac{n}{2}} \sqrt{|S_i|}} \exp\left(-\frac{1}{2} \upsilon_i^T S_i^{-1} \upsilon_i\right), \tag{3.8}$$

odnosno, u logaritamskom obliku:

$$N_i = \upsilon_i^T S_i^{-1} \upsilon_i + \ln(|S_i|). \tag{3.9}$$

Ukoliko nijedno obilježje ne zadovoljava kriterijum kapije, mjerenje se smatra novim i dodaje se mapi. Pseudokod GNN algoritma prikazan je u Algoritmu 7.

Funkcije `predict_observation` i `compute_jacobian` predstavljaju komponente koje zavise od konkretne implementacije SLAM algoritma, budući da svaki algoritam definiše sopstveni način predikcije mjerenja i izračunavanja Jacobijevih matrica. Simboli z_r, c i z_n označavaju redom: mjerenja pridružena postojećim obilježjima, vektor asocijacija i vektor novih obilježja koja će biti dodata mapi.

Algoritam 7 Gated Nearest–Neighbor (GNN) asocijacija podataka

function GNN_ASSOCIATION($\bar{\mu}, \bar{\Sigma}, z, Q,$ gate)
  $z_f = [\,]$
  $c = [\,]$
  $z_n = [\,]$
  for each observation z_i in z **do**
    $j_{\text{best}} =$ None
    $d_{\min} = \infty$
    for each landmark j **do**
      $\hat{z}_j =$ predict_observation$(\bar{\mu}, j)$
      $\Delta = z_i - \hat{z}_j$
      $H_j =$ calculate_jacobian(x, j)
      $S = H_j \bar{P} H_j^T + Q$
      $d = \Delta^T S^{-1} \Delta$
      if $d <$ gate and $d < d_{\min}$ **then**
        $d_{\min} = d$
        $j_{\text{best}} = j$
      end if
    end for
    if $j_{\text{best}} \neq$ None **then**
      Append z_i to z_f
      Append j_{best} to c
    else
      Append z_i to z_n
    end if
  end for
  return z_f, c, z_n
end function

Uspješnost GNN metode uveliko zavisi od odnosa između mjerne nesigurnosti i složenosti okruženja. Kada je broj pogrešnih opservacija (poput šumova ili refleksija) relativno mali u poređenju sa brojem stvarnih obilježja i kada su mjerne nesigurnosti dovoljno velike da pravilno obuhvate ta obilježja, GNN daje pouzdane rezultate. Međutim, u gustim ili vizuelno složenim okruženjima, gdje postoji veliki

broj potencijalno pogrešnih mjerenja, performanse GNN-a mogu znatno opasti zbog povećane vjerovatnoće pogrešne asocijacije. U takvim slučajevima, GNN efektivno koristi stroži prag za validaciju – prihvata samo mjerenja koja se veoma dobro poklapaju s očekivanjima, čime se povećava tolerancija na greške, ali po cijenu gubitka potencijalno korisnih asocijacija.

3.2 Kalman SLAM

U ovom poglavlju razmatraju se SLAM algoritmi zasnovani na Kalmanovom filteru, konkretno EKF i UKF varijante. Kalman SLAM primjenjuje princip filtriranja na problem *online* SLAM-a, uz asocijaciju podataka zasnovanu na maksimalnoj vjerovatnoći. Ovakav pristup, međutim, podrazumijeva niz aproksimacija i ograničavajućih pretpostavki [10]:

- Mape su predstavljene kao skup jednoznačno definisanih, međusobno nezavisnih tačkastih obilježja. Ako obilježja ne ispunjavaju ove uslove, neophodna je dodatna obrada, poput redukcije složenih objekata u tačkaste reprezentacije.
- Broj obilježja mora biti ograničen (tipično ispod 1000), jer svako novo obilježje povećava dimenzionalnost matrica koje Kalmanov filter održava, što značajno utiče na memorijske i računske zahtjeve.
- Pretpostavlja se da su nesigurnosti u kretanju robota i mjerenjima senzora modelovane kao Gaussovi šumovi. Linearizacija modela u EKF-u daje dobre rezultate samo uz relativno malu nesigurnost u poziciji robota; u suprotnom, greške mogu postati značajne. Ova pretpostavka se ne odnosi na UKF, koji koristi nelinearnu propagaciju sigma tačaka.
- Kalman SLAM ne uključuje mehanizme za uklanjanje obilježja niti za korekciju pogrešnih asocijacija. Pogrešna asocijacija može dovesti do trajne degradacije u procjeni mape i pozicije robota.

3.2.1 Prošireni Kalmanov filter za SLAM

EKF-SLAM simultano procjenjuje poziciju robota i pozicije obilježja u prostoru, oslanjajući se na zajednički vektor stanja dimenzije $3 + 2n$, gdje n označava broj obilježja [10, 34, 39, 40]. Reprezentacija vektora stanja ima sljedeći oblik:

$$\mu_t = \begin{pmatrix} x_t \\ m \end{pmatrix} = \Big(\underbrace{x, y, \theta}_{\text{pozicija robota}} , \underbrace{m_{1,x}, m_{1,y}}_{\text{obilježje 1}}, \ \ldots, \ \underbrace{m_{n,x}, m_{n,y}}_{\text{obilježje n}} \Big)^T. \tag{3.10}$$

Ovdje x, y, θ opisuju poziciju robota u dvodimenzionalnom prostoru (dvije koordinate i jedna rotaciona komponenta), dok $m_{j,x}, m_{j,y}$ predstavljaju koordinate pojedi-

načnih obilježja. U slučaju trodimenzionalnog prostora, vektor konfiguracije robota ima šest komponenti (tri translacije i tri rotacije), a svako obilježje se opisuje sa tri koordinate. Ova struktura implicira ograničenje EKF SLAM-a: dimenzija vektora stanja linearno raste sa brojem obilježja, što dovodi do povećanih memorijskih zahtjeva i većeg računskog opterećenja prilikom ažuriranja i propagacije stanja.

Matrica kovarijanse Σ u EKF SLAM-u obuhvata nesigurnosti u poziciji robota, položajima obilježja, kao i njihove međusobne korelacije. Za prostor sa n obilježja, Σ je matrica dimenzije $(3+2n) \times (3+2n)$, pri čemu su prvi redovi i kolone povezani sa pozicijom robota, dok preostali dijelovi opisuju varijanse i kovarijanse obilježja i njihove zavisnosti. Njena struktura može se kompaktno predstaviti kao:

$$\Sigma = \begin{pmatrix} \Sigma_{xx} & \Sigma_{xm} \\ \Sigma_{mx} & \Sigma_{mm} \end{pmatrix}, \tag{3.11}$$

gdje:

- Σ_{xx} predstavlja kovarijansu pozicije robota,
- Σ_{mm} kovarijanse obilježja,
- Σ_{xm} i Σ_{mx} međusobne kovarijanse između robota i obilježja.

Inicijalizacija

Na početku, robot se postavlja u koordinatni početak mape, a ova pozicija se smatra apsolutno sigurnom. Zbog toga se vektor pozicije i matrica kovarijanse robota inicijalizuju na sljedeći način:

$$\mu_x = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \quad \Sigma_{xx} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{3.12}$$

Kros-korelacione matrice kovarijanse se inicijalizuju nulama:

$$\Sigma_{xm} = \begin{bmatrix} 0 & \cdots & 0 \\ 0 & \cdots & 0 \\ 0 & \cdots & 0 \end{bmatrix}, \quad \Sigma_{mx} = \begin{bmatrix} 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 0 \end{bmatrix}. \tag{3.13}$$

Auto-kovarijaciona matrica obilježja inicijalizuje se kao blok dijagonalna matrica sa vrijednostima beskonačno na glavnoj dijagonali:

$$L = [0\ 0\ \cdots\ 0]^T, \quad \Sigma = \begin{bmatrix} \infty & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \infty \end{bmatrix}. \tag{3.14}$$

Ovo označava potpunu neizvjesnost u vezi sa pozicijama obilježja. Ako se koristi numerička aproksimacija beskonačnosti, preporučuje se konkretna brojčana vrijednost (npr. 10^6 ili veća), koja u implementaciji simulira beskonačnost.

Pošto obilježja u početku nisu poznata (ni njihov broj, ni zavisnost od pozicije robota), njihove dimenzije nisu određene, osim za auto-korelacionu matricu kovarijanse robota. Sa uočavanjem novih obilježja, njima se dodeljuju kovarijanse, a vektor stanja i matrica kovarijanse se proširuju. Ako je broj obilježja poznat unaprijed, moguće je u potpunosti inicijalizovati vektor stanja i pripadajuće matrice, čime se izbjegava dinamičko proširivanje i preračunavanje tokom izvršavanja.

Predikcioni korak

U Algoritmu 2, linije 2 i 3 implementiraju predikcioni korak EKF-a.

$$\bar{\mu}_t = g(u_t, \mu_{t-1}), \tag{3.15}$$

$$\bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R_t. \tag{3.16}$$

Najprije se vrši predikcija narednog vektora stanja na osnovu modela kretanja robota zasnovanog na upravljačkom signalu u_t i prethodnoj procjeni μ_{t-1}. Ova funkcija g predstavlja nelinearni model kretanja u obliku:

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \underbrace{\begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} -\frac{v_t}{\omega_t}\sin(\theta) + \frac{v_t}{\omega_t}\sin(\theta + \omega_t \Delta t) \\ \frac{v_t}{\omega_t}\cos(\theta) - \frac{v_t}{\omega_t}\cos(\theta + \omega_t \Delta t) \\ \omega_t \Delta t \end{bmatrix}}_{g_{x,y,\theta}(u_t, (x,y,\theta)^T)}. \tag{3.17}$$

Pošto ova funkcija utiče samo na prve tri dimenzije vektora stanja (poziciju robota), koristi se proširena matrica:

$$F_x^T = \begin{pmatrix} I_{3\times 3} & 0_{3\times 2n} \end{pmatrix}^T. \tag{3.18}$$

Na osnovu toga, nova procjena dobija se kao:

$$\bar{\mu}_t = \mu_{t-1} + F_x^T \cdot \begin{bmatrix} -\frac{v_t}{\omega_t}\sin(\theta) + \frac{v_t}{\omega_t}\sin(\theta + \omega_t \Delta t) \\ \frac{v_t}{\omega_t}\cos(\theta) - \frac{v_t}{\omega_t}\cos(\theta + \omega_t \Delta t) \\ \omega_t \Delta t \end{bmatrix} + \mathcal{N}(0, F_x^T R_t F_x). \tag{3.19}$$

Nelinearna funkcija g aproksimira se prvim članom Tejlorovog razvoja oko tačke μ_{t-1}:

$$g(u_t, x_{t-1}) \approx g(u_t, \mu_{t-1}) + G_t(x_{t-1} - \mu_{t-1}), \tag{3.20}$$

gdje je G_t Jakobijan funkcije kretanja po poziciji, dimenzije $(3 + 2n) \times (3 + 2n)$, i

ima oblik:

$$G_t = \begin{pmatrix} G_t^x & 0 \\ 0 & I \end{pmatrix}. \tag{3.21}$$

Matrica G_t^x predstavlja Jakobijan funkcije kretanja po x, y, θ, i izračunava se kao:

$$\begin{aligned} G_t^x &= \frac{\partial}{\partial(x,y,\theta)^T} \left[\begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} -\frac{v_t}{\omega_t}\sin(\theta) + \frac{v_t}{\omega_t}\sin(\theta + \omega_t \Delta t) \\ \frac{v_t}{\omega_t}\cos(\theta) - \frac{v_t}{\omega_t}\cos(\theta + \omega_t \Delta t) \\ \omega_t \Delta t \end{bmatrix} \right] \\ &= \begin{bmatrix} 1 & 0 & -\frac{v_t}{\omega_t}\cos(\theta) + \frac{v_t}{\omega_t}\cos(\theta + \omega_t \Delta t) \\ 0 & 1 & -\frac{v_t}{\omega_t}\sin(\theta) + \frac{v_t}{\omega_t}\sin(\theta + \omega_t \Delta t) \\ 0 & 0 & 1 \end{bmatrix}. \end{aligned} \tag{3.22}$$

Na kraju, kovarijansa predikcije se ažurira prema:

$$\bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R_t, \tag{3.23}$$

gdje je R_t matrica kovarijanse odometrijskog šuma. Eksplicitno, koristeći strukturu matrice stanja, ovo se može proširiti kao:

$$\bar{\Sigma}_t = \begin{pmatrix} G_t^x & 0 \\ 0 & I \end{pmatrix} \begin{pmatrix} \Sigma_{xx} & \Sigma_{xm} \\ \Sigma_{mx} & \Sigma_{mm} \end{pmatrix} \begin{pmatrix} G_t^{xT} & 0 \\ 0 & I \end{pmatrix} + R_t, \tag{3.24}$$

ili

$$\bar{\Sigma}_t = \begin{pmatrix} G_t^x \Sigma_{xx} (G_t^x)^T & G_t^x \Sigma_{xm} \\ (\Sigma_{xm})^T (G_t^x)^T & \Sigma_{mm} \end{pmatrix} + R_t, \tag{3.25}$$

gdje su:

- Σ_{xx}: kovarijansa robota,
- Σ_{mm}: kovarijansa obilježja,
- Σ_{xm}, Σ_{mx}: kros-kovarijanse između pozicije robota i obilježja.

Napomena: Zbog rijetke strukture ovih matrica, moguće je optimizovati računanje koristeći sparse linearne alate u numeričkoj implementaciji.

Korekcioni korak

Drugi dio EKF algoritma, tačnije linije 4 do 6 iz Algoritma 2, predstavljaju korekcioni korak. Ova faza koristi nova senzorska mjerenja kako bi se korigovala predikcija dobijena u prethodnom koraku.

U trenutku t, robot opaža i-to obilježje i registruje mjerenje:

$$z_t^{[i]} = \begin{pmatrix} r_t^{[i]} \\ \phi_t^{[i]} \end{pmatrix}, \tag{3.26}$$

gdje $r_t^{[i]}$ predstavlja udaljenost, a $\phi_t^{[i]}$ ugao obilježja u odnosu na pravac robota. Na osnovu ovog mjerenja, pozicija obilježja u mapi može se procijeniti inverzijom funkcije mjernog modela h^{-1}, što rezultira:

$$\begin{pmatrix} \bar{\mu}_{j,x} \\ \bar{\mu}_{j,y} \end{pmatrix} = \begin{pmatrix} \bar{\mu}_{t,x} \\ \bar{\mu}_{t,y} \end{pmatrix} + \begin{pmatrix} r_t^{[i]} \cos(\phi_t^{[i]} + \bar{\mu}_{t,\theta}) \\ r_t^{[i]} \sin(\phi_t^{[i]} + \bar{\mu}_{t,\theta}) \end{pmatrix}. \tag{3.27}$$

Međutim, kako EKF koristi poređenje između stvarnog i očekivanog mjerenja, potrebno je izračunati i očekivanu opservaciju:

$$\delta = \begin{pmatrix} \delta_x \\ \delta_y \end{pmatrix} = \begin{pmatrix} \bar{\mu}_{j,x} - \bar{\mu}_{t,x} \\ \bar{\mu}_{j,y} - \bar{\mu}_{t,y} \end{pmatrix}, \qquad q = \delta^T \delta, \tag{3.28}$$

$$\hat{z}_t^{[i]} = h(\bar{\mu}_t) = \begin{pmatrix} \sqrt{q} \\ \text{atan2}(\delta_y, \delta_x) - \bar{\mu}_{t,\theta} \end{pmatrix}. \tag{3.29}$$

Zatim se koristi Jakobijan funkcije h, koji linearizuje mjerni model oko trenutne procjene:

$$H_t = \frac{\partial h(\bar{\mu}_t)}{\partial \bar{\mu}_t}. \tag{3.30}$$

Lokalni Jakobijan $H_t^{[i]}$, izračunat za obilježje i, ima sljedeći oblik:

$$H_t^{[i]} = \frac{1}{q} \begin{bmatrix} -\sqrt{q}\delta_x & -\sqrt{q}\delta_y & 0 & \sqrt{q}\delta_x & \sqrt{q}\delta_y \\ \delta_y & -\delta_x & -q & -\delta_y & \delta_x \end{bmatrix}. \tag{3.31}$$

Pošto ova matrica djeluje samo na ograničeni broj elemenata stanja, koristi se proširena matrica $F_{x,j}$ da bi se projektovao Jakobijan u cjelokupan prostor stanja. Matrica $F_{x,j}$ dimenzije $5 \times (3 + 2n)$ ima oblik:

$$F_{x,j} = \begin{pmatrix} 1 & 0 & 0 & \underbrace{0 \cdots 0}_{2j-2} & 0 & 0 & \underbrace{0 \cdots 0}_{2n-2j} \\ 0 & 1 & 0 & 0 \cdots 0 & 0 & 0 & 0 \cdots 0 \\ 0 & 0 & 1 & 0 \cdots 0 & 0 & 0 & 0 \cdots 0 \\ 0 & 0 & 0 & 0 \cdots 0 & 1 & 0 & 0 \cdots 0 \\ 0 & 0 & 0 & 0 \cdots 0 & 0 & 1 & 0 \cdots 0 \end{pmatrix}. \tag{3.32}$$

Ova matrica omogućava preslikavanje lokalnog Jakobijana – izračunatog u koordinatnom sistemu senzora (ili lokalne reference) – u globalni prostor stanja.

Pri uočavanju novih obilježja, vektor stanja i matrica kovarijanse proširuju se kako bi uključili njihove pozicije i kovarijanse. Iako ovaj korak nije eksplicitno prikazan u osnovnim algoritmima, on predstavlja sastavni dio svake praktične implementacije. Kao i ranije, koristi se činjenica da su mnoge od ovih matrica rijetke (engl. *sparse*), što omogućava efikasniju računsku obradu.

Potpuni pseudokod EKF SLAM algoritma nalazi se u Dodatku (Algoritam 8).

3.2.2 Unscented Kalmanov filter za SLAM

Unscented Kalman Filter (UKF) za SLAM koristi istu formulaciju problema kao i prošireni Kalmanov filter (vidjeti poglavlje 3.2.1), ali se oslanja na Unscented transformaciju kako bi izbjegao greške koje nastaju usljed linearizacije nelinearnih modela [10, 41–45]. Algoritam se sastoji od sljedećih koraka:

Inicijalizacija

Budući da je inicijalizacija UKF SLAM-a ista kao kod EKF SLAM-a, njeno ponovno izvođenje se preskače.

Generisanje sigma-tačaka

Za prostor stanja dimenzije n, generiše se simetričan skup od $2n+1$ sigma-tačaka $\{\chi_{t-1}^{[i]}\}_{i=0}^{2n}$, zajedno sa težinskim koeficijentima $W_m^{[i]}$ i $W_c^{[i]}$ (vidjeti poglavlje 1.6.3).

Za efikasno i numerički stabilno računanje kvadratnog korijena kovarijanse Σ_{t-1} koristi se Čoleski dekompozicija:

$$\Sigma_{t-1} = SS^T, \tag{3.33}$$

nasuprot klasičnoj dekompoziciji

$$\Sigma_{t-1} = V D V^{-1}, \tag{3.34}$$

zbog bolje numeričke stabilnosti, većih performansi i efikasnije implementacije.

Predikcioni korak

Svaka sigma-tačka se propagira kroz nelinearni model kretanja, čime se dobija:

$$\chi_{t|t-1}^{[i]} = g\big(u_t, \chi_{t-1}^{[i]}\big). \tag{3.35}$$

Zatim se iz njih izračunavaju predikcija srednje vrijednosti i kovarijanse:

$$\bar{\mu}_t = \sum_{i=0}^{2n} W_m^{[i]}\, \chi_{t|t-1}^{[i]}, \tag{3.36}$$

$$\bar{\Sigma}_t = \sum_{i=0}^{2n} W_c^{[i]} \left(\chi_{t|t-1}^{[i]} - \bar{\mu}_t\right)\left(\chi_{t|t-1}^{[i]} - \bar{\mu}_t\right)^T + R_t. \tag{3.37}$$

Korekcioni korak

Predviđene sigma-tačke se zatim projektuju kroz mjerni model:

$$z_{t|t-1}^{[i]} = h\big(\chi_{t|t-1}^{[i]}\big). \tag{3.38}$$

Na osnovu toga, izračunavaju se očekivana mjerenja, kovarijansa inovacije i međukovarijansa:

$$\hat{z}_t = \sum_{i=0}^{2n} W_m^{[i]}\, z_{t|t-1}^{[i]}, \tag{3.39}$$

$$S_t = \sum_{i=0}^{2n} W_c^{[i]} \left(z_{t|t-1}^{[i]} - \hat{z}_t\right)\left(z_{t|t-1}^{[i]} - \hat{z}_t\right)^T + Q_t, \tag{3.40}$$

$$\Sigma_t^{x,z} = \sum_{i=0}^{2n} W_c^{[i]} \left(\chi_{t|t-1}^{[i]} - \bar{\mu}_t\right)\left(z_{t|t-1}^{[i]} - \hat{z}_t\right)^T. \tag{3.41}$$

Na osnovu ovih vrijednosti, izračunavaju se Kalmanovo pojačanje, ažurirana srednja vrijednost i kovarijansa:

$$K_t = \Sigma_t^{x,z}\, S_t^{-1}, \tag{3.42}$$

$$\mu_t = \bar{\mu}_t + K_t\big(z_t - \hat{z}_t\big), \tag{3.43}$$

$$\Sigma_t = \bar{\Sigma}_t - K_t\, S_t\, K_t^T. \tag{3.44}$$

Potpuni pseudokod UKF SLAM algoritma nalazi se u Dodatku (Algoritam 9).

3.3 Particle SLAM

SLAM algoritmi suočavaju se sa značajnim izazovima kada je u pitanju skaliranje na prostore velike dimenzionalnosti. Iako su particle filteri efikasni u problemima manje dimenzionalnosti, njihova računska složenost raste eksponencijalno sa brojem promjenjivih, za razliku od Kalmanovih filtera, koji skaliraju kvadratno [2,10,46,47]. Zbog toga su direktne implementacije klasičnih particle filtera često nepraktične u složenijim SLAM scenarijima.

Kako bi se prevazišao ovaj problem, koristi se *Rao-Blackwellizovani particle filter (RBPF)* [48], koji se oslanja na uslovnu nezavisnost karakteristika mape pod pretpostavkom da je putanja robota poznata. Ova faktorizacija omogućava razdvajanje SLAM problema na dva podproblema:

- Procjenu distribucije pozicije robota, koja se rješava pomoću particle filtera, i
- Nezavisnu procjenu pozicija obilježja, pri čemu se za svako obilježje koristi Gaussov filter (najčešće EKF).

Ovakva dekompozicija pruža značajne računske prednosti u odnosu na metode koje zajednički procjenjuju cijeli vektor stanja unutar jedne Gaussove distribucije.

Pristup zasnovan na ovim principima poznat je kao **FastSLAM**. On koristi particle filtere za praćenje putanje robota, dok istovremeno za svaku česticu održava skup nezavisnih EKF-ova male dimenzije za procjenu položaja obilježja. Ova dvostruka reprezentacija omogućava FastSLAM algoritmu da:

- Aproksimira kompletnu distribuciju putanje robota,
- Istovremeno razmatra višestruke hipoteze o asocijaciji podataka, čime se povećava robusnost u uslovima pogrešnih asocijacija.

Za razliku od metoda koje se oslanjaju na jednu najvjerovatniju asocijaciju, FastSLAM omogućava istovremeno održavanje i propagaciju više alternativnih asocijacija, što povećava otpornost algoritma na greške u asocijaciji podataka.

Dodatna prednost FastSLAM-a je podrška za nelinearne modele kretanja, bez potrebe za lokalnom linearnom aproksimacijom, što ga čini pogodnim za primjenu u sistemima sa složenom kinematikom i visokom nesigurnošću u poziciji robota.

3.3.1 Postavka i faktorizacija SLAM problema

Particle filter se zasniva na tri glavna koraka [49]:

- *Sampling* – uzorkovanje iz predložene distribucije $\chi_t^{[k]} \sim p(\chi_t \mid \ldots)$,
- Ažuriranje težinskih koeficijenata $w_t^{[k]} = \frac{\text{target}(\chi_t^{[k]})}{\text{proposal}(\chi_t^{[k]})}$,
- *Resampling* – ponovno uzorkovanje na osnovu težina $w_t^{[k]}$.

Pošto su obilježja mape uslovno nezavisna jedna od drugih – ako je putanja robota poznata – možemo izvršiti faktorizaciju zajedničke vjerovatnoće:

$$p(\chi, m) = p(m \mid \chi)p(\chi). \tag{3.45}$$

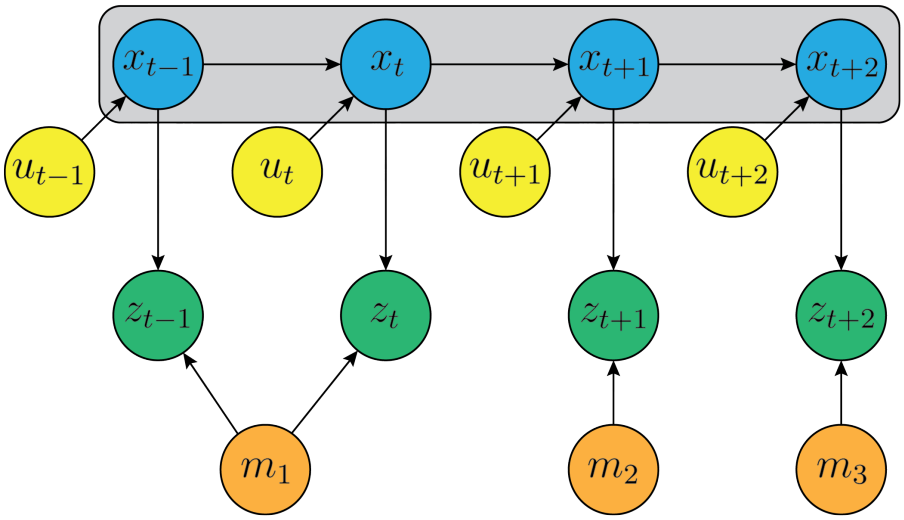
U kontekstu SLAM-a, zajednička uslovna vjerovatnoća se razlaže kao:

$$p(\chi_{0:t}, m \mid z_{1:t}, u_{1:t}, c_{1:t}) = \underbrace{p(\chi_{0:t} \mid z_{1:t}, c_{1:t}, u_{1:t})}_{\text{PF lokalizacija}} \prod_{j=1}^{M} \underbrace{p(m_j \mid \chi_{0:t}, z_{1:t}, c_{1:t})}_{\text{EKF mapiranje}}, \tag{3.46}$$

gdje $\chi_{0:t}$ predstavlja putanju robota, a m_j su pojedinačna obilježja u mapi.

Ova faktorizacija poznata je kao **Rao-Blackwellizacija**, koja koristi tehniku iz statistike za redukciju varijanse procjene – analitičko rješenje (npr. EKF za m_j) kombinuje se sa uzorkovanjem (particle filter za $\chi_{0:t}$).

Na ovaj način, SLAM problem se razlaže na jedan visokodimenzionalni zadatak procjene putanje robota i M nezavisnih estimatora za pojedinačna obilježja. Ključna prednost ovog pristupa je u tome što rezultujuća distribucija, koja se dobija kao produkt marginalnih distribucija, predstavlja **tačan** opis posteriorne vjerovatnoće – a ne njenu aproksimaciju.



Slika 3.3: Bayesova mreža koja ilustruje uslovnu nezavisnost obilježja pri poznatoj putanji robota – osnovu Rao-Blackwellizacije u SLAM-u.

Radi ilustracije uslovne nezavisnosti, na slici 3.3 prikazan je proces prikupljanja podataka i zavisnosti među promjenjivim u Bayesovoj mreži. Pretpostavimo da se robot kreće iz pozicije x_{t-1}, zatim kroz x_t, sve do x_{t+2}, vođen nizom upravljačkih signala. U svakoj od tih pozicija robot opaža neko od obilježja iz mape $m = \{m_1, m_2, m_3\}$. Graf prikazuje da su pozicije robota ključna veza (engl. *key link*) između obilježja. Kada su poznate, ne postoji alternativni put koji povezuje različita obilježja m_i, čime ona postaju uslovno nezavisna.[2] To znači da poznavanje položaja jednog obilježja ne pruža dodatne informacije o drugima.

[2]Primjer: Zamislimo da robot uočava dva obilježja (m_1 i m_2), ali ne znamo gdje je robot bio kad ih je opazio. Ako pogriješimo u procjeni pozicije robota, onda ćemo i za m_1 i za m_2 pogriješiti u istom "smjeru". Dakle, greške u procjeni m_1 i m_2 su povezane ukoliko pozicija robota nije poznata.

3.3.2 FastSLAM 1.0

Kao što je već pomenuto, FastSLAM koristi particle filter za procjenu vjerovatnoće položaja robota, dok se lokacije obilježja u mapi procjenjuju pomoću EKF-ova. Zahvaljujući faktorizaciji, moguće je za svako obilježje koristiti poseban EKF, što omogućava efikasnije ažuriranje nego u EKF SLAM-u. Budući da je svaki EKF uslovljen pozicijom robota, svaka čestica u particle filteru održava svoj sopstveni skup EKF-ova. Ukupno postoji $N \times M$ EKF-ova – po jedan za svako obilježje m_i u mapi, za svaku od N čestica.

U slučaju poznate asocijacije podataka, čestice imaju sljedeću strukturu:

$$\chi_t^{[k]} = \left[x_t^{[k]}, \left\{ \left(\mu_{1,t}^{[k]}, \Sigma_{1,t}^{[k]} \right), \ldots, \left(\mu_{M,t}^{[k]}, \Sigma_{M,t}^{[k]} \right) \right\}, w_t^{[k]} \right]. \tag{3.47}$$

Svaka čestica $\chi_t^{[k]}$ sadrži:

- poziciju robota $x_t^{[k]}$,
- skup Kalmanovih filtera $(\mu_{j,t}^{[k]}, \Sigma_{j,t}^{[k]})$ za svako obilježje m_j,
- težinski koeficijent $w_t^{[k]}$.

Ovde je k indeks čestice, dok je M ukupan broj obilježja u mapi. Računanje pozicije u trenutku t zasniva se na generisanju novog skupa čestica χ_t iz prethodnog skupa χ_{t-1}, korišćenjem odgovarajuće distribucije i ažuriranjem mjerenja, što je detaljno prikazano u nastavku.

Predlog odabiranja

Na osnovu upravljačkog signala u_t se uzorkuje nova pozicija $x_t^{[k]}$ za svaku česticu iz:

$$x_t^{[k]} \sim p(x_t \mid x_{t-1}^{[k]}, u_t). \tag{3.48}$$

Ažuriranje obilježja

Za svako obilježje m_{c_t}, nova srednja vrijednost i kovarijansa ažuriraju se pomoću EKF-a. Ako obilježje m_j nije uočeno u trenutku t ($j \neq c_t$), njegovo stanje ostaje nepromijenjeno:

$$\left\langle \mu_{j,t}^{[k]}, \Sigma_{j,t}^{[k]} \right\rangle = \left\langle \mu_{j,t-1}^{[k]}, \Sigma_{j,t-1}^{[k]} \right\rangle. \tag{3.49}$$

Korišćenjem EKF-a, mjerni model se linearizuje oko trenutne procjene pozicije robota i položaja obilježja. Funkcija mjerenja $h(m_{c_t}, x_t)$ aproksimira se Tejlorovim

razvojem:

$$h\left(m_{c_t}, x_t^{[k]}\right) \approx \underbrace{h\left(\mu_{c_t,t-1}^{[k]}, x_t^{[k]}\right)}_{=:\hat{z}_t^{[k]}} + \underbrace{h'\left(\mu_{c_t,t-1}^{[k]}, x_t^{[k]}\right)}_{=:H_t^{[k]}}\left(m_{c_t} - \mu_{c_t,t-1}^{[k]}\right) \tag{3.50}$$

$$= \hat{z}_t^{[k]} + H_t^{[k]}\left(m_{c_t} - \mu_{c_t,t-1}^{[k]}\right), \tag{3.51}$$

gdje je $H_t^{[k]}$ Jakobijan funkcije mjerenja po m_{c_t}.

Na osnovu toga, ažuriranje se vrši prema klasičnim EKF formulama:

$$K_t^{[k]} = \Sigma_{c_t,t-1}^{[k]} H_t^{[k]T}\left(H_t^{[k]}\Sigma_{c_t,t-1}^{[k]}H_t^{[k]T} + Q_t\right)^{-1}, \tag{3.52}$$

$$\mu_{c_t,t}^{[k]} = \mu_{c_t,t-1}^{[k]} + K_t^{[k]}(z_t - \hat{z}_t^{[k]}), \tag{3.53}$$

$$\Sigma_{c_t,t}^{[k]} = (I - K_t^{[k]}H_t^{[k]})\Sigma_{c_t,t-1}^{[k]}. \tag{3.54}$$

Ova dva koraka se ponavljaju za svaku česticu $k = 1, \ldots, N$, nakon čega se vrši resampling i formira novi skup χ_t.

Resampling i težinski faktori

Jedan od ključnih aspekata koji utiču na performanse particle filtera je proces ponovnog odabiranja (resampling). Tokom ovog koraka, čestice s niskim težinskim koeficijentima zamjenjuju se onima s većim težinama, čime se poboljšava reprezentacija posteriorne distribucije. Odluka o ponovnom uzorkovanju zasniva se na *efektivnom broju čestica*, koji se procjenjuje kao:

$$N_{\text{eff}} = \frac{1}{\sum_{k=1}^{N}\left(\hat{w}_t^{[k]}\right)^2}, \tag{3.55}$$

gdje je $\hat{w}_t^{[k]}$ normalizovana težina k-te čestice. Kada je N_{eff} ispod zadatog praga (npr. 50% od ukupnog broja čestica), pokreće se postupak ponovnog odabiranja kako bi se izbjegla degeneracija čestica, koja nastaje kada većina čestica ima zanemarljivo niske težinske faktore, što smanjuje njihov doprinos ukupnoj procjeni stanja.

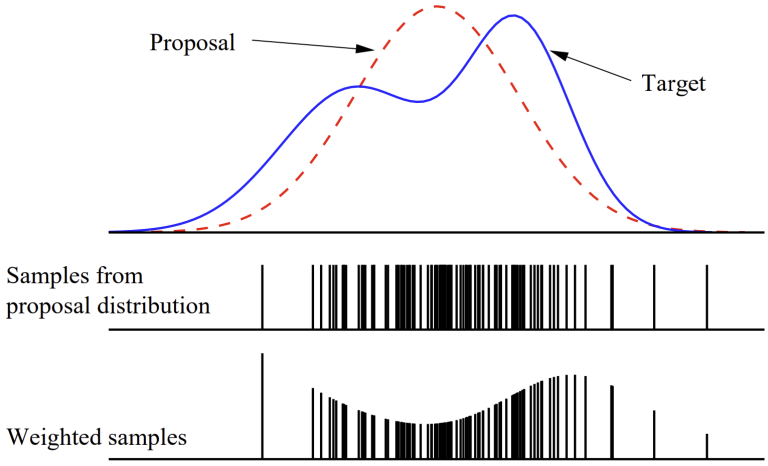
U FastSLAM-u, novi skup čestica χ_t formira se tako što se čestice nasumično odabiraju iz privremenog skupa, pri čemu je vjerovatnoća izbora proporcionalna njihovim težinskim koeficijentima[3]. Ovaj postupak omogućava da se više resursa usmjeri na čestice koje bolje aproksimiraju trenutno stanje sistema.

Budući da početni skup čestica odražava samo upravljačke ulaze, bez uticaja

[3]Čestice se mogu birati s ponavljanjem.

mjerenja, težinski faktori služe kao korektivni mehanizam. Međutim, tek se ponovnim odabiranjem postiže raspodjela koja zaista odražava posteriornu vjerovatnoću – onu koja objedinjuje informacije upravljanja i senzora.

Na slici 3.4 ilustrovan je princip resamplovanja u jednoj dimenziji. Isprekidana linija prikazuje *predloženu distribuciju* – raspodjelu iz koje su prvobitno generisane čestice, dok puna linija predstavlja *ciljnu distribuciju*, koja uključuje najnovija mjerenja i predstavlja željenu posteriornu raspodjelu.



Slika 3.4: Princip odabiranja i dodjele težinskih koeficijenata u procesu resamplovanja [2].

Proces resamplovanja kompenzuje razliku između ciljne i predložene distribucije. Težinski koeficijenti $w_t^{[k]}$ dodjeljuju se česticama na osnovu njihove usklađenosti sa ciljnom distribucijom, kao što je to uobičajeno u pristupu zasnovanom na važnosnom uzorkovanju (*importance sampling*). Nakon što se čestice ponovo odaberu proporcionalno tim težinama, dobijeni skup bolje aproksimira ciljnu distribuciju.

Za određivanje težinskih faktora potrebno je razumjeti razliku između predložene i ciljne distribucije u trenutnom vremenskom koraku. Ako pretpostavimo da su čestice u prethodnom skupu raspoređene prema distribuciji:

$$p\left(x_{t-1}^{[k]} \mid z_{1:t-1}, c_{1:t-1}, u_{1:t-1}\right), \tag{3.56}$$

onda se nove čestice generišu iz predložene distribucije sljedećim izrazom:

$$p\left(x_t^{[k]} \mid z_{1:t-1}, c_{1:t-1}, u_{1:t}\right) = p\left(x_t^{[k]} \mid x_{t-1}^{[k]}, u_t\right) \cdot p\left(x_{t-1}^{[k]} \mid z_{1:t-1}, c_{1:t-1}, u_{1:t-1}\right). \tag{3.57}$$

Nasuprot tome, distribucija koja nas interesuje – tzv. *ciljna distribucija* – uzima u obzir i trenutno mjerenje z_t i korespondenciju c_t, te ima oblik:

$$p(x_{1:t}^{[k]} \mid z_{1:t}, c_{1:t}, u_{1:t}). \tag{3.58}$$

Ova ciljna distribucija može se dalje rastaviti pomoću Bayesove formule:

$$p(x_{1:t}^{[k]} \mid z_{1:t}, c_{1:t}, u_{1:t}) = \eta\, p(z_t \mid x_{1:t}^{[k]}, z_{1:t-1}, c_{1:t}, u_{1:t}) \cdot p(x_{1:t}^{[k]} \mid z_{1:t-1}, c_{1:t}, u_{1:t}) \quad (3.59)$$

$$= \eta\, p(z_t \mid x_t^{[k]}, c_t) \cdot p(x_{1:t}^{[k]} \mid z_{1:t-1}, c_{1:t-1}, u_{1:t}), \quad (3.60)$$

gdje je iskorištena uslovna nezavisnost na osnovu Markovljevih pretpostavki modela senzora. Na osnovu toga, težinski faktor za česticu k definiše se kao odnos ciljne i predložene distribucije:

$$w_t^{[k]} = \frac{p(x_{1:t}^{[k]} \mid z_{1:t}, c_{1:t}, u_{1:t})}{p(x_{1:t}^{[k]} \mid z_{1:t-1}, c_{1:t-1}, u_{1:t})} = \eta\, p(z_t \mid x_t^{[k]}, c_t). \quad (3.61)$$

Izračunavanje vjerovatnoće $p(z_t \mid x_t^{[k]}, c_t)$, zahtijeva dodatnu transformaciju izraza. Konkretno, ova vjerovatnoća odgovara sljedećoj integraciji, gdje se izostavljaju varijable koje nisu relevantne za predikciju senzorskih mjerenja:

$$w_t^{[k]} = \eta \int p(z_t \mid m_{c_t}, x_t^{[k]}, c_t) \underbrace{p(m_{c_t} \mid x_{t-1}^{[k]}, z_{1:t-1}, c_{1:t-1})}_{\sim \mathcal{N}(\mu_{c_t,t-1}^{[k]}, \Sigma_{c_t,t-1}^{[k]})} dm_{c_t}, \quad (3.62)$$

gdje je η faktor normalizacije. Ovdje se koristi prethodna vjerovatnoća nad obilježjem m_{c_t}, koja odgovara Gaussovoj distribuciji sa srednjom vrijednošću $\mu_{c_t,t-1}^{[k]}$ i kovarijansom $\Sigma_{c_t,t-1}^{[k]}$.

Kako bi se ova integracija riješila u zatvorenom obliku, FastSLAM koristi istu linearnu aproksimaciju kao u EKF ažuriranju mjerenja. Rezultujući izraz za težinski faktor postaje:

$$w_t^{[k]} \approx \eta \left| 2\pi Q_t^{[k]} \right|^{-1/2} \exp\left(-\frac{1}{2} (z_t - \hat{z}_t^{[k]})^T (Q_t^{[k]})^{-1} (z_t - \hat{z}_t^{[k]}) \right), \quad (3.63)$$

gdje je kovarijansa predikcije:

$$Q_t^{[k]} = H_t^{[k]} \Sigma_{c_t,t-1}^{[k]} H_t^{[k]T} + Q_t. \quad (3.64)$$

Ova tri koraka zajedno čine pravilo ažuriranja algoritma FastSLAM 1.0 za SLAM sa poznatom asocijacijom podataka. Važno je istaći da se vrijeme izvršavanja ne povećava sa dužinom putanje robota. Naime, u svakom trenutku t koristi se samo najnovija pozicija x_t, pa se prethodne pozicije mogu odbaciti, čime se postiže efikasnost u memorijskom i računskom smislu. Pseudokod algoritma FastSLAM 1.0 prikazan je u dodatku (Algoritam 10).

3.3.3 FastSLAM 2.0 – Bolji predlog distribicuje

FastSLAM 2.0 predstavlja proširenje algoritma FastSLAM 1.0, pri čemu se prilikom uzorkovanja novih pozicija robota ne koristi isključivo kontrolni signal u_t, već se u obzir uzimaju i najnovija mjerenja z_t:

$$x_t^{[k]} \sim p\left(x_t \mid x_{t-1}^{[k]}, u_{1:t}, z_{1:t}, c_{1:t}\right). \tag{3.65}$$

Ovakav pristup omogućava preciznije generisanje uzoraka kada je nesigurnost upravljanja velika u odnosu na nesigurnost u mjerenjima, jer koristi informaciju iz senzora za bolju procjenu stanja. Time se uzorkovanje vrši iz unaprijeđene distribucije koja se bolje poklapa sa stvarnom posteriornom raspodjelom [10].

Predlog odabiranja

Izražavanje ove nove predložene distribucije zahtijeva integraciju mjernog modela, koji je najčešće nelinearan:

$$\begin{aligned} &p\left(x_t \mid x_{1:t-1}^{[k]},\, u_{1:t},\, z_{1:t},\, c_{1:t}\right) = \\ &\eta^{[k]} \int \underbrace{p\left(z_t \mid m_{c_t}, x_t, c_t\right)}_{\sim\mathcal{N}(z_t; h(m_{c_t}, x_t), Q_t)} \underbrace{p\left(m_{c_t} \mid x_{1:t-1}^{[k]}, z_{1:t-1}, c_{1:t-1}\right)}_{\sim\mathcal{N}(m_{c_t}; \mu_{c_t,t-1}^{[k]}, \Sigma_{c_t,t-1}^{[k]})} dm_{c_t} \cdot \underbrace{p\left(x_t \mid x_{t-1}^{[k]}, u_t\right)}_{\sim\mathcal{N}(x_t; g(x_{t-1}^{[k]}, u_t), R_t)}. \end{aligned} \tag{3.66}$$

Zbog toga, distribucija nema rješenje u zatvorenom obliku i mora se aproksimirati linearizacijom nelinearne funkcije $h(m_{c_t}, x_t)$ Tejlorovim razvojem:

$$h(m_{c_t}, x_t) \approx \hat{z}_t^{[k]} + H_m\left(m_{c_t} - \mu_{c_t,t-1}^{[k]}\right) + H_x\left(x_t - \hat{x}_t^{[k]}\right), \tag{3.67}$$

gdje su:

$$\hat{z}_t^{[k]} = h\left(\mu_{c_t,t-1}^{[k]}, \hat{x}_t^{[k]}\right), \tag{3.68}$$

$$\hat{x}_t^{[k]} = g(x_{t-1}^{[k]}, u_t), \tag{3.69}$$

$$H_m = \nabla_{m_{c_t}} h(m_{c_t}, x_t)\big|_{x_t=\hat{x}_t^{[k]}, m_{c_t}=\mu_{c_t,t-1}^{[k]}}, \tag{3.70}$$

$$H_x = \nabla_{x_t} h(m_{c_t}, x_t)\big|_{x_t=\hat{x}_t^{[k]}, m_{c_t}=\mu_{c_t,t-1}^{[k]}}. \tag{3.71}$$

Sa ovim linearnim aproksimacijama, predložena distribucija uzorkovanja dobija oblik Gaussove raspodjele:

$$\Sigma_{x_t}^{[k]} = \left(H_x^T Q_t^{[k]^{-1}} H_x + R_t^{-1}\right)^{-1}, \tag{3.72}$$

$$\mu_{x_t}^{[k]} = \Sigma_{x_t}^{[k]} H_x^T Q_t^{[k]^{-1}} (z_t - \hat{z}_t^{[k]}) + \hat{x}_t^{[k]}, \tag{3.73}$$

gdje je Q_t efektivna kovarijansa mjerenja definisana izrazom:

$$Q_t^{[k]} = Q_t + H_m \Sigma_{c_t,t-1}^{[k]} H_m^T. \tag{3.74}$$

Kao rezultat, nova predložena distribucija može se interpretirati kao Gaussova raspodjela koja uzima u obzir i upravljačke signale i senzorska mjerenja, pružajući time efikasnije uzorkovanje i manju varijansu u odnosu na osnovni FastSLAM 1.0.

Ažuriranje obilježja

Kao i kod FastSLAM-a 1.0, ako mjerenje za obilježje nise dostupno ($j \neq c_t$), pozicija tog obilježja ostaje nepromijenjena i koristi se prethodno izračunata vrijednost. U suprotnom, kada dođe do novog opažanja ($j = c_t$), posteriorna vjerovatnoća pozicije obilježja u mapi m_{c_t} se ažurira prema sljedećem izrazu:

$$p\left(m_{c_t} \mid x_{1:t}^{[k]}, c_{1:t}, z_{1:t}\right) = \eta\, p\left(z_t \mid m_{c_t}, x_t^{[k]}, c_t\right)\, p\left(m_{c_t} \mid x_{1:t-1}^{[k]}, z_{1:t-1}, c_{1:t-1}\right). \tag{3.75}$$

Kao što je navedeno u prethodnom poglavlju, nelinearnost funkcije opažanja $h(m_{c_t}, x_t)$ dovodi do raspodjele koja odstupa od Gaussove forme. Zbog toga se vrši linearizacija funkcije opažanja oko prethodne procjene srednje vrijednosti:

$$h(m_{c_t}, x_t) \approx \hat{z}_t^{[k]} + H_m \left(m_{c_t} - \mu_{c_t,t-1}^{[k]}\right). \tag{3.76}$$

S obzirom na to da se trenutno pozicija robota x_t smatra poznatom (budući da je procijenjena u prethodnom koraku), može se izostaviti iz izraza. Na osnovu toga, prethodna raspodjela dobija oblik:

$$\begin{aligned} p\big(m_{c_t} \mid x_{1:t}^{[k]}, c_{1:t}, z_{1:t}\big) = \eta \exp\Big\{ & -\tfrac{1}{2}\big(z_t - \hat{z}_t^{[k]} - H_m(m_{c_t} - \mu_{c_t,t-1}^{[k]})\big)^T Q_t^{-1} \\ & \cdot \big(z_t - \hat{z}_t^{[k]} - H_m(m_{c_t} - \mu_{c_t,t-1}^{[k]})\big) \\ & - \tfrac{1}{2}(m_{c_t} - \mu_{c_t,t-1}^{[k]})^T \big(\Sigma_{c_t,t-1}^{[k]}\big)^{-1} (m_{c_t} - \mu_{c_t,t-1}^{[k]})\Big\}. \end{aligned} \tag{3.77}$$

Umjesto direktne manipulacije punim oblikom posteriorne raspodjele, koristi se EKF, koji nudi efikasan algoritam za ažuriranje srednje vrijednosti i kovarijanse Gaussove raspodjele zapisan u formi:

$$K_t^{[k]} = \Sigma_{c_t,t-1}^{[k]} H_m^T Q_t^{-1}, \tag{3.78}$$

$$\mu_{c_t,t}^{[k]} = \mu_{c_t,t-1}^{[k]} + K_t^{[k]} \left(z_t - \hat{z}_t^{[k]}\right), \tag{3.79}$$

$$\Sigma_{c_t,t}^{[k]} = \left(I - K_t^{[k]} H_m\right) \Sigma_{c_t,t-1}^{[k]}. \tag{3.80}$$

Težinski faktori

Kao i kod FastSLAM-a 1.0, težinski koeficijenti $w_t^{[k]}$ određuju se izrazom 3.61. Ciljna distribucija u FastSLAM-u 2.0 je:

$$p\left(x_t^{[k]} \mid z_{1:t}, u_{1:t}, c_{1:t}\right), \tag{3.81}$$

dok se trenutna distribucija dobija uzorkovanjem iz ciljne distribucije prethodnog koraka:

$$x_{t-1}^{[k]} \sim p\left(x_{t-1}^{[k]} \mid z_{1:t-1}, u_{1:t-1}, c_{1:t-1}\right). \tag{3.82}$$

U okviru FastSLAM-a 2.0, procjena težinskog faktora svake čestice zasniva se na vjerovatnoći da se trenutno opažanje z_t može objasniti prethodnim stanjem čestice i svim prethodnim informacijama:

$$w_t^{[k]} = \eta\, p\left(z_t \mid x_{t-1}^{[k]}, u_{1:t}, z_{1:t-1}, c_{1:t}\right). \tag{3.83}$$

Ukoliko se koristi optimalna važnosna funkcija zasnovana na trenutnom opažanju, ona uključuje i mjerenje z_t, te se koristi oblik:

$$\begin{aligned} w_t^{[k]} = \eta \int \underbrace{p\Big(x_t \mid x_{t-1}^{[k]}, u_t\Big)}_{\sim\mathcal{N}(x;g(x_{t-1}^{[k]},u_t),R_t)} \int \underbrace{p\Big(z_t \mid m_{c_t}, x_t, c_t\Big)}_{\sim\mathcal{N}(z_t;h(m_{c_t},x_t),Q_t)} \\ \underbrace{p\Big(m_{c_t} \mid x_{1:t-1}^{[k]}, u_{1:t-1}, z_{1:t-1}, c_{1:t-1}\Big)}_{\sim\mathcal{N}(m_{c_t};\mu_{c_t,t-1}^{[k]},\Sigma_{c_t,t-1}^{[k]})}\, dm_{c_t}\, dx_t. \end{aligned} \tag{3.84}$$

Ponovno se pristupa linearizaciji funkcije opažanja, pri čemu nova srednja vrijednost normalne raspodjele postaje $\hat{z}_t^{[k]}$, dok kovarijansa prelazi u oblik:

$$L_t^{[k]} = H_x^T Q_t H_x + H_m \Sigma_{c_t,t-1}^{[k]} H_m^T + R_t. \tag{3.85}$$

Na kraju, konačni izraz za računanje težinskog koeficijenta svake čestice je:

$$w_t^{[k]} = \left|2\pi L_t^{[k]}\right|^{-1/2} \exp\left\{-\frac{1}{2}(z_t - \hat{z}_t)^T (L_t^{[k]})^{-1}(z_t - \hat{z}_t)\right\}. \tag{3.86}$$

Posljednji korak je ponovno odabiranje (resampling), koje ostaje identično kao kod FastSLAM-a 1.0, i detaljno je opisano u poglavlju 3.3.2.

Pseudokod algoritma FastSLAM 2.0 prikazan je u dodatku (Algoritam 11).

3.3.4 uFastSLAM

Tradicionalni algoritmi particle filtera, kao što su FastSLAM 1.0 i FastSLAM 2.0 koriste Rao-Blackwellizovani particle filter (RBPF) za procjenu putanje robota i mapiranje karakteristika u nepoznatom okruženju. Iako ovi algoritmi pokazuju visoku efikasnost, njihova primjena je ograničena potrebom za linearnim aproksimacijama i računanjem Jakobijevih matrica. Takve aproksimacije mogu dovesti do akumulacije grešaka, što zahtijeva korišćenje većeg broja čestica kako bi se održala potrebna preciznost.

Algoritam uFastSLAM unapređuje ovaj pristup tako što smanjuje broj potrebnih čestica, a istovremeno zadržava visoku preciznost estimacije. To se postiže primjenom unscented transformacije, koja omogućava bolje modelovanje nelinearnih sistema bez potrebe za eksplicitnim izračunavanjem Jakobijevih matrica.

Predlog odabiranja

Prvo se proširuje vektor stanja kontrolnim signalima, dok se matrica kovarijanse proširuje kovarijansama mjerenja [50]:

$$x_{t-1}^{\alpha[k]} = \begin{bmatrix} x_{t-1}^{[k]} \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} x_{x,t-1}^{[k]} \\ x_{y,t-1}^{[k]} \\ x_{\theta,t-1}^{[k]} \\ 0 \\ 0 \end{bmatrix}, \qquad \Sigma_{t-1}^{\alpha[k]} = \begin{bmatrix} \Sigma_{t-1}^{[k]} & 0 & 0 \\ 0 & Q_t & 0 \\ 0 & 0 & R_t \end{bmatrix}. \tag{3.87}$$

Oznaka x^{α} predstavlja proširenu verziju vektora, odnosno matrice stanja. Nakon toga, kreira se set simetričnih sigma tačaka i odgovarajući težinski koeficijenti prema pravilima Unscented transformacije (poglavlje 1.6.3), čime se omogućava da svaka čestica sadrži informacije o stanju robota, kontrolnim signalima i mjerenjima.

$$x_{t-1}^{\alpha[i][k]} = \begin{bmatrix} x_{t-1}^{[i][k]} \\ x_t^{u[i][k]} \\ x_t^{z[i][k]} \end{bmatrix}. \tag{3.88}$$

Sigma tačke se zatim propagiraju kroz nelinearni model kretanja

$$\bar{x}_t^{[i][k]} = g\left(u_t^{[k]} + x_t^{u[i][k]}, x_{t-1}^{[i][k]}\right), \tag{3.89}$$

a srednja vrijednost i kovarijansa dobijaju se sumiranjem ponderisanih sigma tačaka:

$$\chi_{t|t-1}^{[k]} = \sum_{i=0}^{2n} W_g^{[i]}\, \bar{x}_t^{[i][k]}, \tag{3.90}$$

$$\Sigma_{t|t-1}^{[k]} = \sum_{i=0}^{2n} W_c^{[i]} \left(\bar{x}_t^{[i][k]} - \chi_{t|t-1}^{[k]}\right) \left(\bar{x}_t^{[i][k]} - \chi_{t|t-1}^{[k]}\right)^T. \tag{3.91}$$

Ako nema opservacija, koristi se samo predikcioni korak Unscented transformacije (formule (3.90) – (3.91)). Ukoliko su opservacije prisutne, postupak se proširuje. Za svaku opservaciju z_t posebno, primjenjuju se sljedeći koraci:

Prvo se kreira set sigma tačaka za mjerenje pomoću nelinearnog mjernog modela:

$$\bar{z}_t^{[i][k]} = h\left(\bar{x}_t^{[i][k]}, \mu_{c_t,t-1}^{[k]}\right) + x_t^{z[i][k]}, \tag{3.92}$$

zatim se predviđeno mjerenje dobija kao:

$$\hat{z}_t^{[k]} = \sum_{i=0}^{2n} w_g^{[i]}\, \bar{z}_t^{[i][k]}, \tag{3.93}$$

dok se inovaciona kovarijansa računa kao:

$$S_t^{[k]} = \sum_{i=0}^{2n} w_c^{[i]} \left(\bar{z}_t^{[i][k]} - \hat{z}_t^{[k]}\right) \left(\bar{z}_t^{[i][k]} - \hat{z}_t^{[k]}\right)^T + R_t. \tag{3.94}$$

Kros-kovarijansa između stanja i mjerenja izračunava se:

$$\Sigma_t^{\chi z[k]} = \sum_{i=0}^{2n} w_c^{[i]} \left(\bar{x}_t^{[i][k]} - \chi_{t|t-1}^{[k]}\right) \left(\bar{z}_t^{[i][k]} - \hat{z}_t^{[k]}\right)^T. \tag{3.95}$$

Na osnovu toga određuje se Kalmanovo pojačanje:

$$K_t^{[k]} = \Sigma_t^{\chi z[k]} \left(S_t^{[k]}\right)^{-1}, \tag{3.96}$$

i ažurira se srednja vrijednost i kovarijansa vektora stanja:

$$\chi_t^{[k]} = \chi_{t|t-1}^{[k]} + K_t^{[k]} \left(z_t - \hat{z}_t^{[k]}\right), \tag{3.97}$$

$$\Sigma_t^{[k]} = \Sigma_{t|t-1}^{[k]} - K_t^{[k]} S_t^{[k]} \left(K_t^{[k]}\right)^T. \tag{3.98}$$

Nakon svakog ažuriranja, sigma tačke se regenerišu na osnovu ažurirane srednje

vrijednosti i kovarijanse:

$$x_t^{\alpha[i][k]} = \left[\chi_t^{\alpha[k]} \quad \chi_t^{\alpha[k]} \pm \sqrt{(n+\lambda)\Sigma_t^{\alpha[k]}}\right], \tag{3.99}$$

$$\bar{x}_t^{[i][k]} = x_t^{[i][k]}. \tag{3.100}$$

Ova procedura se ponavlja za svaku pojedinačnu opservaciju z_t.

Ažuriranje obilježja

Za estimaciju položaja obilježja koristi se Unscented Kalman filter (UKF) umjesto Extended Kalman filtera (EKF). Najprije se kreira set sigma tačaka za obilježje na osnovu prethodne srednje vrijednosti $\mu_{c_t,t-1}^{[k]}$ i kovarijanse $\Sigma_{c_t,t-1}^{[k]}$:

$$x_t^{[0][m]} = \mu_{c_t,t-1}^{[m]}, \tag{3.101}$$

$$x_t^{[i][m]} = \mu_{c_t,t-1}^{[m]} + \left(\sqrt{(n+\lambda)\,\Sigma_{c_t,t-1}^{[m]}}\right)_i, \qquad i = 1,\ldots,n, \tag{3.102}$$

$$x_t^{[i][m]} = \mu_{c_t,t-1}^{[m]} - \left(\sqrt{(n+\lambda)\,\Sigma_{c_t,t-1}^{[m]}}\right)_{i-n}, \qquad i = n+1,\ldots,2n. \tag{3.103}$$

Predikcija mjerenja za svaku sigma tačku vrši se korišćenjem nelinearnog mjernog modela:

$$\bar{z}_t^{[i][k]} = h\left(x_t^{[i][k]}, \chi_t^{[k]}\right), \quad i = 0,\ldots,2n, \tag{3.104}$$

Predviđeno mjerenje zatim se računa kao ponderisana suma sigma tačaka:

$$\hat{z}_t^{[k]} = \sum_{i=0}^{2n} w_g^{[i]}\, \bar{z}_t^{[i][k]}, \tag{3.105}$$

Inovaciona kovarijansa mjerenja izračunava se kao:

$$\bar{S}_t^{[k]} = \sum_{i=0}^{2n} w_c^{[i]} \left(\bar{z}_t^{[i][k]} - \hat{z}_t^{[k]}\right)\left(\bar{z}_t^{[i][k]} - \hat{z}_t^{[k]}\right)^T + R_t, \tag{3.106}$$

Kros-kovarijansa između sigma tačaka obilježja i mjerenja računa se kao:

$$\bar{\Sigma}_t^{[k]} = \sum_{i=0}^{2n} w_c^{[i]} \left(\chi^{[i][k]} - \mu_{c_t,t-1}^{[k]}\right)\left(\bar{z}_t^{[i][k]} - \hat{z}_t^{[k]}\right)^T, \tag{3.107}$$

Na osnovu ovih izraza, Kalmanovo pojačanje je dato kao:

$$\bar{K}_t^{[k]} = \bar{\Sigma}_t^{[k]} \left(\bar{S}_t^{[k]}\right)^{-1}, \tag{3.108}$$

Konačno, srednja vrijednost ažuriranog obilježja je:

$$\mu_{c_t,t}^{[k]} = \mu_{c_t,t-1}^{[k]} + \bar{K}_t^{[k]} \left(z_t - \hat{z}_t^{[k]} \right), \tag{3.109}$$

a kovarijansa obilježja ažurira se prema:

$$\Sigma_{c_t,t}^{[k]} = \Sigma_{c_t,t-1}^{[k]} - \bar{K}_t^{[k]} \bar{S}_t^{[k]} \bar{K}_t^{[k]T}. \tag{3.110}$$

Težinski faktori

Analogno prethodnim FastSLAM algoritmima, težinski faktori $w_t^{[k]}$ u uFastSLAM-u računaju se pomoću formule:

$$w_t^{[k]} = \left| 2\pi L_t^{[k]} \right|^{-\frac{1}{2}} \exp \left\{ -\frac{1}{2} \left(z_t - \hat{z}_t^{[k]} \right)^T \left(L_t^{[k]} \right)^{-1} \left(z_t - \hat{z}_t^{[k]} \right) \right\}, \tag{3.111}$$

gdje je $L_t^{[k]}$ definisano kao:

$$L_t^{[k]} = \left(\Sigma_t^{x,c_t[k]} \right)^T \left(\Sigma_t^{[k]} \right)^{-1} \Sigma_t^{x,c_t[k]} + \bar{S}_t^{[k]}, \tag{3.112}$$

Nakon računanja težinskih faktora za sve čestice, ponovno odabiranje (resampling) vrši se ukoliko je zadovoljen uslov (3.55), pri čemu je procedura identična onoj opisanoj u prethodnim poglavljima.

Potpuni pseudokod EKF SLAM algoritma nalazi se u Dodatku (Algoritam 12).

3.3.5 Mapa obilježja

Inicijalizacija i čuvanje mape obilježja slična je implementacijama u EKF i UKF SLAM-u, s tim što se verzija mape sada čuva unutar svake čestice pojedinačno. U našem slučaju, mjerni model je invertibilan, što omogućava kreiranje nedegradirajuće estimacije obilježja.

Kreiranje EKF-a za novo obilježje prati sljedeće korake [10]:

$$x_t^{[k]} \sim p \left(x_t \mid x_{t-1}^{[k]}, u_t \right), \tag{3.113}$$

$$\mu_{n,t}^{[k]} = h^{-1} \left(z_t, x_t^{[k]} \right), \tag{3.114}$$

$$\Sigma_{n,t}^{[k]} = \left(H_z^{[k]T} Q_z^{-1} H_z^{[k]} \right)^{-1}, \tag{3.115}$$

$$w_t^{[k]} = p_0, \tag{3.116}$$

gdje je $H_z^{[k]} = h\left(\mu_{n,t}^{[k]}, x_t^{[k]}\right)$, dok je za UKF postupak sljedeći [50]:

$$\psi_t^{[0][k]} = z_t, \tag{3.117}$$

$$\psi_t^{[i][k]} = z_t + \left(\sqrt{(l+\lambda)R_t}\right)_i, \qquad i = 1, \ldots, l, \tag{3.118}$$

$$\psi_t^{[i][k]} = z_t - \left(\sqrt{(l+\lambda)R_t}\right)_{i-l}, \qquad i = l+1, \ldots, 2l, \tag{3.119}$$

$$\bar{M}_t^{[i][k]} = h^{-1}\left(\psi_t^{[i][k]}, x_t^{[k]}\right), \qquad i = 0, \ldots, 2l, \tag{3.120}$$

$$\mu_{c_t,t}^{[k]} = \sum_{i=0}^{2l} w_g^{[i]}\, \bar{M}_t^{[i][k]}, \tag{3.121}$$

$$\Sigma_{c_t,t}^{[k]} = \sum_{i=0}^{2l} w_c^{[i]} \left(\bar{M}_t^{[i][k]} - \mu_{c_t,t}^{[k]}\right)\left(\bar{M}_t^{[i][k]} - \mu_{c_t,t}^{[k]}\right)^T. \tag{3.122}$$

Za razliku od Kalmanovih filtera, particle filteri omogućavaju obradu negativnih informacija, što im daje veću fleksibilnost u upravljanju postojanjem obilježja na individualnom nivou. Budući da se svako obilježje obrađuje nezavisno unutar svake čestice, moguće je dodavati, uklanjati ili čak privremeno zadržavati, bez narušavanja ukupne konzistentnosti mape.

Jedan od načina za upravljanje inicijalizovanim obilježjima je održavanjem njihove logaritamske vjerovatnoće. Kada se neko obilježje opazi, njegova vjerovatnoća se povećava za fiksnu vrijednost. Ako obilježje nije opaženo, iako bi prema modelu trebalo biti, vjerovatnoća se smanjuje (potencijalno drugom fiksnom vrijednošću). Obilježja čija logaritamska vjerovatnoća padne ispod određenog praga mogu se ukloniti iz mape.

Glava 4

Računarska složenost i performanse

Pri razvoju i implementaciji SLAM algoritama, od ključnog je značaja razumjeti faktore koji utiču na njihovu primjenjivost u realnim uslovima. Pored tačnosti i pouzdanosti, računski zahtjevi često predstavljaju ograničavajući faktor, posebno u okruženjima sa ograničenim resursima ili kada je potrebno izvršavanje u realnom vremenu. Zbog toga je važno sistematski analizirati ove aspekte kako bi se omogućilo objektivno poređenje različitih algoritamskih pristupa i donošenje informisanih odluka pri izboru rješenja za konkretne robotske zadatke.

U skladu s tim, u narednom poglavlju biće predstavljene osnove analize računske i vremenske složenosti SLAM algoritama, čime se postavlja temelj za dalju komparativnu analizu.

4.1 Računska složenost

Procjena složenosti algoritama temelji se na analizi potrošnje resursa kao što su vrijeme i memorija. Kako bi poređenje različitih algoritama bilo moguće i nezavisno od konkretnih implementacija i hardverskih karakteristika, uvodi se formalni matematički okvir za upoređivanje brzina rasta funkcija koje opisuju složenost. U ovom poglavlju biće predstavljene osnovne asimptotske oznake, njihovo značenje, kao i tipični primjeri primjene.

Osnovne definicije

U literaturi se najčešće navode četiri osnovne asimptotske definicije složenosti [3]:

- **Veliko-O (Big-Oh) – Gornja asimptotska granica:**
  Kažemo da je $T(N) = O(f(N))$ ako postoje pozitivne konstante c i n_0 takve

da za sve $N \geq n_0$ važi $T(N) \leq cf(N)$.

- **Veliko-Omega (Big-Omega) – Donja asimptotska granica:** Kažemo da je $T(N) = \Omega(g(N))$ ako postoje pozitivne konstante c i n_0 takve da za sve $N \geq n_0$ važi $T(N) \geq cg(N)$.
- **Veliko-Theta (Big-Theta) – Precizna asimptotska granica:** Kažemo da je $T(N) = \Theta(h(N))$ ako i samo ako važi i $T(N) = O(h(N))$ i $T(N) = \Omega(h(N))$, odnosno ako funkcija $T(N)$ ima isto asimptotsko ponašanje kao $h(N)$.
- **Malo-o (Little-oh) – Zanemarljivo u odnosu na $p(N)$ za velike N:** Kažemo da je $T(N) = o(p(N))$ ako za svaku pozitivnu konstantu c postoji n_0 takvo da za sve $N > n_0$ važi $T(N) < cp(N)$. Neformalno, $T(N) = o(p(N))$ ako je $T(N) = O(p(N))$, ali $T(N) \neq \Theta(p(N))$; odnosno, $T(N)$ raste znatno sporije od $p(N)$ kada N teži beskonačnosti.

Uporedni rast funkcija

Osnovna ideja ovih definicija je uspostavljanje relativnog reda među funkcijama koje opisuju složenost algoritama. Često, za male vrijednosti N, jedna funkcija može imati veće vrijednosti od druge, ali kako N raste, njihov odnos se mijenja. Zbog toga se ne porede funkcije direktno ($f(N) < g(N)$), već se analizira njihova brzina rasta u granici kada N teži beskonačnosti. U daljem tekstu koristićemo isključivo notaciju *veliko-O* (Big-Oh), koja daje gornju asimptotsku granicu složenosti algoritama. Na primjer, iako funkcija $c \cdot N$ može biti veća od N^2 za manje vrijednosti N, kvadratna funkcija N^2 raste brže i za dovoljno velika N postaje dominantna. Ova tačka preloma zavisi od konkretnih konstanti, ali za asimptotsku analizu konstante zanemarujemo.

Primjer: Za $f(N) = 100N$ i $g(N) = N^2$, postoji n_0 (u ovom slučaju $n_0 = 100$) tako da za sve $N \geq n_0$ važi $f(N) < g(N)$.

U tabeli 4.1 prikazane su najčešće korišćene funkcije i odgovarajući nazivi:

Pravila kombinovanja složenosti

U praksi se često analiziraju algoritmi koji kombinuju više dijelova. Ako su $T_1(N) = O(f(N))$ i $T_2(N) = O(g(N))$, važe sljedeća pravila kombinovanja složenosti:

- **Sabiranje složenosti:** $T_1(N) + T_2(N) = O(\max(f(N), g(N)))$. [1]
- **Proizvod složenosti:** $T_1(N) \cdot T_2(N) = O(f(N) \cdot g(N))$.
- Svaki polinom stepena k je $O(N^k)$.
- Svaka logaritamska funkcija raste sporije od bilo koje linearne: $\log^k N = O(N)$.

[1] U praksi se ovo često pojednostavljuje tako da je ukupna složenost određena funkcijom koja brže raste.

Funkcija	Naziv
c	Konstanta
$\log N$	Logaritamska
$\log^2 N$	Kvadrat logaritma (log-squared)
N	Linearna
$N \log N$	Linearno-logaritamska
N^2	Kvadratna
N^3	Kubna
N^k	Polinomska (stepen k)
2^N	Eksponencijalna
$N!$	Faktorijelna

Tabela 4.1: Tipične funkcije rasta složenosti [3]

Oznake i pretpostavke

U nastavku su navedene sve promjenjive, oznake i pretpostavke koje će biti korišćene prilikom analize složenosti SLAM algoritama.

Promjenjive:

- k – broj čestica u filteru (kod Particle Filter algoritama)
- m – broj obilježja (landmarkova) u mapi
- z – broj opservacija (mjerenja) po vremenskom koraku
- n – dimenzija stanja (m + dimenzije vektora pozicije robota)

Svaki SLAM algoritam se sastoji iz sledećih koraka:

- **Inicijalizacija** – Postavljanje početnih vrijednosti pozicije robota i obilježja u mapi.
- **Predikcija** – Ažuriranje procjene pozicije robota na osnovu modela kretanja i izvršenih akcija.
- **Korekcija** – Ažuriranje procjene stanja na osnovu novih mjerenja (opservacija) i modela mjerenja.
- **Asocijacija podataka** – Povezivanje dobijenih mjerenja sa postojećim obilježjima u mapi.

Takođe, u svim analizama podrazumijeva se da su osnovne aritmetičke i matematičke operacije (sabiranje, množenje, trigonometrijske funkcije itd.) konstantne složenosti, osim ako nije posebno naznačeno.

4.1.1 Inicijalizacija

Inicijalizacija obuhvata određivanje početne pozicije i orijentacije robota, postavljanje početnih vrijednosti za obilježja u mapi (ako su poznata), te inicijalizaciju

pripadajućih kovarijansi i struktura podataka potrebnih za dalji rad algoritma.

Kalman SLAM

Inicijalizacija EKF/UKF SLAM algoritma podrazumijeva postavljanje početnih vrijednosti za vektor stanja i pripadajuću kovarijacionu matricu. U vektor stanja se unosi početna pozicija i orijentacija robota. Ukoliko je broj obilježja unaprijed poznat, formira se potpuni vektor stanja i kovarijaciona matrica odgovarajućih dimenzija, čime se rezerviše prostor za njihove vrijednosti. U suprotnom, obilježja se dodaju dinamički, prilikom njihovog uočavanja tokom rada algoritma.

Ako je broj obilježja unaprijed poznat, moguća je potpuna inicijalizacija na početku, uz sljedeće složenosti [51]:

- **Postavljanje početne pozicije robota:** vremenska i memorijska složenost su konstantne, tj. $O(1)$, jer se inicijalizuje samo pozicija i orijentacija robota (x, y, θ u 2D prostoru).
- **Inicijalizacija kovarijacione matrice robota:** takođe ima složenost $O(1)$, jer se popunjava samo gornji lijevi podblok dimenzija 3×3 u matrici kovarijanse.
- **Postavljanje unakrsnih kovarijansi:** sve unakrsne kovarijanse između pozicije robota i obilježja, kao i međusobne kovarijanse između obilježja, inicijalno se postavljaju na nulu sa složenošću $O(m^2)$.
- **Postavljanje početnih kovarijansi za obilježja:** ukoliko ne postoje prethodne informacije o obilježjima, njihove kovarijanse se postavljaju na visoke (npr. beskonačne) vrijednosti, što ima vremensku složenost $O(m)$.

Ukupna složenost inicijalizacije:

- **Vremenska složenost:** $O(m^2)$, pri čemu se u optimizovanim implementacijama može svesti na $O(m)$.
- **Memorijska složenost:** $O(m^2)$, zbog činjenice da kovarijaciona matrica dimenzija $(3 + 2m) \times (3 + 2m)$ mora biti rezervisana i inicijalizovana.

Particle SLAM

Inicijalizacija Particle SLAM algoritama FastSLAM 1.0, FastSLAM 2.0 i uFastSLAM zasniva se na generisanju skupa od k čestica, gdje svaka čestica sadrži:

- Početnu poziciju robota,
- Inicijalnu težinu,
- Strukturu za predstavljanje obilježja mape, koja zavisi od konkretne varijante algoritma (EKF za FastSLAM 1.0 i 2.0, UKF za uFastSLAM).

Ukoliko je broj obilježja unaprijed poznat, u svakoj čestici se rezerviše prostor za m Kalmanovih filtera, po jedan za svako obilježje. U suprotnom, filteri se dinamički dodaju tokom rada algoritma, pri čemu se novo obilježje inicijalizuje u trenutku kada se prvi put uoči, na osnovu pridruženog mjerenja. Proces inicijalizacije u Particle SLAM algoritmima obuhvata sljedeće korake [52]:

- **Generisanje skupa čestica:** kreira se k čestica, pri čemu svaka dobija početnu poziciju robota i jednaku inicijalnu težinu (najčešće $\frac{1}{k}$). Ovo ima složenost $O(k)$.
- **Postavljanje početnih pozicija robota:** za svaku česticu se inicijalizuje početna pozicija robota (koordinatni početak), što ukupno ima složenost $O(k)$.
- **Rezervacija memorije za strukture obilježja:** u svakoj čestici se rezerviše memorija za m Kalmanovih filtera (EKF ili UKF, u zavisnosti od algoritma). Ukupna memorijska složenost za sve čestice je $O(k \cdot m)$.
- **Inicijalizacija interne strukture filtera:** kod FastSLAM 1.0 i 2.0 to podrazumijeva EKF parametre (srednju vrijednost i kovarijansu), dok u uFastSLAM-u uključuje i dodatnu memoriju za sigma tačke, što povećava konstantni faktor unutar složenosti $O(k \cdot m)$.

Ukupna složenost inicijalizacije:

- **Vremenska složenost:** $O(k \cdot m)$, jer se za svaku od k čestica inicijalizuje do m lokalnih filtera za obilježja. Inicijalizacija pozicije robota ima konstantnu složenost i zanemarljiva je u odnosu na inicijalizaciju obilježja.
- **Memorijska složenost:** $O(k \cdot m)$, pošto svaka čestica čuva svoju verziju procjene mape sa do m Kalmanovih filtera (EKF/UKF), nezavisno od ostalih čestica. Budući da je memorijski zahtjev inicijalizacije pozicije robota fiksan i mali, ukupna složenost je uslovljena brojem obilježja.

4.1.2 Predikcioni korak

U predikcionom koraku se na osnovu modela kretanja i upravljačkih ulaza ažurira procjena trenutnog stanja robota. U zavisnosti od tipa SLAM algoritma, ovaj korak može obuhvatiti propagaciju kovarijacione matrice, generisanje sigma tačaka i/ili uzorkovanje skupa čestica.

EKF SLAM

Predikcioni korak EKF SLAM algoritma uključuje ažuriranje pozicije robota i kovarijacione matrice prema modelu kretanja. Složenost ovog koraka zavisi od broja obilježja, a sastoji se od sljedećih komponenti:

- **Vektor stanja:** Ažuriranje vektora stanja μ_t ima konstantnu složenost $O(1)$, jer se ažurira samo dio koji se odnosi na poziciju robota (3×3 matrica).
- **Jakobijan:** Konstrukcija Jakobijana G_t ima vremensku i memorijsku složenost $O(m^2)$, jer se popunjava matrica dimenzija $(3 + 2m) \times (3 + 2m)$.
- **Kovarijaciona matrica** Ažuriranje Σ_t je najzahtjevniji dio i uključuje množenje matrica dimenzije $(3 + 2m) \times (3 + 2m)$. To rezultira vremenskom složenošću $O(m^3)$ [51, 53, 54].

Ukupna složenost predikcionog koraka:

- **Vremenska složenost:** $O(m^3)$, zbog množenja velikih matrica prilikom ažuriranja kovarijanse.
- **Memorijska složenost:** $O(m^2)$, jer se mora čuvati matrica kovarijanse dimenzija $(3 + 2m) \times (3 + 2m)$.

UKF SLAM

Predikcioni korak UKF SLAM algoritma koristi Unscented Transform za propagaciju nelinearnog modela kretanja kroz distribuciju stanja. Ovaj proces uključuje sljedeće korake:

- **Generisanje sigma tačaka:** Oko trenutne srednje vrijednosti stanja μ_{t-1} generiše se $2n + 1$ sigma tačaka, gdje je n dimenzija stanja (pozicija robota i svih obilježja). Ovo uključuje izračunavanje Čoleskijeve dekompozicije matrice kovarijanse Σ_{t-1}, što ima vremensku složenost $O(n^3)$. Generisanje i skaliranje sigma tačaka zatim zahtijeva $O(n^2)$ operacija [51, 53, 55].
- **Propagacija sigma tačaka:** Svaka od $2n+1$ sigma tačaka se propagira kroz nelinearni model kretanja robota, koji ima složenost $O(1)$ po tački. Ukupna složenost ovog koraka je stoga $O(n)$.
- **Rekonstrukcija predikcije:**
  - Izračunavanje nove srednje vrijednosti stanja se vrši ponderisanim sumiranjem $2n + 1$ transformisanih tačaka dimenzije n, što ima složenost $O(n^2)$.
  - Izračunavanje kovarijacione matrice zahtijeva $2n+1$ *outer* proizvoda vektora dimenzije n, što rezultira vremenskom složenošću $O(n^3)$.

Ukupna složenost predikcionog koraka:

- **Vremenska složenost:** $O(n^3) \approx O(m^3)$, dominirana Čoleskijevom dekompozicijom i računanjem kovarijanse, pri čemu je $n = 3 + 2m \approx O(m)$ za veće vrijednosti m.

- **Memorijska složenost:** $O(n^2) \approx O(m^2)$, jer se čuva $2n + 1$ sigma tačaka dimenzije n i kovarijaciona matrica dimenzije $n \times n$. Za veći broj obilježja, složenost se može aproksimirati kao $O(m^2)$.

FastSLAM 1.0

U FastSLAM 1.0 algoritmu, predikcija se obavlja nezavisno za svaku od k čestica, i zasniva se na uzorkovanju pozicije robota iz modela kretanja. Kako se operacije primjenjuju lokalno po čestici, složenost predikcionog koraka zavisi isključivo od njihovog ukupnog broja:

- **Uzorkovanje iz modela kretanja:** Nova pozicija robota se u svakoj čestici generiše uzorkovanjem iz modela kretanja. Ova operacija koristi osnovne matematičke funkcije i ima konstantnu vremensku složenost $O(1)$ [52].

Ukupna složenost predikcionog koraka:

- **Vremenska složenost:** $O(k)$, jer se uzorkovanje robota vrši nezavisno u svakoj od k čestica.
- **Memorijska složenost:** $O(k)$, pošto svaka čestica čuva samo poziciju robota kao vektor konstantne dimenzije.

FastSLAM 2.0

Predikcioni korak FastSLAM 2.0 algoritma zasniva se na propagaciji pozicije robota unutar svake čestice korišćenjem modela kretanja i dostupnih mjerenja. Budući da se ovaj proces izvodi nezavisno za svaku od k čestica, složenost koraka zavisi od njihovog ukupnog broja i obuhvata sljedeće komponente:

- **Jakobijani:** Za svaku česticu se izračunavaju Jakobijani modela kretanja — G_v u odnosu na poziciju robota (dimenzija 3×3), i G_u u odnosu na upravljačke ulaze (dimenzija 3×2). Budući da su ove dimenzije fiksne, složenost ovih operacija je konstantna, tj. $O(1)$.
- **Kovarijaciona matrica:** Ažuriranje lokalne matrice kovarijanse obuhvata množenje i sabiranje fiksnih matrica dimenzije 3×3, zbog čega ova operacija ima konstantnu vremensku složenost $O(1)$.
- **Vektor stanja:** Predikcija nove pozicije robota uključuje evaluaciju modela kretanja na osnovu ulaza i prethodnog stanja, što takođe ima složenost $O(1)$.
- **Predložena distribucija:** U svakoj čestici se nova pozicija robota uzorkuje iz *optimalne* distribucije. Konstrukcija ove distribucije uključuje obradu svih pridruženih mjerenja i ima složenost $O(z)$, dok samo uzorkovanje iz konačne Gaussove distribucije ima složenost $O(1)$.

Ukupna složenost predikcionog koraka: [52]

- **Vremenska složenost:** $O(k \cdot z)$, jer se u svakoj od k čestica konstruiše predložena distribucija na osnovu z pridruženih mjerenja i potom vrši uzorkovanje.
- **Memorijska složenost:** $O(k)$, jer svaka čestica čuva poziciju robota i kovarijacionu matricu konstantnih dimenzija.

uFastSLAM

Predikcioni korak uFastSLAM algoritma koristi Unscented Transform za propagaciju nelinearnog modela kretanja kroz distribuciju pozicija robota. Ovaj proces se izvodi nezavisno u svakoj od k čestica i uključuje sljedeće komponente [50]:

- **Augmentacija stanja:** Za svako od z trenutno pridruženih mjerenja, formira se prošireno stanje koje uključuje poziciju robota i odgovarajuće obilježje. Pošto se obrađuju samo pridružena obilježja (a ne cijela mapa), složenost ovog koraka po čestici iznosi $O(z)$.
- **Generisanje i propagacija sigma tačaka:** Za svako prošireno stanje generišu se sigma tačke i transformišu kroz nelinearni model mjerenja. Ovaj proces se ponavlja z puta, pa ima složenost $O(z)$, uz napomenu da je konstantni faktor značajan zbog generisanja i obrade $2n+1$ sigma tačaka za svako prošireno stanje.
- **Računanje kovarijanse inovacije i kros-kovarijanse:** Na osnovu sigma tačaka, računa se inovaciona kovarijacija i kros-kovarijacija između pozicije robota i predikcije mjerenja, što priprema osnovu za računanje Kalmanovog pojačanja. Pošto se za svako od z mjerenja obrađuje fiksni broj sigma tačaka, ukupna složenost ovog koraka po čestici iznosi $O(z)$.

Ukupna složenost predikcionog koraka:

- **Vremenska složenost:** $O(k \cdot z)$, jer se svi koraci izvršavaju nezavisno u svakoj od k čestica i proporcionalni su broju pridruženih mjerenja z.
- **Memorijska složenost:** $O(k)$, jer svaka čestica čuva samo jedan predložni uzorak i pomoćne matrice fiksnih dimenzija.

4.1.3 Korekcioni korak

Korekcioni korak kod Kalman SLAM-a obuhvata izračunavanje inovacije, Kalmanovog pojačanja, te korekciju vektora stanja i kovarijacione matrice. Kod particle filter pristupa, korekcioni korak uključuje, ažuriranje procjena obilježja u svakoj čestici, izračunavanje težinskih faktora čestica i ponovno odabiranje (*resampling*).

EKF SLAM

Korekcioni korak EKF SLAM algoritma podrazumijeva ažuriranje procjene pozicije robota i obilježja na osnovu novih mjerenja. Ovaj proces uključuje proračun inovacije, Kalmanovog pojačanja, te ažuriranje vektora stanja i kovarijacione matrice. Složenost korekcije zavisi od broja uočenih i ukupnih obilježja u stanju, i sastoji se od sljedećih komponenti [51, 53, 54]:

- **Inovacija:** Za svaku opservaciju računa se razlika između mjerenja i predikcije, što uključuje transformacije i izračunavanje očekivanog mjerenja. Vrijeme potrebno za ovu operaciju je $O(z \cdot m)$.
- **Kalmanovo pojačanje i ažuriranje:** Simultano ažuriranje svih obilježja zahtijeva množenje matrica dimenzije $(3+2m) \times (3+2m)$, što daje vremensku složenost $O(m^3)$.
- **Kovarijaciona matrica:** Memorijska složenost određena je dimenzijom kovarijacione matrice Σ_t, koja je veličine $(3+2m) \times (3+2m)$, što daje memorijsku složenost $O(m^2)$.

Ukupna složenost korekcionog koraka:

- **Vremenska složenost:** $O(m^3)$, zbog ažuriranja kovarijacione matrice dimenzije $(3+2m) \times (3+2m)$.[2]
- **Memorijska složenost:** $O(m^2)$, zbog čuvanja matrice dimenzije $(3+2m) \times (3+2m)$.

UKF SLAM

Računarska složenost korekcionog koraka UKF SLAM algoritma prvenstveno zavisi od dimenzije proširenog vektora stanja $n = 3 + 2m$, kao i od broja opservacija z. Korekcioni korak se sastoji od sljedećih komponenti [51, 53, 55]:

- **Generisanje sigma tačaka:** Zahtijeva Čoleski dekompoziciju kovarijacione matrice dimenzije $n \times n$, što ima vremensku složenost $O(n^3)$.
- **Predikcija mjerenja:** Za svaku od $2n + 1$ sigma tačaka predviđa se odgovarajuća opservacija, što ima ukupnu složenost $O(n \cdot z)$.
- **Srednja vrijednost i kovarijansa mjerenja:** Računanje ponderisane sredine ima složenost $O(n)$, a kovarijanse mjerenja $O(z^2)$.
- **Međukovarijansa:** Računanje međukovarijanse između sigma tačaka i predikcija mjerenja ima složenost $O(n \cdot z)$.
- **Kalmanovo pojačanje:** Dobija se rješavanjem sistema sa kovarijansom mje-

[2] U praksi se češće koristi inkrementalna metoda, gdje se svaka od z opservacija obrađuje zasebno. Na taj način ukupna složenost se smanjuje na $O(z \cdot m^2)$, uz povećanje numeričke stabilnosti [10,37].

renja i međukovarijansom, što ima složenost $O(z^2 \cdot n)$.

- **Ažuriranje stanja i kovarijacije:** Vektor stanja se ažurira u $O(n)$, a kovarijacija u $O(n^2)$.

Ukupna složenost korekcionog koraka:

- **Vremenska složenost:** $O(n^3) \approx O(m^3)$, zbog dominacije Čoleski dekompozicije pri generisanju sigma tačaka.[3]
- **Memorijska složenost:** $O(n^2) \approx O(m^2)$, zbog skladištenja kovarijacione matrice dimenzije $n \times n$.

FastSLAM 1.0

Korekcioni korak FastSLAM 1.0 algoritma podrazumijeva ažuriranje težinskih faktora na osnovu novih mjerenja, kao i ažuriranje procjene obilježja unutar svake čestice. Svaka čestica posjeduje sopstvene procjene obilježja, implementirane kao lokalni EKF-ovi, čime se omogućava nezavisno ažuriranje svakog obilježja bez uvođenja globalne korelacije među njima [52, 53].

- **Težinski faktori:** Za svaku česticu k, računa se vjerovatnoća posmatranog mjerenja z_t u odnosu na trenutno stanje čestice $x_t^{[k]}$ i procjenu obilježja $m_{c_t}^{[k]}$. To uključuje izračunavanje Mahalanobisove udaljenosti između stvarnog i predviđenog mjerenja, kao i kovarijacione matrice predikcije. Budući da se ovo radi za z opservacija u k čestica, složenost je $O(k \cdot z)$.
- **Ažuriranje obilježja:** Kada je obilježje c_t već posmatrano, njegov lokalni EKF se ažurira u svakoj čestici korišćenjem standardnih EKF pravila. Ažuriranje se sprovodi samo za trenutno osmotrena obilježja, čime se izbjegava zavisnost od ukupnog broja obilježja m. Složenost ostaje $O(k \cdot z)$.
- **Resamplovanje:** Nakon što su težine normalizovane, vrši se ponovni odabir čestica proporcionalno njihovim težinama. Ova operacija se sprovodi u $O(k)$ vremenskoj složenosti.

Ukupna složenost korekcionog koraka:

- **Vremenska složenost:** $O(k \cdot z)$, jer se za svaku od k čestica obrađuje do z opservacija putem evaluacije vjerovatnoće i lokalnog EKF ažuriranja.
- **Memorijska složenost:** $O(k \cdot m)$, jer svaka čestica čuva do m lokalnih EKF instanci (srednja vrijednost i kovarijancu) za obilježja.

[3] Za razliku od EKF SLAM-a, gdje se korekcioni korak može efikasno izvesti inkrementalno za svaku opservaciju posebno, UKF SLAM koristi Unscented Transform koji zahtijeva propagaciju cijele distribucije sigma tačaka kroz nelinearnu funkciju. Zbog toga nije moguće jednostavno izolovati pojedina obilježja, jer bi parcijalna ažuriranja zahtijevala nezavisno generisanje novih sigma tačaka za svaki podskup, što bi poništilo računsku uštedu. [10, 41]

FastSLAM 2.0

Korekcioni korak FastSLAM 2.0 algoritma uključuje uzorkovanje pozicije robota iz poboljšane predložne distribucije, ažuriranje obilježja putem lokalnih EKF-ova i izračunavanje težinskih faktora. Ili detaljnije [52, 53]:

- **Ažuriranje obilježja:** Lokalni EKF-ovi u svakoj čestici se ažuriraju za z opservacija. Budući da se ažuriranje vrši samo za trenutno posmatrana obilježja, složenost je $O(k \cdot z)$.
- **Težinski faktori:** Za svaku česticu se procjenjuju vjerovatnoće z opservacija na osnovu nove pozicije i ažuriranih obilježja. Proračun uključuje Mahalanobisovu udaljenost, što daje složenost $O(k \cdot z)$.
- **Resamplovanje:** Nakon normalizacije težinskih faktora, vrši se ponovni odabir čestica proporcionalno njihovim težinama. Ova operacija ima složenost $O(k)$.

Ukupna složenost korekcionog koraka:

- **Vremenska složenost:** $O(k \cdot z)$, jer se za svaku od k čestica obrađuje do z opservacija kroz uzorkovanje pozicije, ažuriranje obilježja i izračunavanje težinskih faktora.
- **Memorijska složenost:** $O(k \cdot m)$, jer svaka čestica čuva do m lokalnih EKF instanci (srednja vrijednost i kovarijancu) za obilježja.

uFastSLAM

Korekcioni korak uFastSLAM algoritma koristi Unscented Transform u svakoj čestici radi ažuriranja procjene pozicije robota i obilježja na osnovu novih mjerenja. Korekcioni korak obuhvata sljedeće komponente [50]:

- **Augmentacija stanja:** Za svaku česticu, stanje se proširuje uključivanjem pozicije robota i obilježja pridruženih za z opservacija. Ova augmentacija ima složenost $O(z)$ po čestici.
- **Sigma tačke:** Za svako prošireno stanje generišu se sigma tačke i propagiraju kroz nelinearni model mjerenja. Postupak se ponavlja z puta po čestici, sa složenošću $O(z)$, uz značajan konstantni faktor zbog obrade $2n + 1$ sigma tačaka.
- **Inovacija i kovarijanse:** Na osnovu propagiranih sigma tačaka, računaju se inovacija, kovarijansa mjerenja i međukovarijansa između stanja i mjerenja, što omogućava računanje Kalmanovog pojačanja i ažuriranje procjene. Složenost ovog koraka je $O(z)$ po čestici.
- **Ažuriranje stanja i kovarijacije:** Svaka čestica ažurira svoje procjene stanja

i kovarijacione matrice za pridružena obilježja nezavisno, sa složenošću $O(z)$ po čestici.

- **Težinski faktori:** Za svaku česticu se izračunava vjerovatnoća posmatranih mjerenja na osnovu ažuriranog stanja i obilježja, uključujući evaluaciju Gaussove gustine. Složenost je $O(k \cdot z)$.
- **Resamplovanje:** Nakon normalizacije težinskih faktora, vrši se ponovni odabir čestica proporcionalno njihovim težinama, u vremenskoj složenosti $O(k)$.

Ukupna složenost korekcionog koraka:

- **Vremenska složenost:** $O(k \cdot z)$, jer se svi koraci izvršavaju nezavisno u svakoj od k čestica i proporcionalni su broju pridruženih mjerenja z.
- **Memorijska složenost:** $O(k \cdot m)$, pošto svaka čestica čuva do m lokalnih UKF-ova (srednja vrijednost i kovarijancu) za obilježja.

4.1.4 Asocijacija mjerenja

Gated Nearest-Neighbor (GNN) metoda za asocijaciju podataka u EKF SLAM algoritmu funkcioniše tako što za svaku opservaciju prolazi kroz sva trenutno poznata obilježja, izračunavajući očekivanu poziciju obilježja i Mahalanobis-ovu udaljenost između mjerenja i predikcije. Ovaj postupak obuhvata sljedeće korake [38]:

- Za z opservacija i m obilježja, ukupno se računa $z \cdot m$ Mahalanobis-ovih udaljenosti.
- Svaka udaljenost zahtijeva inverziju kovarijacione matrice dimenzije 2×2 i množenje malih vektora, što ima konstantnu složenost $O(1)$.
- Dobijeni rezultati se porede sa unaprijed definisanom graničnom vrijednošću (*gate*-ing) radi eliminacije loših podudaranja.

Ako su podudaranja između opservacija i obilježja unaprijed poznata (npr. kroz ground truth ili eksterni algoritam), ovaj korak se izostavlja.

Ukupna složenost indeksiranja (GNN metoda):

- **Vremenska složenost:** $O(z \cdot m)$, zbog računanja svih Mahalanobis-ovih udaljenosti.
- **Memorijska složenost:** $O(z)$, jer se čuva informacija o najboljem podudaranju za svaku opservaciju.

4.1.5 Sažetak računarske složenosti

U ovoj sekciji sumirani su dominantni izrazi za vremensku i memorijsku složenost glavnih SLAM algoritama razmatranih u radu. Prikazane vrijednosti odgovaraju

najzahtjevnijem koraku unutar svakog algoritma, jer on određuje ukupne zahtjeve implementacije pri velikom broju obilježja m, čestica k i opservacija z.

Algoritam	**Vremenska složenost**	**Memorijska složenost**	**Dominantna komponenta**
EKF SLAM	$O(m^3)$	$O(m^2)$	Korekcija (ažuriranje kovarijacije)
UKF SLAM	$O(m^3)$	$O(m^2)$	Korekcija (sigma tačke, Čoleski)
FastSLAM 1.0	$O(k \cdot m)$	$O(k \cdot m)$	Korekcija (ažuriranje i težine)
FastSLAM 2.0	$O(k \cdot m)$	$O(k \cdot m)$	Korekcija (predložena distribucija)
uFastSLAM	$O(k \cdot m)$	$O(k \cdot m)$	Korekcija (UKF unutar čestica)

Tabela 4.2: Pregled dominantnih složenosti SLAM algoritama

Kao što se vidi iz tabele 4.2, EKF SLAM i UKF SLAM imaju kubnu vremensku složenost po broju obilježja m, što ih čini nepovoljnim za veoma velike mape. Particle SLAM pristupi (FastSLAM 1.0, FastSLAM 2.0, uFastSLAM) imaju linearnu složenost koja je određena proizvodom $k \cdot z$. Memorijska složenost je uvijek proporcionalna proizvodu $k \cdot m$, jer svaka čestica sadrži nezavisne procjene za sva obilježja. Budući da su u oba slučaja promjenjive nezavisne, ukupna složenost je drugog reda.

Dominantni faktor složenosti zavisi od konkretne implementacije, broja obilježja i zahtjeva aplikacije. Kod EKF/UKF SLAM-a, ažuriranje kovarijacione matrice (korekcioni korak) najviše doprinosi ukupnoj složenosti, dok kod Particle SLAM algoritama dominiraju korekcioni korak i upravljanje nezavisnim filterima obilježja u svakoj čestici.

Napomena: U tabeli 4.2 su prikazane neoptimizovane vrijednosti složenosti (tzv. "raw" algoritmi). U praksi, moguće su dodatne optimizacije koje poboljšavaju efikasnost algoritma, često bez narušavanja tačnosti ili konzistentnosti procjene [56–58]. U tabeli 4.3 prikazane su optimizovane vremenske i memorijske složenosti različitih SLAM algoritama. Ove složenosti su rezultat primjene različitih optimizacijskih tehnika koje smanjuju računarske zahtjeve bez značajnog kompromisa u tačnosti.

Algoritam	**Vremenska složenost**	**Memorijska složenost**	**Dominantna komponenta**
EKF SLAM	$O(z \cdot m^2)$	$O(m^2)$	Korekcija (ažuriranje kovarijacije)
UKF SLAM	$O(z \cdot m^2)$	$O(m^2)$	Korekcija (sigma tačke, Čoleski)
FastSLAM 1.0	$O(k \cdot \log m)$	$O(k \cdot m)$	Korekcija (ažuriranje i težine)
FastSLAM 2.0	$O(k \cdot \log m)$	$O(k \cdot m)$	Korekcija (predložena distribucija)
uFastSLAM	$O(k \cdot \log m)$	$O(k \cdot m)$	Korekcija (UKF unutar čestica)

Tabela 4.3: Optimizovane složenosti SLAM algoritama

Prikazane složenosti odražavaju optimizovane implementacije algoritama. U praksi, različite tehnike kao što su parcijalna ažuriranja, korišćenje svojstva rijetkosti ma-

trica, i hijerarhijske strukture podmapa koriste se za smanjenje računarskih zahtjeva bez značajnog kompromisa u tačnosti [56–58].

Glava 5

Rezultati simulacija

U ovom poglavlju prikazani su rezultati simulacija kojima se ilustruje praktično ponašanje različitih SLAM algoritama. Primarni cilj simulacija jeste poređenje tačnosti i robusnosti algoritama u uslovima koji približno odgovaraju realnim scenarijima rada mobilnog robota. Dodatno, algoritmi su analizirani i u zahtjevnijim konfiguracijama parametara i šumova, s ciljem ispitivanja granica njihove stabilnosti i pouzdanosti.

Analizom su obuhvaćeni algoritmi EKF-SLAM, UKF-SLAM, FastSLAM 1.0, FastSLAM 2.0 i uFastSLAM. Sve simulacije sprovedene su u MATLAB-u R2022b, koji je korišćen kao standardno okruženje za implementaciju i testiranje. U eksperimentima su razmatrani: različiti scenariji kretanja robota, različite vrijednosti parametara algoritama, promjene broja čestica (za particle filter algoritme), varijacije kovarijanse procesnog i mjernog šuma, promjene dometa senzora, kao i različite konfiguracije mape. Kao indikatori performansi korišćeni su korijen srednje kvadratne greške (RMSE), zasebno izračunate za procjenu pozicije robota i za procjenu obilježja, a za Kalmanove pristupe (EKF-SLAM i UKF-SLAM)[1] dodatno su analizirane i vrijednosti normalizovane greške procjene (NEES), radi ocjene konzistentnosti filtera.

NEES (engl. *Normalized Estimation Error Squared*) je mjera konzistentnosti estimatora koja pokazuje u kojoj mjeri stvarna greška procjene odgovara kovarijansi procijenjenoj od strane estimatora. Formalno, za stanje robota x_t, njegovu procjenu $\hat{x}_t$ i procijenjenu kovarijansu P_t, NEES se definiše na sljedeći način [60]:

$$\text{NEES}_t = (x_t - \hat{x}_t)^T P_t^{-1} (x_t - \hat{x}_t). \tag{5.1}$$

[1]NEES nije primjenljiv na particle filter algoritme, jer oni za svaku česticu održavaju poseban EKF/UKF nad obilježjima. Budući da se kovarijanse ovih lokalnih filtera međusobno razlikuju, ne postoji jedinstvena kovarijansa sistema na osnovu koje bi se mogla izračunati globalna vrijednost NEES-a. [59]

NEES pokazuje da li je procijenjena nesigurnost (kovarijansa P_t) u skladu sa stvarnom greškom procjene. Ako je prosječna vrijednost NEES-a približna dimenziji stanja, estimator se smatra statistički konzistentnim. Radi poređenja i analize performansi estimatora koristi se normalizovani oblik NEES-a, čija je očekivana vrijednost bliska jedinici [60].

Vrijeme izvršavanja simulacija nije razmatrano u okviru ove analize, budući da ono u značajnoj mjeri zavisi od implementacionih detalja[2]. Iz tog razloga, umjesto empirijskog mjerenja trajanja simulacija, u prethodnom poglavlju analizirana je vremenska i memorijska složenost, pri čemu su algoritmi upoređeni nezavisno od implementacionih detalja.

5.1 Postavka simulacija

U simulacijama je razmatrano kretanje robota po putanji zadatoj nizom unaprijed definisanih tačaka, počevši iz koordinatnog početka i vraćajući se do njega dvaput. Mape i pripadajuće putanje koje su korišćene u simulacijama su opisane u nastavku.

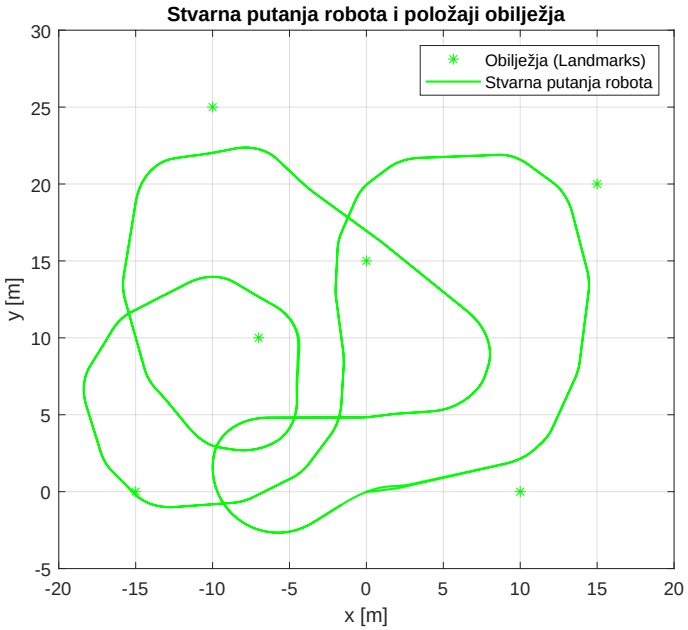
5.1.1 Mapa

U skladu sa definicijom iz Poglavlja 3, mape se predstavljaju kao skup jednoznačno definisanih i međusobno nezavisnih tačkastih obilježja. Za potrebe simulacija korišćene su tri mape, koje se razlikuju po dimenzijama prostora i rasporedu obilježja, dok je na svim mapama naznačena stvarna putanja robota.

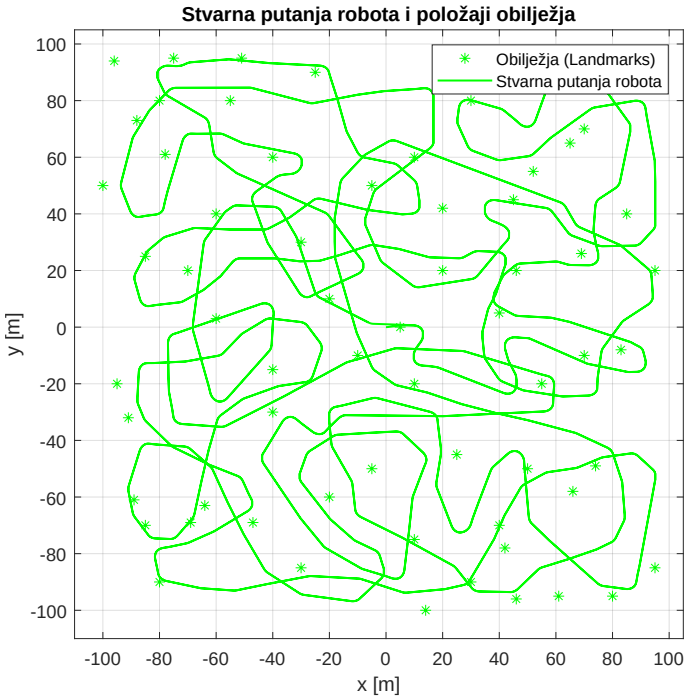
- **Mala mapa:** sastoji se od 6 obilježja raspoređenih na površini od 15×25 metara. Ova konfiguracija pogodna je za brze testove i procjenu osnovne funkcionalnosti algoritama, pri čemu mali broj obilježja ograničava količinu informacija za SLAM problem. Konfiguracija male mape, zajedno sa pripadnom putanjom, prikazana je na slici 5.1.

- **Velika mapa:** sadrži 60 obilježja raspoređenih na površini od 200×200 metara. Ova mapa se svojim dimenzijama i brojem obilježja približava realnim okruženjima u kojima se SLAM algoritmi primjenjuju. Konfiguracija velike mape, zajedno sa pripadnom putanjom, prikazana je na slici 5.2.

[2]Način na koji MATLAB upravlja memorijom podrazumijeva da svaka promjena stanja i/ili čestica zahtijeva kopiranje velikih struktura, što predstavlja dominantan izvor usporenja. Ovo ograničenje proizlazi iz same prirode platforme i ne može se otkloniti bez prelaska na drugačije programsko okruženje, pa bi uključivanje mjerenja vremena rezultiralo pogrešnom interpretacijom stvarnih razlika u računarskoj složenosti algoritama.

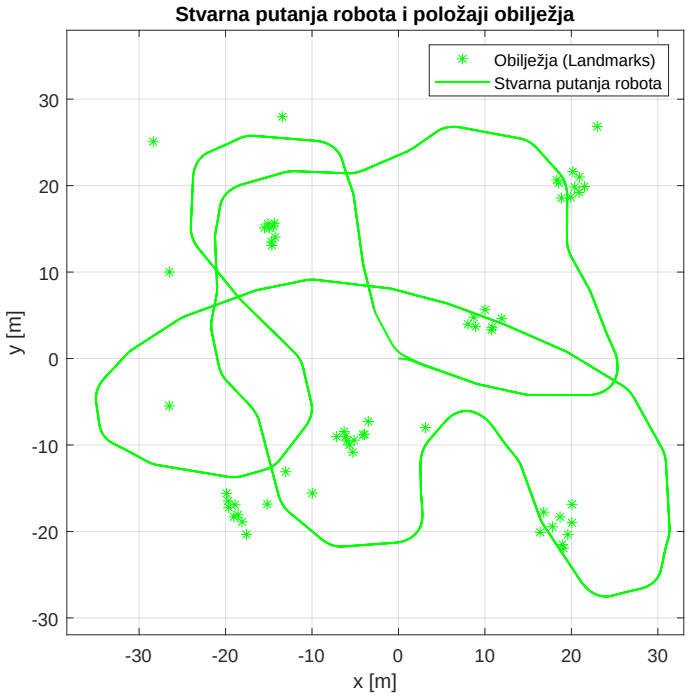


Slika 5.1: Stvarna putanja robota i položaji obilježja na maloj mapi



Slika 5.2: Stvarna putanja robota i položaji obilježja na velikoj mapi

- **Cluster mapa:** također sadrži 60 obilježja, ali na znatno manjoj površini od 50×55 metara. Obilježja su koncentrisana u grupe (*clusters*), što ovu mapu čini korisnom za testiranje skalabilnosti i numeričke stabilnosti filtera u okruženju sa visokom gustinom obilježja. Konfiguracija *cluster* mape, zajedno sa pripadnom putanjom, prikazana je na slici 5.3.



Slika 5.3: Stvarna putanja robota i položaji obilježja na *cluster* mapi

5.1.2 Parametri simulacija

Fiksne vrijednosti parametara korišćene u svim simulacijama prikazane su u tabeli 5.1, dok su promjenjivi parametri, koji zavise od scenarija, navedeni u tabeli 5.2.

Oznaka	**Opis**	**Vrijednost**	**Jedinica**
v	Brzina	3	m/s
$\delta_{\max}$	Maksimalni ugao upravljanja	± 45	$^\circ$
$\dot{\delta}_{\max}$	Maksimalna promjena ugla upravljanja	30	$^\circ$/s
L	Razmak između prednje i zadnje osovine	3	m
$\Delta t_{\text{controls}}$	Vremenski korak upravljanja	0.025	s
Δt_{obs}	Period senzorskih mjerenja	0.2	s
d_{goal}	Udaljenost do tačke vođenja[3]	2	m
N_{eff}	Minimalni broj efektivnih čestica	$0.75N$	–
N	Ukupan broj čestica	–	–
(α, β, κ)	Parametri unscented transformacije	(0.9, 2, 0)	–

Tabela 5.1: Fiksni parametri u simulacijama

Realizacije šuma generisane su pomoću MATLAB-ovog podrazumijevanog generatora slučajnih brojeva. Svaki algoritam je testiran kroz 30 ponavljanja na unaprijed definisanim mapama i scenarijima, a prikazani rezultati predstavljaju prosječne vrijednosti izračunate na osnovu svih ponavljanja.

[3]Označava toleranciju unutar koje se smatra da je robot dostigao zadatu tačku putanje. Parametar je uveden radi izbjegavanja oscilacija oko ciljne tačke; premala vrijednost otežava prelazak na narednu tačku, dok prevelika smanjuje preciznost praćenja.

Oznaka	Opis	Jedinica
$R_{\max}$	Maksimalni domet senzora	m
$\theta_{\max}$	Maksimalni ugao osmatranja	rad
Q	Kovarijansa procesnog šuma	–
R	Kovarijansa mjernog šuma	–
Broj krugova	Broj ponavljanja zadate putanje	–
N	Broj čestica u FastSLAM/uFastSLAM	–

Tabela 5.2: Promjenjivi parametri u simulacijama

5.2 Scenariji simulacija

Različiti scenariji simulacija uvedeni su kako bi se sistematski ispitali pojedini aspekti ponašanja SLAM algoritama. Umjesto jedinstvenog eksperimenta, primijenjena je serija odvojenih scenarija, pri čemu svaki naglašava određenu karakteristiku ili potencijalnu slabost metoda. Na taj način postiže se detaljnije razumijevanje pouzdanosti i ograničenja različitih filtracionih pristupa u uslovima koji su kontrolisani, ali ujedno i dovoljno raznovrsni da imitiraju realistične situacije.

U okviru simulacija razmatrani su sljedeći scenariji:

- **Osnovni slučaj:** predstavlja referentni scenarij u kojem se svi algoritmi testiraju u identičnim, kontrolisanim i realističnim uslovima.
- **Uticaj kovarijanse šuma:** razmatra kako različiti nivoi procesnog i mjernog šuma utiču na preciznost i konzistentnost procjene.
- **Uticaj dometa senzora:** ispituje efekte ograničene vidljivosti na performanse algoritama, posebno u okruženjima sa rjeđim obilježjima.
- **Uticaj broja čestica:** analizira uticaj broja čestica na tačnost i efikasnost particle filter algoritama.
- **Test robusnosti:** simuliraju se mjerni i procesni šumovi koji odsupaju od gausove raspdjele (bias, drift, odstupanja (engl. *outliers*)), kako bi se ispitala robusnost algoritama u slučaju kada bazne pretpostavke nisu ispunjene.

Osnovni slučaj

U osnovnom scenariju svi algoritmi se izvršavaju u identičnim i kontrolisanim uslovima, koji približno odgovaraju realnim [61, 62]. U simulacijama su korišćene fiksne vrijednosti parametara navedene u prethodnom odjeljku, uz postavke preostalih parametara datih u tabeli 5.3.

Oznaka	Opis	Vrijednost
Q	Kovarijansa procesnog šuma	$\begin{bmatrix} 0.3 & 0 \\ 0 & 0.0524 \end{bmatrix}^2$
R	Kovarijansa mjernog šuma	$\begin{bmatrix} 0.01 & 0 \\ 0 & 0.0349 \end{bmatrix}^2$
$R_{\max}$	Maksimalni domet senzora	30 m
$\theta_{\max}$	Vidni ugao senzora	240°
N	Broj čestica (FastSLAM 1.0/2.0)	100
N	Broj čestica (uFastSLAM)	10

Tabela 5.3: Parametri osnovnog slučaja

Uticaj kovarijanse šuma

U ovom scenariju analiziran je uticaj različitih nivoa procesnog i mjernog šuma na performanse SLAM algoritama. Eksperimenti su sprovedeni na *cluster* mapi, jer veliki broj obilježja omogućava da se efekti šuma jasno ispolje, posebno u slučaju većih vrijednosti mjernog šuma. Razmotrene vijednosti kovarijacionih matrica Q i R prikazane su u tabeli 5.4, dok svi ostali parametri ostaju isti kao u osnovnom slučaju.

Nivo	Procesni šum Q	Mjerni šum R
Nizak	$\begin{bmatrix} 0.3 & 0 \\ 0 & 0.0524 \end{bmatrix}^2$	$\begin{bmatrix} 0.1 & 0 \\ 0 & 0.0175 \end{bmatrix}^2$
Srednji	$\begin{bmatrix} 0.5 & 0 \\ 0 & 0.0873 \end{bmatrix}^2$	$\begin{bmatrix} 0.5 & 0 \\ 0 & 0.0436 \end{bmatrix}^2$
Visok	$\begin{bmatrix} 1.0 & 0 \\ 0 & 0.1745 \end{bmatrix}^2$	$\begin{bmatrix} 1.0 & 0 \\ 0 & 0.0873 \end{bmatrix}^2$

Tabela 5.4: Nivoi procesnog i mjernog šuma

Uticaj šuma na procjenu stanja nije linearan, jer se šum propagira kroz nelinearni kinematički model pomoću Jakobijana H i G. Zbog toga povećanje šuma ne mora dovesti do proporcionalnog pogoršanja performansi algoritma. Pored toga, procesni i mjerni šum nisu međusobno direktno uporedivi, jer opisuju različite pojave – procesni šum modeluje nesigurnost u dinamici kretanja, dok mjerni šum zavisi od preciznosti senzora.[4]

Uticaj dometa senzora

U ovom scenariju ispitan je uticaj ograničenja dometa senzora na performanse SLAM algoritama. Procesni i mjerni šum, kao i ostali parametri, isti su kao u

[4]Uticaj šuma takođe zavisi od udaljenosti i rasporeda obilježja, strukture mape i vrijednosti ostalih parametara.

osnovnom slučaju, osim što vidni ugao senzora nije ograničen, tj. $\theta_{\max} = 360°$. Eksperimenti su izvedeni na velikoj mapi, jer ređi raspored obilježja omogućava jasnije poređenje efekata različitih dometa senzora. Korišćene vrijednosti maksimalnog dometa $R_{\max}$ date su u tabeli 5.5.

Konfiguracija	**Maksimalni domet ($R_{\max}$)**
Kratak domet	20 m
Srednji domet	35 m
Širok domet	50 m

Tabela 5.5: Varijante maksimalnog dometa senzora

Uticaj broja čestica

Cilj ovog scenarija je da se ispita zavisnost performansi algoritama zasnovanih na particle filterima od broja čestica. Budući da EKF-SLAM i UKF-SLAM ne primjenjuju reprezentaciju stanja zasnovanu na česticama, promjena njihovog broja nema uticaja na rezultate, pa oni ostaju nepromijenjeni. Kod FastSLAM 1.0, FastSLAM 2.0 i uFastSLAM-a broj čestica je variran, uz primjenu niskog nivoa procesnog i mjernog šuma, dok su ostali parametri isti kao u osnovnom slučaju. Razmatrane su sljedeće vrijednosti broja čestica: $N \in \{10, 50, 100, 500\}$.

Test robusnosti

U ovom scenariju je analizrana robusnost SLAM algoritama na poremećaje koji odstupaju od teorijskih pretpostavki. Razmatrane su sljedeće situacije:

- **Student-t raspodjela šuma:** zamjena standardne Gausove raspodjele procesnog i mjernog šuma Student-t raspodjelom sa znatno težim repovima (engl. *heavy tails*). Ovaj pristup omogućava realistično modelovanje izrazito velikih grešaka u modelu kretanja i senzorskim mjerenjima koje Gausova raspodjela ne opisuje adekvatno. Student-t šum predstavlja stroži test robusnosti algoritama na impulsni i neregularni šum, te direktno ispituje očuvanje tačnosti estimacije u prisustvu većih odstupanja.
- **Konstantni bias upravljačkog ugla:** trajni sistematski pomak u upravljačkom ulazu za orijentaciju robota. U simulaciji je realizovan dodavanjem konstantnog poremećaja $\Delta\gamma = 0.05$ na komandni ugao G, što uzrokuje da se robot u svakom koraku blago rotira ulijevo u odnosu na stvarno kretanje. Za razliku od nasumičnih šumova, čija je srednja vrijednost nula, ova greška se ne poništava, već se akumulira tokom vremena.
- **Povremeni balistički drift:** procesni poremećaj pri kojem robot povremeno prelazi u režim konstantnog proklizavanja. Kada se aktivira, u slučajno odabra-

nom smjeru djeluje konstantno ubrzanje. Za razliku od Gaussovog šuma, čija je srednja vrijednost nula, ovaj poremećaj se ne poništava tokom vremena, već se sistematski akumulira i raste kvadratno s vremenom. On je nestacionaran, pokazuje memoriju i dovodi do odstupanja od pretpostavljenog kinematičkog modela (vidjeti Algoritam 13).

U ovom scenariju razmatrana je mala mapa, jer omogućava izvođenje brzih i preglednih eksperimenata u kojima se efekti velikih odstupanja (*outlier*-a), nelinearnih poremećaja i proklizavanja jasno uočavaju. Simulacije su sprovedene bez ponovnog prolaska kroz listu putokaza radi bolje kontrole i lakše interpretacije rezultata. Za sve particle filter pristupe korišćeno je N=500 čestica, što je dovoljno za pouzdanu estimaciju stanja u ovim uslovima.

5.3 Rezultati simulacija

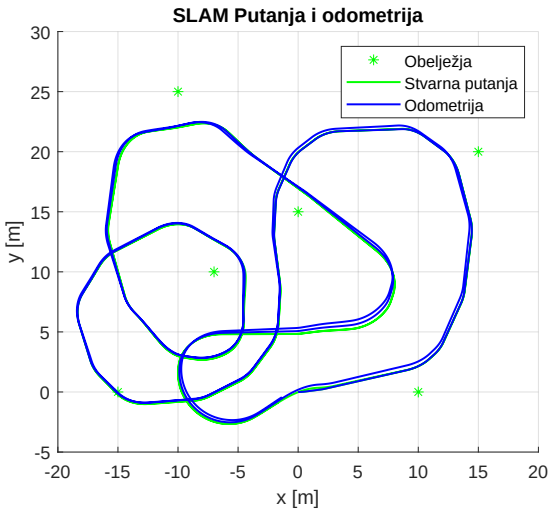
5.3.1 Osnovni slučaj

Na slici 5.4a prikazana je procijenjena trajektorija robota zasnovana isključivo na modelu kretanja (odometrija), u poređenju sa stvarnom putanjom. Može se uočiti da se greška odometrije akumulira tokom kretanja, što dovodi do sve većeg odstupanja od stvarne trajektorije. Treba napomenuti da se zbog prirode Gaussovog šuma, čija je srednja vrijednost jednaka nuli, povremeno dolazi do lokalnog smanjenja greške procjene. Ipak, golobalno gledano greška odometrije se sistematski akumulira tokom vremena. Ovo je očekivano ponašanje, jer odometrija zavisi isključivo od mjerenja senzora kretanja i ne koristi korekciju [10]. Ovaj rezultat potvrđuje osnovnu motivaciju za primjenu SLAM algoritama, koji omogućavaju značajno smanjenje ovakvih grešaka.

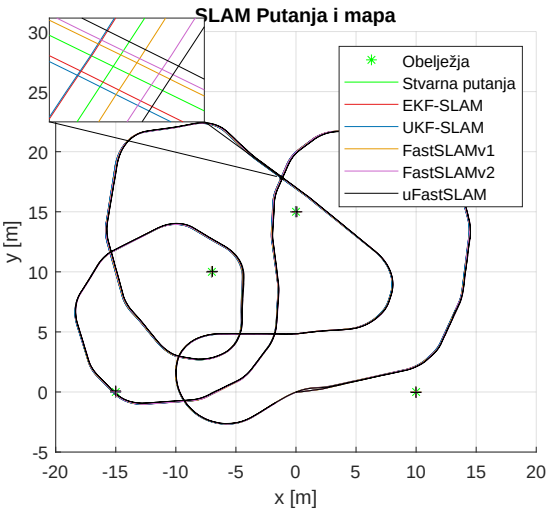
Na slici 5.4b prikazane su estimirane putanje dobijene primjenom razmatranih algoritama, kao i stvarna trajektorija robota. Uočava se da razlike među algoritmima na ovoj mapi nisu izražene, što je posljedica relativno malih kovarijansi Gaussovih šumova. Na slici 5.4c prikazane su RMSE vrijednosti, koje pokazuju da Kalmanovi filteri postižu nešto preciznije rezultate od metoda zasnovanih na particle filteru. Ovakvi rezultati su očekivani, jer model robota nije izrazito nelinearan, a šum je Gausov, pa Kalmanov filter u ovom slučaju daje vrlo dobre rezultate.

NEES vrijednosti za EKF i UKF su prikazane na slici 5.4d, odakle si vidi da EKF-SLAM i UKF-SLAM ostaju unutar 95% intervala pouzdanosti,[5] što znači da procijenjena kovarijansa dobro odražava stvarnu grešku [60].

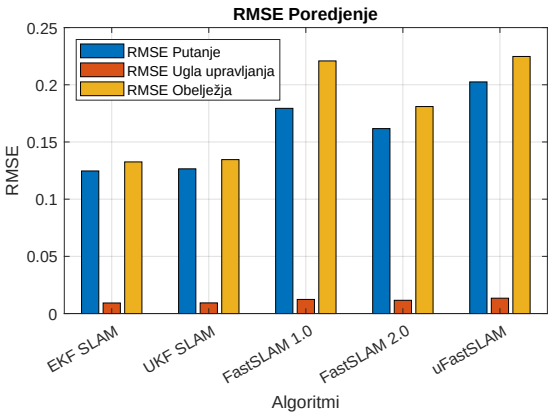
[5]Prag od 95% u NEES analizi predstavlja statističko očekivanje da procijenjena kovarijansa obuhvata stvarnu grešku u većini slučajeva. Odstupanja izvan ovog intervala ne znače da je filter neispravan, već ukazuju na to da procijenjena kovarijansa ne odražava stvarnu grešku.



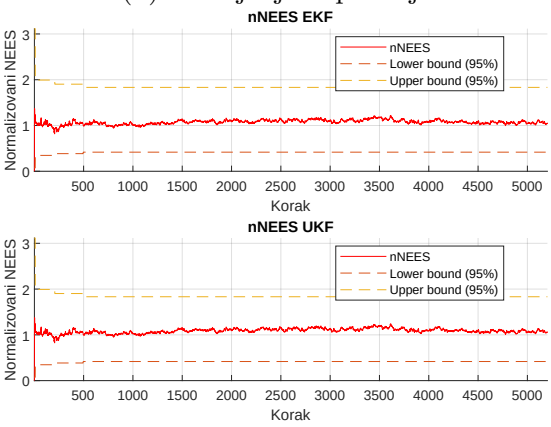
(a) Odometrija male mape



(b) Procijenjene putanje



(c) RMSE poređenje



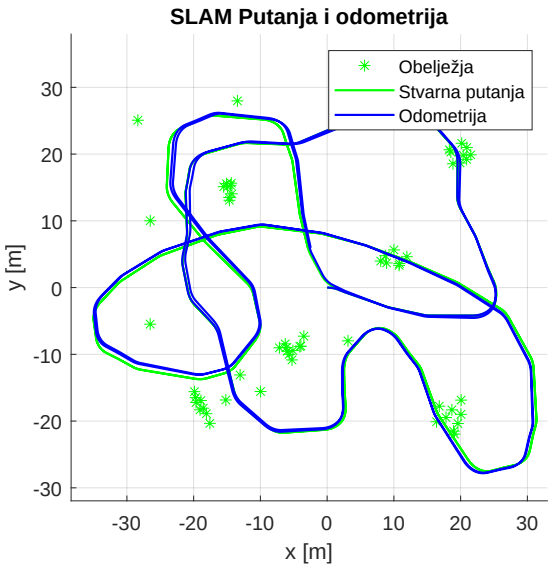
(d) NEES za Kalmanove filtere

Slika 5.4: Rezultati simulacija za malu mapu (Osnovni slučaj): (a) odometrija, (b) procijenjene putanje, (c) RMSE metrika, (d) NEES za Kalmanove filtere.

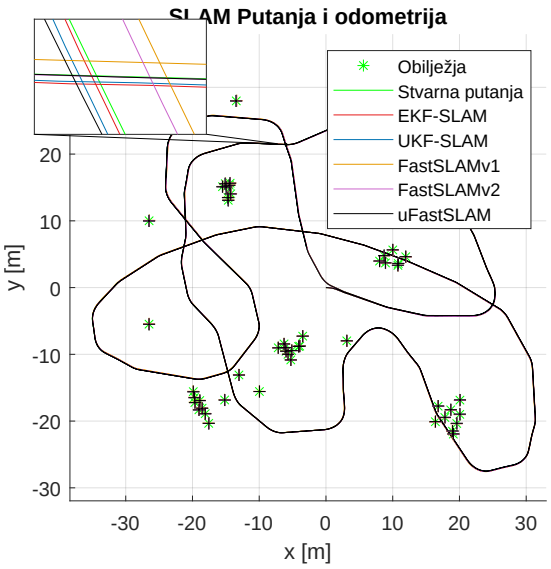
U slučaju *cluster* mape, odomotrija robota, estimirane trajektorije, RMSE i NEES vrijednosti prikazane su na slikama 5.5a–5.5d respektivno. Može se uočiti da su ovom slučaju razlike u performansama algoritama manje izražene. Naime, EKF-SLAM i UKF-SLAM i dalje postižu najmanje vrijednosti RMSE, dok FastSLAM 1.0 i 2.0 postižu uporedive performanse. Ovo se može objasniti time što je particle filtrima u mapama sa većim brojem obilježja dostupno više mjernih podataka, što značajno povećava tačnost procjene. Sa druge strane, uFastSLAM ima nešto slabiju tačnost zbog malog broja čestica.

NEES vrijednosti za EKF i UKF prikazane su na slici 5.4d, odakle se vidi da se procjene oba filtera kreću unutar granica konzistentnosti tokom cijelog trajanja eksperimenta, što ukazuje da procijenjene kovarijanse odgovaraju stvarnim greškama i da oba filtera daju statistički konzistentne procjene stanja u posmatranom scenariju.

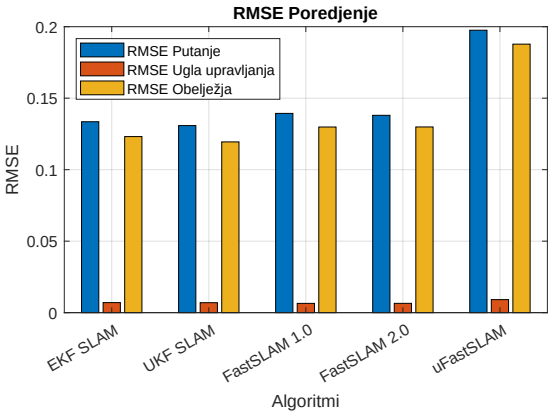
Rezultati simulacija za veliku mapu prikazani su na slikama 5.6a–5.6d. Može se uočiti da razlike među algoritmima ovdje postaju izraženije. EKF-SLAM i UKF-



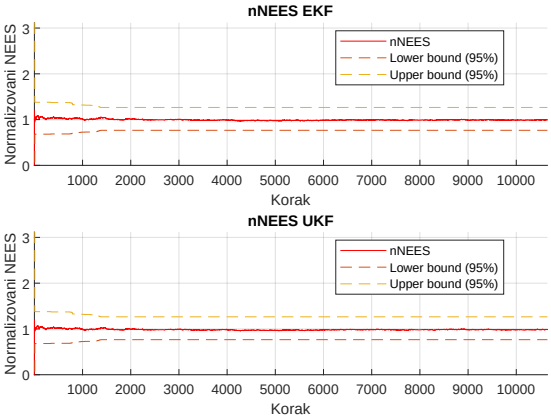
(a) Odometrija cluster mape



(b) Procijenjene putanje



(c) RMSE poređenje



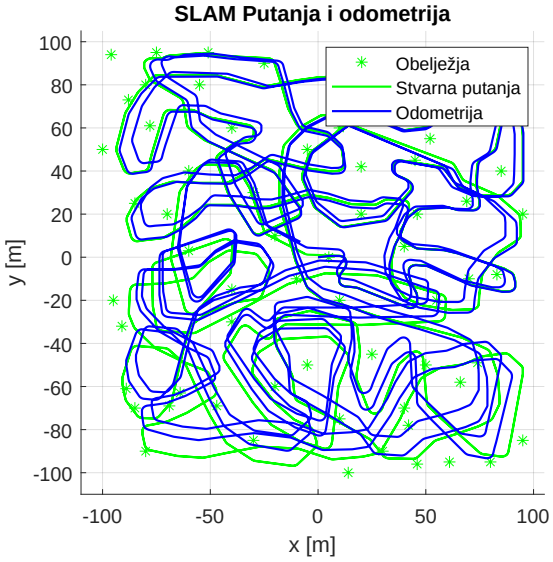
(d) NEES za Kalmanove filtere

Slika 5.5: Rezultati simulacija za *cluster* mapu (Osnovni slučaj): (a) odometrija, (b) procijenjene putanje, (c) RMSE metrika, (d) NEES za Kalmanove filtere.

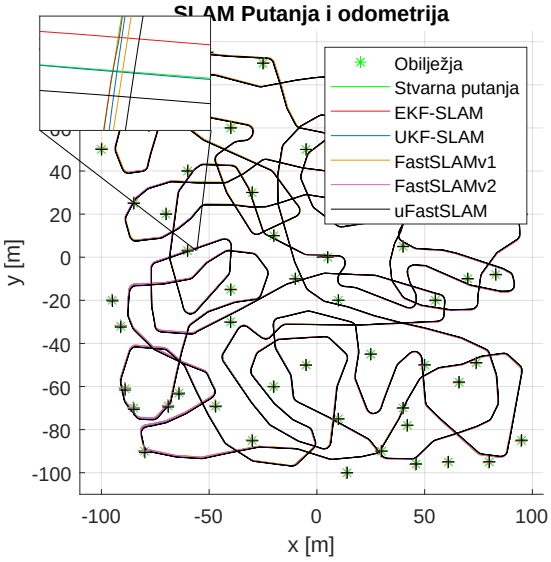
SLAM i dalje ostvaruju najmanji RMSE (slika 5.6c), dok FastSLAM 1.0 i 2.0 imaju prihvatljivu tačnost zahvaljujući velikom broju čestica. Nasuprot tome, uFastSLAM sa samo N=10 čestica pokazuje slabiju tačnost u složenijim scenarijima, iako je taj broj bio dovoljan za manje i jednostavnije mape. Ovo potvrđuje da, bez obzira na varijantu particle filter SLAM-a, broj čestica ima presudan uticaj na tačnost procjene.

5.3.2 Uticaj kovarijanse šuma

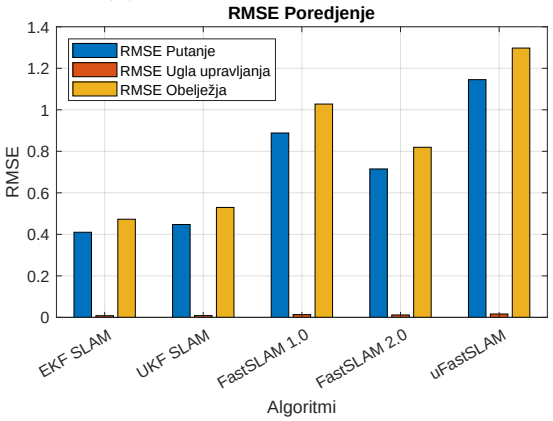
Rezultati za ovaj scenario prikazani su u tabelama 5.6 i 5.7. Tabele prikazuju prosječne vrijednosti RMSE za estimaciju putanje robota i položaja obilježja, redom. Rezultati su po redovima raspoređeni prema različitim kombinacijama procesnog (Q) i mjernog (R) šuma, dok kolone odgovaraju performansama pojedinačnih algoritama.



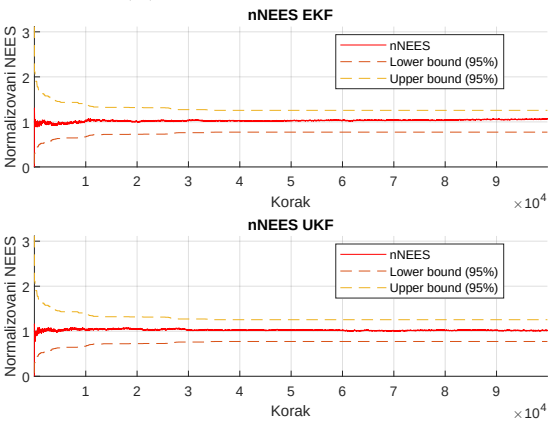
(a) Odometrija velike mape



(b) Procijenjene putanje



(c) RMSE poređenje



(d) NEES za Kalmanove filtere

Slika 5.6: Rezultati simulacija za veliku mapu (Osnovni slučaj): (a) odometrija, (b) procijenjene putanje, (c) RMSE metrika, (d) NEES za Kalmanove filtere.

Scenario	**EKF**	**UKF**	**FS 1.0**	**FS 2.0**	**uFast**
malo Q, malo R	0.1335	**0.1308**	0.1394	0.1380	0.1975
srednje Q, srednje R	**0.2094**	0.2103	0.3129	0.2322	0.3649
veliko Q, malo R	0.2888	0.2906	0.3662	**0.2695**	0.3762
malo Q, veliko R	**0.2252**	0.2259	0.3044	0.3174	0.3694
veliko Q, veliko R	**0.4451**	0.4568	0.6455	0.5146	0.6603

Tabela 5.6: Prosječne vrijednosti RMSE putanje na *cluster* mapi za različite kombinacije procesnog (Q) i mjernog (R) šuma.

U slučaju malih kovarijansi procesnog i mjernog šuma (Q malo, R malo) svi algoritmi rade u gotovo idealnim uslovima: model kretanja je tačan, a mjerenja precizna, što omogućava pouzdanu procjenu stanja. EKF-SLAM i UKF-SLAM u ovim uslovima ostvaruju najmanje i gotovo iste vrijednosti grešaka, jer model robota nije izraženo nelineran, pa prednosti korišćenja sigma tačaka kod UKF-a ne dolaze do iz-

Scenario	**EKF**	**UKF**	**FS 1.0**	**FS 2.0**	**uFast**
malo Q, malo R	0.1231	**0.1194**	0.1298	0.1299	0.1878
srednje Q, srednje R	0.1790	**0.1784**	0.2924	0.2192	0.3444
veliko Q, malo R	0.2597	0.2611	0.3747	**0.2568**	0.3520
malo Q, veliko R	0.1962	**0.1954**	0.2892	0.2905	0.3455
veliko Q, veliko R	**0.3836**	0.3852	0.5988	0.4768	0.6098

Tabela 5.7: Prosječne vrijednosti RMSE obilježja na *cluster* mapi za različite kombinacije procesnog (Q) i mjernog (R) šuma.

ražaja. Metode zasnovane na česticama postižu nešto manju tačnost, a uFastSLAM, zbog malog broja čestica (N=10), ima najmanju tačnost, iako procjena ostaje u prihvatljivim granicama.

Porastom vrijednosti šuma (Q srednje, R srednje) greške očekivano rastu, ali međusobni odnos performansi algoritama ostaje približno isti: EKF-SLAM i UKF-SLAM i dalje ostvaruju veoma slične rezultate; FastSLAM 2.0 ima manji rast greške u odnosu na FastSLAM 1.0 zahvaljujući poboljšanom prijedlogu distribucije čestica; uFastSLAM, zbog malog broja čestica, ostvaruje nižu tačnost, ali procjena ostaje stabilna.

Na gornjoj granici (Q veliko, R veliko) svi algoritmi pokazuju najizraženiju degradaciju. Predikcija je nepouzdana, a korekcija zbog velikog mjernog šuma ima veoma ograničen uticaj, pa RMSE u ovom scenariju ima najveće vrijednosti. EKF-SLAM i UKF-SLAM imaju medjusobono uporedive performanse, dok FastSLAM 1.0 i FastSLAM 2.0 takođe pokazuju povećanje greške. Kod FastSLAM-a 1.0 izražena je degeneracija težinskih koeficijenata, dok FastSLAM 2.0 gubi prednost poboljšanog predloga jer zašumljena mjerenja ne mogu pouzdano usmjeriti uzorkovanje. uFastSLAM, i pored malog broja čestica, postiže rezultate bliske FastSLAM-u 1.0, što pokazuje da se prihvatljiva tačnost može postići uz znatno manje računske resurse.

Dobijeni rezultati potvrđuju očekivanu zavisnost tačnosti od kvaliteta modela i mjerenja: povećanje Q dovodi do većih grešaka u predikciji, dok povećanje R umanjuje uticaj mjerenja na korekciju procjene. Najmanje greške ostvaruju EKF-SLAM i UKF-SLAM, što je očekivano, jer ove metode polaze od pretpostavke Gaussovog procesnog i mjernog šuma, koja u potpunosti odgovara uslovima simulacije. FastSLAM varijante postižu slabiju tačnost, jer se oslanjaju na uzorkovanje čestica. Iako je ovakav rezultat očekivan u tipičnim Gaussovim uslovima, u određenim konfiguracijama šuma algoritmi pokazuju različite obrasce ponašanja. Zbog toga se u nastavku izdvajaju dvije granične konfiguracije: slučajevi u kojima je procesni šum veliki, a mjerni mali, te obrnuto, koje jasno ilustruju ove razlike.

U scenariju velikog procesnog i malog mjernog šuma (Q veliko, R malo) Fast-

SLAM 2.0 ostvaruje najmanje greške. Razlog je u poboljšanom predlogu uzorkovanja, koji uključuje mjerenja, čime se efekti velikog procesnog šuma u predikciji ublažavaju. Za razliku od particle metoda, kod EKF-SLAM i UKF-SLAM sve komponente stanja dijele zajedničku kovarijansu, pa se greške u predikciji i mjerenju propagiraju kroz čitavo stanje. Zbog toga ovi pristupi imaju ograničene mogućnosti da iskoriste pouzdana mjerenja kada je Q veliko. Suprotno tome, u slučaju malog procesnog i velikog mjernog šuma (Q malo, R veliko), mjerenja nisu dovoljno pouzdana da bi unaprijeđeni prijedlog uzorkovanja znatno uticao na tačnost, pa su FastSLAM 1.0 i FastSLAM 2.0 međusobno vrlo slični.

5.3.3 Uticaj dometa senzora

Rezultati za ovaj scenario prikazani su u tabelama 5.8 i 5.9, gdje su date prosječne vrijednosti RMSE za putanju i obilježja. Redovi odgovaraju rezultatima za različite domete senzora, dok su u kolonama dati rezultati pojedinačnih algoritama.

Scenario		**EKF**	**UKF**	**FS 1.0**	**FS 2.0**	**uFast**
Mali domet	($R_{\max} = 20$)	**0.9529**	0.9848	1.7012	1.3487	2.1363
Srednji domet	($R_{\max} = 35$)	**0.5071**	0.5079	0.8175	0.5377	0.6617
Veliki domet	($R_{\max} = 50$)	**0.3221**	0.3241	0.5184	0.3895	0.5033

Tabela 5.8: Prosječne vrijednosti RMSE putanje na velikoj mapi za različite domete senzora $R_{\max}$.

Scenario		**EKF**	**UKF**	**FS 1.0**	**FS 2.0**	**uFast**
Mali domet	($R_{\max} = 20$)	**0.9187**	0.9566	1.8500	1.5446	2.3562
Srednji domet	($R_{\max} = 35$)	**0.5958**	0.6037	0.9535	0.6195	0.7718
Veliki domet	($R_{\max} = 50$)	**0.3602**	0.3696	0.6018	0.4552	0.5864

Tabela 5.9: Prosječne vrijednosti RMSE obilježja na velikoj mapi za različite domete senzora $R_{\max}$.

U slučaju malog dometa senzora broj obilježja koja ulaze u vidno polje robota je ograničen, pa filteri imaju manje informacija za korekciju predikcije. Zbog toga su greške veće u odnosu na scenarije sa većim dometom. Najmanje greške ostvaruju EKF-SLAM i UKF-SLAM, što potvrđuje njihovu prednost kada su ispunjene pretpostavke o Gausovoj raspodjeli šuma. Particle filter varijante pokazuju znatno veće vrijednosti RMSE, pri čemu uFastSLAM ostvaruje najslabije rezultate, jer mali broj čestica nije dovoljan da nadomjesti manjak senzorskih podataka.

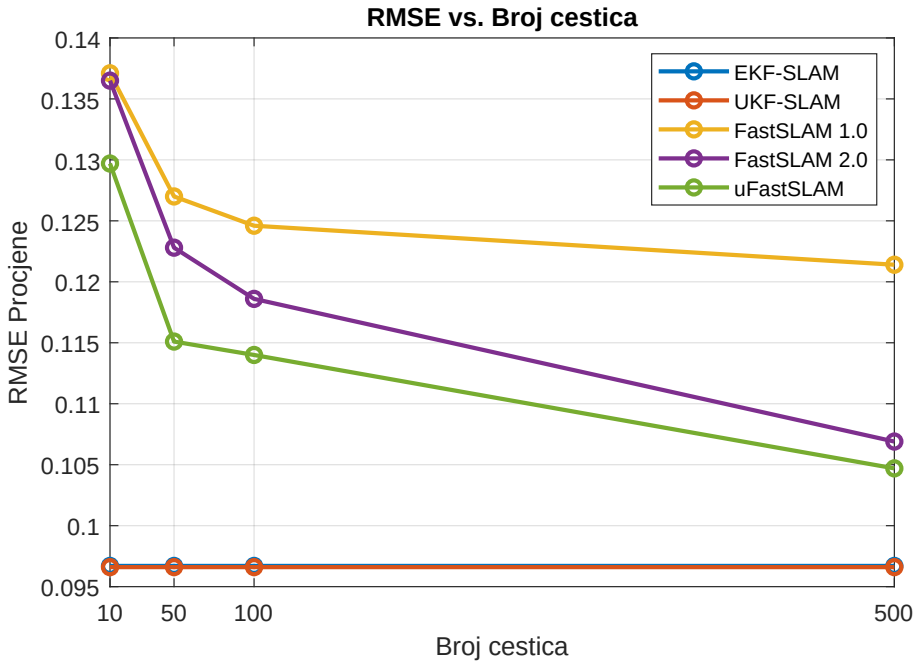
Povećanjem dometa senzora broj raspoloživih obilježja u svakom mjernom intervalu se značajno uvećava, što rezultuje smanjenjem RMSE vrijednosti svih algoritama u odnosu na prethodni slučaj. EKF-SLAM i UKF-SLAM i dalje ostvaruju

najmanje greške i međusobno slične rezultate, dok FastSLAM 2.0 zahvaljujući poboljšanom predlogu distribucije čestica (koji uz odometriju uključuje i mjerenja), uspijeva da se približi njihovoj tačnosti. FastSLAM 1.0 pokazuje najslabije rezultate, dok uFastSLAM postiže tačniju procjenu u odnosu na njega, iako koristi znatno manje čestica.

Pri najvećem dometu senzora robot u svakom koraku opaža veliki broj obilježja, u prosjeku i do četvrtine cijele mape, što omogućava značajne korekcije predikcije i dovodi do najmanjih vrijednosti RMSE-a. EKF-SLAM i UKF-SLAM zadržavaju najbolju tačnost, dok se FastSLAM 2.0 približava njihovim rezultatima zahvaljujući efektivnom korišćenju velikog broja mjerenja. FastSLAM 1.0 pokazuje nižu tačnost, dok u ovom scenariju uFastSLAM ostvaruje bolje rezultate od njega, i to sa samo $N = 10$ čestica. Ovo pokazuje da veliki broj mjerenja omogućava algoritmima poput FastSLAM-a 2.0 i uFastSLAM-a da bolje iskoriste dostupne podatke i time nadmaše FastSLAM 1.0, dok Kalmanovi filteri ostaju najtačniji kada je pretpostavka o Gausovoj raspodjeli šumova ispunjena.

5.3.4 Uticaj broja čestica

Rezultati za ovaj scenario prikazani su u tabeli 5.10 i na slici 5.7, gdje su date prosječne vrijednosti RMSE-a za putanju i obilježja u zavisnosti od broja čestica. U tabeli redovi prestavljaju različite algoritme, dok kolone odgovaraju izabranim vrijednostima broja čestica. Unutar svake ćelije date su vrijednosti RMSE-a za putanju i obilježja u formatu: RMSE putanje / RMSE obilježja.



Slika 5.7: Trend smanjenja greške procjene putanje sa porastom broja čestica kod particle filter metoda.

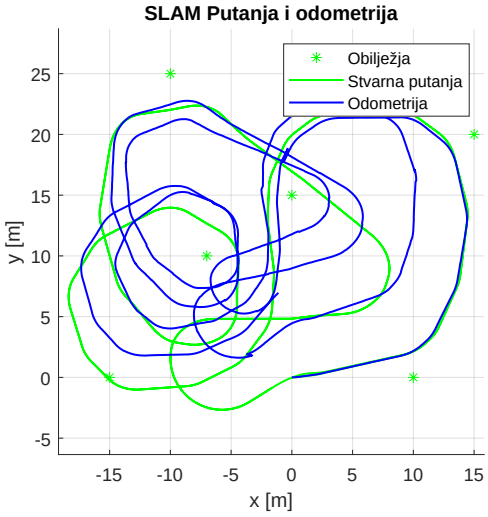
Broj Čestica	**10**	**50**	**100**	**500**
FastSLAM 1.0	0.1371 / 0.1512	0.1270 / 0.1572	0.1246 / 0.1586	0.1214 / 0.1605
FastSLAM 2.0	0.1365 / 0.1443	0.1228 / 0.1402	0.1186 / 0.1398	0.1069 / 0.1336
uFastSLAM	0.1297 / 0.1386	0.1151 / 0.1296	0.1140 / 0.1236	0.1047 / 0.1158

Tabela 5.10: Prosječne vrijednosti RMSE putanje i obilježja na maloj mapi za različite vrijednosti broja čestica.

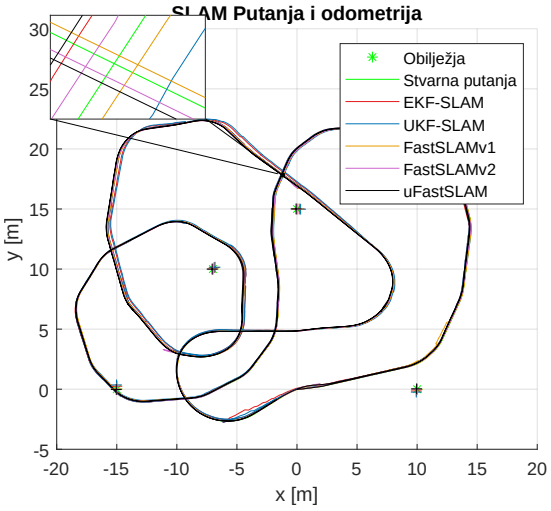
Kako je karakteristično za metode zasnovane na uzorkovanju, greška procjene opada približno po zakonu $1/\sqrt{N}$, gdje N označava broj čestica. Da bi se provjerilo da li greška zaista opada približno proporcionalno tom zakonu, urađena je linearna regresija u odnosu na promjenljivu $1/\sqrt{N}$. Dobijeni koeficijenti determinacije pokazuju dobro slaganje sa očekivanim trendom: FastSLAM 1.0 prati teorijsko smanjenje greške gotovo u potpunosti ($R^2_{\text{det}} \approx 1$), dok FastSLAM 2.0 ($R^2_{\text{det}} \approx 0.94$) i uFastSLAM ($R^2_{\text{det}} \approx 0.97$) imaju nešto veća odstupanja zbog slučajne prirode uzorkovanja. Ovi rezultati potvrđuju da povećanje broja čestica smanjuje RMSE, ali i da se relativno poboljšanje smanjuje kako N raste, što je u skladu sa teorijskim očekivanjima Monte Carlo metoda [63].

Zanimljivo je primijetiti da kod FastSLAM-a 1.0 greška u procjeni obilježja ne pokazuje očekivano smanjenje sa porastom broja čestica. Ovaj efekat proizlazi iz toga što algoritam uzorkuje nove čestice isključivo na osnovu odometrije, dok se mjerenja koriste samo pri ažuriranju težina. Posljedica ovoga je degeneracija težinskih koeficijenata i smanjenje efektivne raznolikosti čestica, što otežava preciznu procjenu obilježja. Drugim riječima, iako veći broj čestica teoretski smanjuje varijansu procjene, u praksi FastSLAM 1.0 ne uspijeva da ih efikasno iskoristi, pa dalji rast broja čestica ne vodi uvjek smanjenju greške i može imati čak i rastući trend [58, 64].

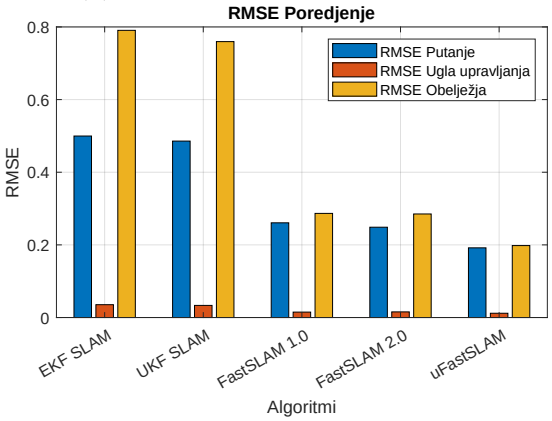
uFastSLAM, pri istom broju čestica, postiže bolje performanse od klasičnih FastSLAM varijanti, što potvrđuje njegovu efikasnost i opravdava ciljanu upotrebu manjeg broja čestica. Ipak, i pored smanjenja greške sa porastom broja čestica, čestične metode ne dostižu nivo tačnosti EKF-SLAM-a i UKF-SLAM-a u ovim uslovima. Razlog za to leži u činjenici da je kretanje robota modelovano uz pretpostavku Gausovog procesnog i mjernog šuma, što u potpunosti odgovara statističkim pretpostavkama na kojima se Kalmanovi pristupi zasnivaju. U takvim uslovima EKF-SLAM i UKF-SLAM generišu optimalne procjene, dok particle filter metode, uprkos većem broju čestica i dalje nose dodatnu (Monte Carlo) varijansu.



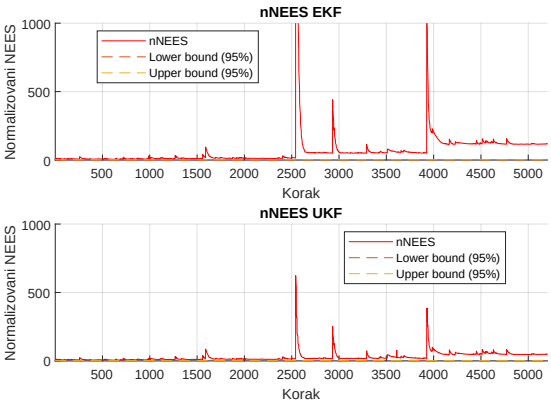
(a) Odometrija male mape



(b) Procijenjene putanje



(c) RMSE poređenje



(d) NEES za Kalmanove filtere

Slika 5.8: Rezultati simulacija za malu mapu (Student-t raspodjela šuma): (a) odometrija, (b) procijenjene putanje, (c) RMSE metrika, (d) NEES za Kalmanove filtere.

5.3.5 Test robusnosti

Student-t raspodjela šuma

Rezultati za scenario sa Student-t raspodjelom šuma prikazani su na slici 5.8. Slika 5.8a prikazuje odometrijsku procjenu putanje; slika 5.8b prikazuje procijenjene putanje za različite algoritme; slika 5.8c poredi vrijednosti RMSE, dok slika 5.8d prikazuje normalizovani NEES za UKF i EKF.

Rezultati prikazani na slici 5.8 ukazuju da šumovi koji podliježu Student-t raspodjeli dovode do značajnog odstupanja odometrije, dok procijenjene putanje svih metoda pokazuju izraženija odstupanja, pri čemu u određenim trenucima dolazi do povećanja greške u estimaciji pozicije (slika 5.8b). U poređenju sa čestičnim pristupima, EKF-SLAM i UKF-SLAM pokazuju veću grešku procjene. To je očekivano, jer Kalmanovi filteri pretpostavljaju Gausovu raspodjelu šuma i nisu u mogućnosti pravilno da obrade velika odstupanja u mjerenjima koja su karakteristična za Student-t

raspodjelu. Iako im procjena nije dramatično narušena, NEES (slika 5.8d) vrijednosti jasno pokazuju da procijenjena kovarijansa ne odražava stvarni nivo greške: filteri postaju nekonzistentni jer nastavljaju da modeluju statistiku šuma kao da potiče iz Gausove raspodjele.

Čestične metode, s druge strane, pokazuju znatno veću robusnost na teške repove i izolovane impulsne greške, jer uzorkovanje omogućava zadržavanje više hipoteza o stanju robota. Kao rezultat, čestične metode, a posebno uFastSLAM, postižu manji RMSE i bolje procjene putanje i mape od EKF-SLAM-a i UKF-SLAM-a. Među čestičnim metodama, uFastSLAM se najjasnije izdvaja po kvalitetu procjene. Zahvaljujući augmentovanom Unscented Transform pristupu, algoritam znatno bolje obrađuje nelinearnosti i ekstremne šumove koji karakterisu Student-t raspodjelu.

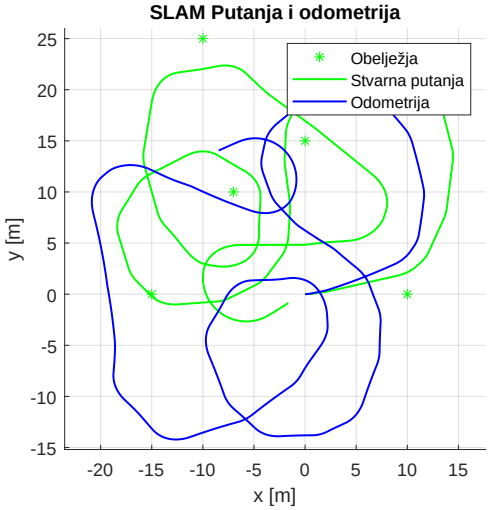
Sveukupno, kada je šum opisan Student-t raspodjelom, dolazi do vidljivih razlika u ponašanju razmatranih metoda. Čestične metode daju tačniju procjenu putanje i obilježja, jer bolje obrađuju rijetka, ali velika odstupanja u mjerenjima koja su posljedica šumova sa teškim repovima. Sa druge strane, EKF SLAM i UKF SLAM pokazuju veća odstupanja i manju konzistentnost, pošto se oslanjaju na pretpostavku Gausove raspodjele šuma, koja nije u potpunosti prilagođena ovakvim uslovima.

Uticaj konstantnog bias-a u upravljačkom uglu

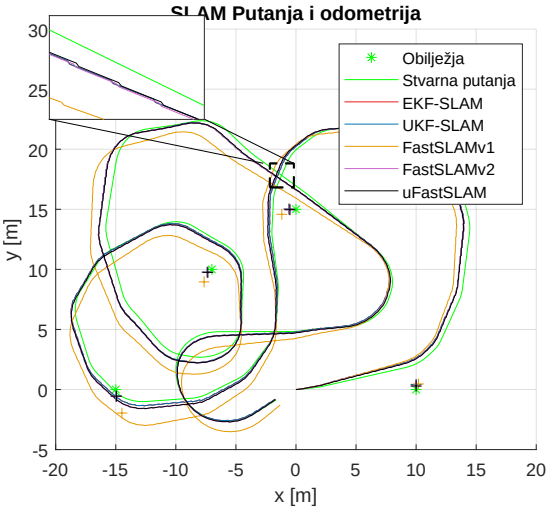
Rezultati za scenario sa konstantnim biasom upravljačkog ugla prikazani su na slici 5.9. Slika 5.9a prikazuje odometrijsku procjenu putanje; slika 5.9b prikazuje procijenjene putanje za različite algoritme; slika 5.9c poredi vrijednosti RMSE, dok slika 5.9d prikazuje normalizovani NEES za UKF i EKF.

Kao što se vidi na slici 5.9b, EKF-SLAM i UKF-SLAM postižu manju grešku procjene putanje u poređenju sa FastSLAM-om 2.0 i uFastSLAM-om. Suprotno tome, u procjeni obilježja FastSLAM 2.0 i uFastSLAM ostvaruju tačnije rezultate, dok FastSLAM 1.0 pokazuje uočljivo slabije rezultate. Razlog ovih rezultata leži u različitim načinima na koje ove metode tretiraju stanje robota i mape. Kalmanovi pristupi modeluju poziciju robota i obilježja kao jedan zajednički vektor stanja, zbog čega se odometrijske greške ne odražavaju samo na procjenu putanje, već se prenose i na mapu, pa se sistematski pomak (bias) odražava i na procijenu obilježja. Kod čestičnih metoda bias upravljačkog ugla utiče na uzorkovanje čestica i pogoršava procjenu putanje, ali se obilježja procjenjuju nezavisnim EKF/UKF-ovima unutar svake čestice, pa njihove procjene ostaju bliže stvarnim vrijednostima. Zbog toga FastSLAM 2.0 i uFastSLAM, uprkos većem odstupanju putanje, postižu precizniju procjenu mape u odnosu na EKF-SLAM i UKF-SLAM.

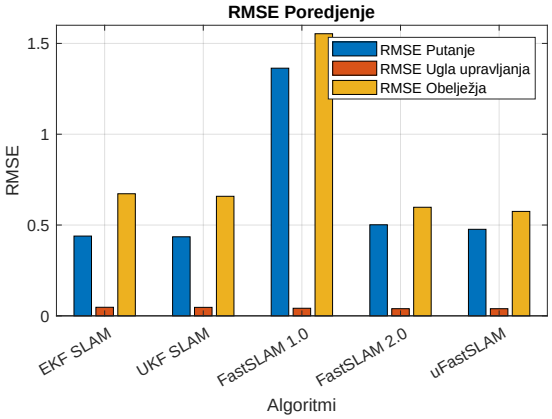
Rezultati ukazuju na kontrast između procjene putanje i mape: EKF-SLAM i UKF-SLAM održavaju bolju procjenu putanje (slika 5.9b), dok FastSLAM 2.0 i



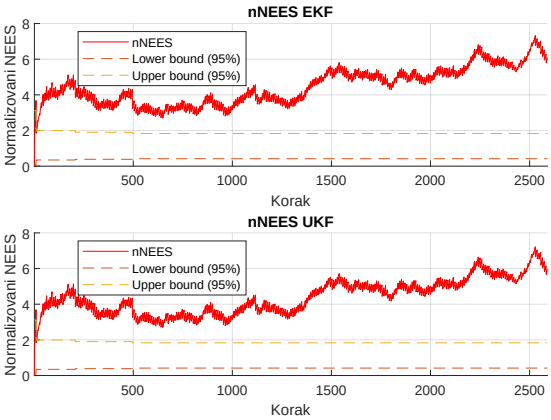
(a) Odometrija male mape



(b) Procijenjene putanje



(c) RMSE poređenje



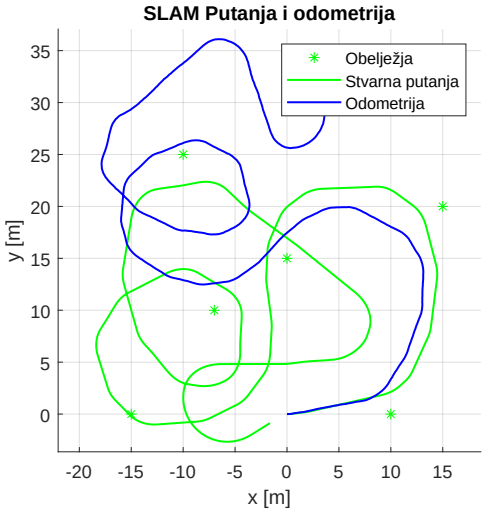
(d) NEES za Kalmanove filtere

Slika 5.9: Rezultati simulacija za malu mapu (bias upravljačkog ugla): (a) odometrija, (b) procijenjene putanje, (c) RMSE metrika, (d) NEES za Kalmanove filtere.

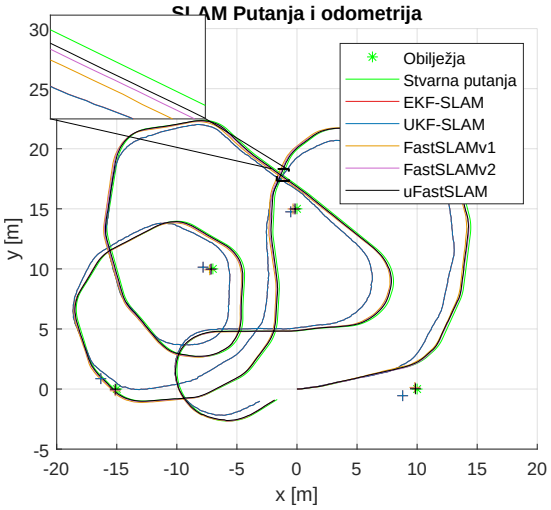
uFastSLAM postižu preciznije mapiranje obilježja. FastSLAM 1.0 pokazuje najslabije performanse u oba aspekta (slika 5.9c), jer koristi predikciju zasnovanu isključivo na modelu kretanja, što dovodi do izražene degeneracije čestica u prisustvu sistematskog bias-a.

NEES vrijednosti za EKF i UKF prikazane su na slici 5.9d. Vrijednosti pretežno ostaju iznad očekivanog intervala pouzdanosti, pri čemu prosječna vrijednost postepeno raste, što ukazuje na smanjenu konzistentnost EKF-SLAM-a i UKF-SLAM-a – procijenjena kovarijansa potcjenjuje stvarnu grešku. Kako se bias u upravljačkom uglu akumulira tokom vremena, očekivano je da se prosječna NEES vrijednost u dužim simulacijama dodatno udaljava od nominalne granice. Za čestične metode NEES u ovom obliku nije direktno primjenjiv, jer ne postoji jedinstvena kovarijansa stanja; ali vizuelni rezultati (slika 5.9c), potvrđuju ranije uočeni kompromis: slabija procjena putanje, ali tačnije pozicije obilježja.

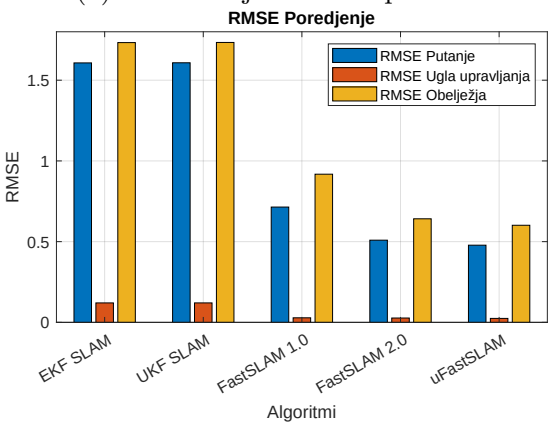
Uticaj balističkog drift-a



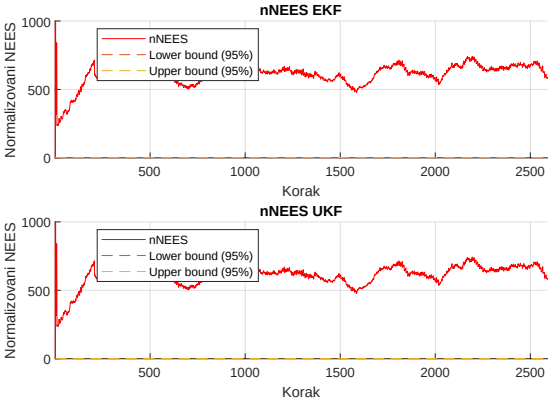
(a) Odometrija male mape



(b) Procijenjene putanje



(c) RMSE poređenje



(d) NEES za Kalmanove filtere

Slika 5.10: Rezultati simulacija za malu mapu (balistički drift): (a) odometrija, (b) procijenjene putanje, (c) RMSE metrika, (d) NEES za Kalmanove filtere.

Rezultati za scenario sa balističkim drift-om prikazani su na slici 5.10. Slika 5.10a prikazuje odometrijsku procjenu putanje; slika 5.10b prikazuje procijenjene putanje za različite algoritme; slika 5.10c poredi vrijednosti RMSE, dok slika 5.10d prikazuje prikazuje normalizovani NEES za UKF i EKF.

Balistički drift uzrokuje postepeno razilaženje odometrijske i stvarne putanje robota (slika 5.10a). Na slici 5.10b uočava se da EKF-SLAM i UKF-SLAM daju gotovo identične rezultate, što ukazuje da oba filtera na sličan način reaguju na ovaj tip poremećaja. Procijenjene putanje vidno odstupaju od stvarne zbog akumulacije sistematske greške u modelu kretanja, jer pretpostavke koje EKF i UKF nameću nad nelinearnom dinamikom i šumovima ograničavaju njihovu sposobnost da koriguju ovaj poremećaj.

Za razliku od Kalmanovih metoda, čestične metode se pokazuju robusnije na ovaj tip poremećaja, što se vidi na slici 5.10b. Rezultat su manje RMSE vrijednosti

(slika 5.10c): FastSLAM 2.0 i uFastSLAM postižu najtačnije procjene, dok FastSLAM 1.0 ostvaruje bolje rezultate od Kalmanovih pristupa, ali slabije od ostalih čestičnih metoda. EKF-SLAM i UKF-SLAM imaju najveće greške jer pretpostavka o Gausovoj raspodjeli šuma i lokalnim linearnim aproksimacijama ne obuhvataju sistematski drift, pa se greška postepeno akumulira kroz vrijeme.

NEES vrijednosti za EKF i UKF su prikazane na slici 5.10d), odakle se vidi da vrijednosti za EKF-SLAM i UKF-SLAM vrlo brzo izlaze iz očekivanih granica i zadržavaju rastući trend, što ukazuje na izraženu nekonzistentnost filtera – procijenjena kovarijansa potcjenjuje stvarnu grešku. Ova pojava je naročito izražena kod balističkog drift-a, jer se greška akumulira kvadratno u vremenu, dok je model šuma i dalje zasnovan na Gaussovoj pretpostavci. Kao što je ranije napomenuto, za čestične metode NEES se u ovom obliku ne može direktno izračunati, ali vizuelni rezultati (slika 5.10c) potvrđuju da one pružaju robusnije i preciznije procjene mape u poređenju sa Kalmanovim filterima.

Glava 6

Zaključak

U ovom radu sprovedena je sveobuhvatna analiza dva pristupa problemu simultane lokalizacije i mapiranja (SLAM), zasnovana na eksperimentalnim rezultatima i kvantitativnom poređenju performansi. Razmatrane su sledeće klase metoda:

- Kalmanovi filteri (EKF-SLAM i UKF-SLAM) – parametarski, deterministički pristupi zasnovani na Gaussovim pretpostavkama i linearizovanim aproksimacijama;
- čestične metode (FastSLAM 1.0, FastSLAM 2.0 i uFastSLAM) – neparametarski, nelinearni i stohastički algoritmi zasnovani na Monte Carlo uzorkovanju.

Analiza je obuhvatila okruženja različite složenosti i scenarije sa kontrolisanim promjenama kovarijansi procesnog i mjernog šuma, dometa senzora, broja čestica, kao i testove robusnosti u prisustvu sistematskih poremećaja (Student-t šum, bias upravljačkog ugla i balistički drift).

Na osnovu dobijenih rezultata mogu se izvesti sljedeći zaključci. EKF-SLAM i UKF-SLAM pokazuju najveću tačnost i konzistentnost kada su šumovi bliski Gausovoj raspodjeli, a pretpostavke modela kretanja dobro odražavaju stvarno kretanje robota. U uslovima povećane nesigurnosti u modelu kretanja, ali uz pouzdana mjerenja, bolje rezultate postižu čestične metode koje u procesu uzorkovanja koriste informacije iz mjerenja (FastSLAM 2.0 i uFastSLAM). Povećanjem dometa senzora dolazi do smanjenja greške procjene kod svih algoritama, budući da veći broj obilježja ulazi u mjerni opseg. Ovi rezultati pokazuju da unapređenje senzorske opreme direktno doprinosi povećanju tačnosti SLAM sistema, pa se u praktičnim primjenama senzori većeg dometa i preciznosti smatraju poželjnim izborom. Kod čestičnih metoda, povećanje broja čestica generalno dovodi do smanjenja greške procjene, ali efekat tog poboljšanja postaje sve manji kako broj čestica raste. uFastSLAM pri istom broju čestica nadmašuje ostale čestične metode, a sa znatno manjim brojem čestica može nadmašiti FastSLAM 1.0 i približiti se performansama FastSLAM 2.0,

što pokazuje da je moguće postići vrlo dobru procjenu čak i uz ograničen broj čestica. Kao posljedica, uFastSLAM predstavlja najpovoljnije rješenje u sistemima gdje su računski resursi ograničeni, a i dalje je poželjna visoka tačnost.

Buduća istraživanja biće usmjerena na proširenje analize i praktičnu validaciju algoritama u realnim uslovima. Planirano je njihovo testiranje na stvarnim mobilnim robotskim platformama opremljenim LIDAR-om, kamerama i inercijalnim senzorima, uz razmatranje metoda za rad sa nepoznatom asocijacijom mjerenja, adaptivno podešavanje kovarijansi procesnog i mjernog šuma i broja čestica (KLD-sampling), te razvoj paralelnih implementacija pogodnih za rad u realnom vremenu. Pored toga, istraživanje će obuhvatiti i grafovske i optimizacione pristupe (Graph-SLAM, Bundle Adjustment), koji omogućavaju globalnu korekciju mape u *offline* režimu i mogu se integrisati sa probabilističkim metodama u vidu hibridnih rješenja.

Dodatak

Oznaka	Opis
	Indeksi, skupovi i dimenzije
t	vremenski korak (diskretno vrijeme)
p	indeks čestice u čestičnim metodama, $p = 1, \ldots, N$
k	indeks mjerenja u trenutku t
j	indeks obilježja (landmarka) u mapi, $j = 1, \ldots, M$
N	broj čestica u FastSLAM-u
M	ukupan broj obilježja u mapi
$N_\ell^{[p]}$	broj inicijalizovanih obilježja u čestici p
n	dimenzija (augmentovanog) stanja za Unscented Transform
χ_t	skup čestica u trenutku t
	Stanje robota i mapa
x_t	pozicija robota u trenutku t (npr. $[x, y, \theta]^T$)
$x_t^{[p]}$	pozicija robota u čestici p
$\hat{\mu}_t$	procijenjeni zajednički vektor stanja (robot + mapa)
$\hat{\Sigma}_t$	kovarijansa zajedničkog stanja
$\mu_{t,j}^{[p]}$	procijenjena pozicija obilježja j u čestici p
$\Sigma_{t,j}^{[p]}$	kovarijansa obilježja j u čestici p
δ	relativni vektor robot-obilježje
	Upravljanje i modeli kretanja
u_t	upravljački ulaz robota
v_t, ω_t	translaciona i ugaona brzina robota
$g(\cdot)$	nelinearni model kretanja
G_t	Jakobijan modela kretanja (EKF)
R_t	kovarijansa šuma modela kretanja
	Mjerni model i korespondencije
z_t	skup mjerenja u trenutku t
$z_t^{[k]} = (r_t^{[k]}, \phi_t^{[k]})^T$	k-to mjerenje
c_t	asocijacije mjerenja i obilježja u vremenu t
$h(\cdot)$	nelinearni mjerni model

Oznaka	Opis
$h^{-1}(\cdot)$	inverzni mjerni model (inicijalizacija obilježja)
Q_t	kovarijansa mjernog šuma
$H_t^{[k]}$	Jakobijan mjerenja za mjerenje k (EKF)
$H_{m,j}$	Jakobijan mjerenja po obilježju j
$H_{x,j}$	Jakobijan mjerenja po stanju robota
$F_{x,j}$	ugradnja robota i obilježja j u prošireno SLAM stanje
$K_t^{[k]}$	Kalmanovo pojačanje
$S_t^{[k]}$	inovacijska kovarijansa
	Unscented Transform
$X_t^{[i]}$	i-ta sigma-tačka stanja (UKF)
$X_{t\|t-1}^{[i]}$	propagirana sigma-tačka kroz model kretanja
$X_{a,t-1}^{[i][p]}$	augmentovana sigma-tačka u čestici p (uFastSLAM)
$Z_{t,j}^{[i]}$	sigma-tačka mjerenja j (UKF)
$z_{t,j}^{[i][p]}$	sigma-tačka mjerenja u uFastSLAM-u
$W_m^{[i]}, W_c^{[i]}$	težine sigma-tačaka
λ	UT parametar
$S_a^{[p]}$	Čoleski faktor augmentovane kovarijanse (uFastSLAM)
$\Sigma_t^{x[p]}$	kovarijansa stanja robota u čestici p (uFastSLAM)
$\Sigma_t^{x,z}$	kros-kovarijansa stanje-mjerenje (UKF)
$\Sigma_t^{x,z[p]}$	kros-kovarijansa stanje-mjerenje (uFastSLAM)
	Čestične metode i težine
$w_t^{[p]}$	težinski koeficijent čestice p prije normalizacije
p_0	početna težina za novo obilježje

Tabela A: Zajednička tabela oznaka za EKF-SLAM, UKF-SLAM, FastSLAM 1.0, FastSLAM 2.0 i uFastSLAM.

Algoritam 8 EKF SLAM

function EKF_SLAM_KNOWN_CORRESPONDENCES($\hat{\mu}_{t-1}, \hat{\Sigma}_{t-1}, u_t, z_t, c_t, R_t, Q_t$)

$\mu_0 = \mathbf{0}_{(3+2n)\times 1}$

$\Sigma_0 = \begin{bmatrix} \mathbf{0}_{3\times 3} & \mathbf{0} \\ \mathbf{0} & \infty \cdot I_{2n\times 2n} \end{bmatrix}$

$F_x = \begin{bmatrix} I_{3\times 3} & \mathbf{0}_{3\times 2n} \end{bmatrix}$

$\bar{\mu}_t = \hat{\mu}_{t-1} + F_x^T \begin{bmatrix} -\frac{v_t}{\omega_t}\sin(\theta) + \frac{v_t}{\omega_t}\sin(\theta+\omega_t\Delta t) \\ \frac{v_t}{\omega_t}\cos(\theta) - \frac{v_t}{\omega_t}\cos(\theta+\omega_t\Delta t) \\ \omega_t\Delta t \end{bmatrix}$

$G_t = I + F_x^T \begin{bmatrix} 0 & 0 & -\frac{v_t}{\omega_t}\cos(\theta) + \frac{v_t}{\omega_t}\cos(\theta+\omega_t\Delta t) \\ 0 & 0 & -\frac{v_t}{\omega_t}\sin(\theta) + \frac{v_t}{\omega_t}\sin(\theta+\omega_t\Delta t) \\ 0 & 0 & 0 \end{bmatrix} F_x$

$\bar{\Sigma}_t = G_t\hat{\Sigma}_{t-1}G_t^T + F_x^T R_t F_x$

$\hat{\mu}_t = \bar{\mu}_t$

$\hat{\Sigma}_t = \bar{\Sigma}_t$

for each observed landmark $z_t^{[k]} = (r_t^{[k]}, \phi_t^{[k]})^T$ **do**

$j = c_t^{[k]}$

if landmark j was never observed **then**

$\begin{bmatrix} \mu_{j,x} \\ \mu_{j,y} \end{bmatrix} = \begin{bmatrix} \hat{\mu}_{t,x} \\ \hat{\mu}_{t,y} \end{bmatrix} + \begin{bmatrix} r_t^{[k]}\cos\big(\phi_t^{[k]} + \hat{\mu}_{t,\theta}\big) \\ r_t^{[k]}\sin\big(\phi_t^{[k]} + \hat{\mu}_{t,\theta}\big) \end{bmatrix}$

end if

$\delta = \begin{bmatrix} \mu_{j,x} - \hat{\mu}_{t,x} \\ \mu_{j,y} - \hat{\mu}_{t,y} \end{bmatrix}$

$q = \delta^T\delta$

$\bar{z}_t^{[k]} = \begin{bmatrix} \sqrt{q} \\ \text{atan2}(\delta_y, \delta_x) - \hat{\mu}_{t,\theta} \end{bmatrix}$

$F_{x,j} = \begin{bmatrix} I_{3\times 3} & \mathbf{0}_{3\times(2j-2)} & \mathbf{0}_{3\times 2} & \mathbf{0}_{3\times(2n-2j)} \\ \mathbf{0}_{2\times 3} & \mathbf{0}_{2\times(2j-2)} & I_{2\times 2} & \mathbf{0}_{2\times(2n-2j)} \end{bmatrix}$

$H_t^{[k]} = \frac{1}{q}\begin{bmatrix} -\sqrt{q}\delta_x & -\sqrt{q}\delta_y & 0 & \sqrt{q}\delta_x & \sqrt{q}\delta_y \\ \delta_y & -\delta_x & -q & -\delta_y & \delta_x \end{bmatrix} F_{x,j}$

$K_t^{[k]} = \hat{\Sigma}_t H_t^{[k]T}\big(H_t^{[k]}\hat{\Sigma}_t H_t^{[k]T} + Q_t\big)^{-1}$

$\hat{\mu}_t = \hat{\mu}_t + K_t^{[k]}(z_t^{[k]} - \bar{z}_t^{[k]})$

$\hat{\Sigma}_t = (I - K_t^{[k]}H_t^{[k]})\hat{\Sigma}_t$

end for

return $\hat{\mu}_t, \hat{\Sigma}_t$

end function

Algoritam 9 UKF SLAM

function UKF_SLAM_KNOWN_CORRESPONDENCES($\hat{\mu}_{t-1}, \hat{\Sigma}_{t-1}, u_t, z_t, c_t, R_t, Q_t$)

$S = \text{chol}\left((n+\lambda)\hat{\Sigma}_{t-1}\right)$

$X_t^{[0]} = \hat{\mu}_{t-1}$

$X_t^{[i]} = \hat{\mu}_{t-1} + S_{:,i}, \quad i = 1, \ldots, n$

$X_t^{[i]} = \hat{\mu}_{t-1} - S_{:,i-n}, \quad i = n+1, \ldots, 2n$

$X_{t|t-1}^{[i]} = g(u_t, X_t^{[i]}), \quad i = 0, \ldots, 2n$

$\bar{\mu}_t = \sum_{i=0}^{2n} W_m^{[i]} X_{t|t-1}^{[i]}$

$\bar{\Sigma}_t = \sum_{i=0}^{2n} W_c^{[i]} (X_{t|t-1}^{[i]} - \bar{\mu}_t)(X_{t|t-1}^{[i]} - \bar{\mu}_t)^T + R_t$

$\hat{\mu}_t = \bar{\mu}_t$

$\hat{\Sigma}_t = \bar{\Sigma}_t$

for each observed measurement $z_t^{[k]} = (r_t^{[k]}, \phi_t^{[k]})^T$ **do**

$j = c_t^{[k]}$

if landmark j was never observed **then**

$$\begin{bmatrix} \mu_{j,x} \\ \mu_{j,y} \end{bmatrix} = \begin{bmatrix} \hat{\mu}_{t,x} \\ \hat{\mu}_{t,y} \end{bmatrix} + \begin{bmatrix} r_t^{[k]} \cos\left(\phi_t^{[k]} + \hat{\mu}_{t,\theta}\right) \\ r_t^{[k]} \sin\left(\phi_t^{[k]} + \hat{\mu}_{t,\theta}\right) \end{bmatrix}$$

end if

$S = \text{chol}\left((n+\lambda)\hat{\Sigma}_t\right)$

$X^{[0]} = \hat{\mu}_t$

$X^{[i]} = \hat{\mu}_t + S_{:,i}, \quad i = 1, \ldots, n$

$X^{[i]} = \hat{\mu}_t - S_{:,i-n}, \quad i = n+1, \ldots, 2n$

for $i = 0$ to $2n$ **do**

$$\delta_j^{[i]} = \begin{bmatrix} \delta_{j,x}^{[i]} \\ \delta_{j,y}^{[i]} \end{bmatrix} = \begin{bmatrix} m_{j,x}^{[i]} - x^{[i]} \\ m_{j,y}^{[i]} - y^{[i]} \end{bmatrix}$$

$q_j^{[i]} = (\delta_j^{[i]})^T \delta_j^{[i]}$

$$Z_{t,j}^{[i]} = \begin{bmatrix} \sqrt{q_j^{[i]}} \\ \text{atan2}(\delta_{j,y}^{[i]}, \delta_{j,x}^{[i]}) - \theta^{[i]} \end{bmatrix}$$

end for

$\bar{z}_t^{[k]} = \sum_{i=0}^{2n} W_m^{[i]} Z_{t,j}^{[i]}$

$S_t^{[k]} = \sum_{i=0}^{2n} W_c^{[i]} (Z_{t,j}^{[i]} - \bar{z}_t^{[k]})(Z_{t,j}^{[i]} - \bar{z}_t^{[k]})^T + Q_t$

$\Sigma_t^{x,z} = \sum_{i=0}^{2n} W_c^{[i]} (X^{[i]} - \hat{\mu}_t)(Z_{t,j}^{[i]} - \bar{z}_t^{[k]})^T$

$K_t^{[k]} = \Sigma_t^{x,z} (S_t^{[k]})^{-1}$

$\hat{\mu}_t = \hat{\mu}_t + K_t^{[k]} (z_t^{[k]} - \bar{z}_t^{[k]})$

$\hat{\Sigma}_t = \hat{\Sigma}_t - K_t^{[k]} S_t^{[k]} K_t^{[k]T}$

end for

return $\hat{\mu}_t, \hat{\Sigma}_t$

end function

Algoritam 10 FastSLAM 1.0

function FastSLAM1.0_known_correspondences($\chi_{t-1}, c_t, u_t, z_t, R_t, Q_t$)
  for $p = 1$ to N **do**
    retrieve $\langle x_{t-1}^{[p]}, (\mu_{t-1,j}^{[p]}, \Sigma_{t-1,j}^{[p]})_{j=1}^{M} \rangle$ from χ_{t-1}
    sample $x_t^{[p]} \sim p(x_t \mid x_{t-1}^{[p]}, u_t, R_t)$
    $j = c_t$
    if landmark j has never been observed **then**
      $\mu_{t,j}^{[p]} = h^{-1}(x_t^{[p]}, z_t)$
      $H_j = \dfrac{\partial h}{\partial m_j}\Big|_{\mu_{t,j}^{[p]}, x_t^{[p]}}$
      $\Sigma_{t,j}^{[p]} = H_j^{-1} Q_t (H_j^{-1})^T$
      $w_t^{[p]} = p_0$
    else
      $\hat{z}_t^{[p]} = h(\mu_{t-1,j}^{[p]}, x_t^{[p]})$
      $H_j = \dfrac{\partial h}{\partial m_j}\Big|_{\mu_{t-1,j}^{[p]}, x_t^{[p]}}$
      $S_t^{[p]} = H_j \Sigma_{t-1,j}^{[p]} H_j^T + Q_t$
      $K_t^{[p]} = \Sigma_{t-1,j}^{[p]} H_j^T (S_t^{[p]})^{-1}$
      $\mu_{t,j}^{[p]} = \mu_{t-1,j}^{[p]} + K_t^{[p]}(z_t - \hat{z}_t^{[p]})$
      $\Sigma_{t,j}^{[p]} = (I - K_t^{[p]} H_j)\Sigma_{t-1,j}^{[p]}$
      $w_t^{[p]} = |2\pi S_t^{[p]}|^{-\frac{1}{2}} \exp\big(-\frac{1}{2}(z_t - \hat{z}_t^{[p]})^T (S_t^{[p]})^{-1}(z_t - \hat{z}_t^{[p]})\big)$
    end if
    for all other features $j \neq c_t$ **do**
      $\mu_{t,j}^{[p]} = \mu_{t-1,j}^{[p]}$
      $\Sigma_{t,j}^{[p]} = \Sigma_{t-1,j}^{[p]}$
    end for
  end for
  $w_t^{[p]} \leftarrow \dfrac{w_t^{[p]}}{\sum_{\ell=1}^{N} w_t^{[\ell]}}, \quad p = 1, \ldots, N$
  $\chi_t = \emptyset$
  for $p = 1$ to N **do**
    draw random p with probability $\propto w_t^{[p]}$
    add $\langle x_t^{[p]}, (\mu_{t,j}^{[p]}, \Sigma_{t,j}^{[p]})_{j=1}^{M} \rangle$ to χ_t
  end for
  return χ_t
end function

Algoritam 11 FastSLAM 2.0

function FastSLAM2.0_known_correspondences($\chi_{t-1}, c_t, u_t, z_t, R_t, Q_t$)
  for $p = 1$ to N **do**
    retrieve $\langle x_{t-1}^{[p]}, (\mu_{t-1,j}^{[p]}, \Sigma_{t-1,j}^{[p]})_{j=1}^{M} \rangle$ from χ_{t-1}
    for $j = 1$ to $N_\ell^{[p]}$ **do**
      $\hat{x}_t = g(x_{t-1}^{[p]}, u_t)$
      $\bar{z}_j = h(\mu_{t-1,j}^{[p]}, \hat{x}_t)$
      $H_{x,j} = \nabla_x h(\mu_{t-1,j}^{[p]}, \hat{x}_t)$
      $H_{m,j} = \nabla_m h(\mu_{t-1,j}^{[p]}, \hat{x}_t)$
      $Q_j = Q_t + H_{m,j} \Sigma_{t-1,j}^{[p]} H_{m,j}^T$
      $\Sigma_x = (H_{x,j}^T Q_j^{-1} H_{x,j} + R_t^{-1})^{-1}$
      $\mu_x = \Sigma_x H_{x,j}^T Q_j^{-1} (z_t - \bar{z}_j)$
      $x_t^{[p]} \sim \mathcal{N}(\mu_x, \Sigma_x)$
      $\hat{z}_j = h(\mu_{t-1,j}^{[p]}, x_t^{[p]})$
    end for
    for each observed measurement $z_t^{[k]} = (r_t^{[k]}, \phi_t^{[k]})^T$ **do**
      $j = c_t^{[k]}$
      if landmark j was never observed **then**
        $x_t^{[p]} \sim p(x_t \mid x_{t-1}^{[p]}, u_t)$
        $\mu_{t,j}^{[p]} = h^{-1}(z_t^{[k]}, x_t^{[p]})$
        $H_{m,j} = \nabla_m h(\mu_{t,j}^{[p]}, x_t^{[p]})$
        $\Sigma_{t,j}^{[p]} = H_{m,j}^{-1} Q_t (H_{m,j}^{-1})^T$
        $w_t^{[p]} = p_0$
      else if $j = c_t^{[k]}$ **then**
        $\hat{z}_t = h(\mu_{t-1,j}^{[p]}, x_t^{[p]})$
        $K = \Sigma_{t-1,j}^{[p]} H_{m,j}^T Q_t^{-1}$
        $\mu_{t,j}^{[p]} = \mu_{t-1,j}^{[p]} + K(z_t^{[k]} - \hat{z}_t)$
        $\Sigma_{t,j}^{[p]} = (I - K H_{m,j}) \Sigma_{t-1,j}^{[p]}$
        $L = H_{x,j} R_t H_{x,j}^T + H_{m,j} \Sigma_{t,j}^{[p]} H_{m,j}^T + Q_t$
        $w_t^{[p]} = |2\pi L|^{-1/2} \exp\Big(-\frac{1}{2}(z_t^{[k]} - \hat{z}_t)^T L^{-1} (z_t^{[k]} - \hat{z}_t)\Big)$
      else
        $\mu_{t,j}^{[p]} = \mu_{t-1,j}^{[p]}$
        $\Sigma_{t,j}^{[p]} = \Sigma_{t-1,j}^{[p]}$
      end if
    end for
  end for
  $\chi_t = \emptyset$
  for $p = 1$ to N **do**
    draw random p with probability $\propto w_t^{[p]}$
    add $\langle x_t^{[p]}, (\mu_{t,j}^{[p]}, \Sigma_{t,j}^{[p]})_{j=1}^{M} \rangle$ to χ_t
  end for
  return χ_t
end function

Algoritam 12 uFastSLAM_known_correspondences

function uFastSLAM_known_correspondences($\chi_{t-1}, c_t, u_t, z_t, R_t, Q_t$)
  for $p = 1$ to N **do**
    retrieve $\langle x_{t-1}^{[p]}, (\mu_{t-1,j}^{[p]}, \Sigma_{t-1,j}^{[p]})_{j=1}^{M} \rangle$ from χ_{t-1}
    $x_{a,t-1}^{[p]} = \begin{bmatrix} x_{t-1}^{[p]} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \quad \Sigma_{a,t-1}^{[p]} = \begin{bmatrix} \Sigma_{t-1}^{[p]} & 0 & 0 \\ 0 & Q_t & 0 \\ 0 & 0 & R_t \end{bmatrix}$
    $S_a^{[p]} = \text{chol}\left((n+\lambda)\Sigma_{a,t-1}^{[p]}\right)$
    $X_{a,t-1}^{[0][p]} = x_{a,t-1}^{[p]}$
    $X_{a,t-1}^{[i][p]} = x_{a,t-1}^{[p]} + S_{a,:,i}^{[p]}, \quad i = 1, \ldots, n$
    $X_{a,t-1}^{[i][p]} = x_{a,t-1}^{[p]} - S_{a,:,i-n}^{[p]}, \quad i = n+1, \ldots, 2n$
    for $i = 0$ to $2n$ **do**
      $X_{a,t-1}^{[i][p]} = \left[(x_{t-1}^{[i][p]})^T,\ (v_t^{[i][p]})^T,\ (w_t^{[i][p]})^T\right]^T$
      $x_{t|t-1}^{[i][p]} = g(u_t + v_t^{[i][p]}, x_{t-1}^{[i][p]})$
    end for
    $\bar{x}_t^{[p]} = \sum_{i=0}^{2n} W_m^{[i]} x_{t|t-1}^{[i][p]}$
    $\Sigma_t^{x[p]} = \sum_{i=0}^{2n} W_c^{[i]} (x_{t|t-1}^{[i][p]} - \bar{x}_t^{[p]})(x_{t|t-1}^{[i][p]} - \bar{x}_t^{[p]})^T$
    $x_t^{[p]} = \bar{x}_t^{[p]}$
    $\Sigma_t^{[p]} = \Sigma_t^{x[p]}$
    for each observed measurement $z_t^{[k]} = (r_t^{[k]}, \phi_t^{[k]})^T$ **do**
      $j = c_t^{[k]}$
      if landmark j was never observed in particle p **then**
        $\mu_{t,j}^{[p]} = h^{-1}(x_t^{[p]}, z_t^{[k]})$
        $H_{m,j} = \left.\dfrac{\partial h}{\partial m_j}\right|_{\mu_{t,j}^{[p]}, x_t^{[p]}}$
        $\Sigma_{t,j}^{[p]} = H_{m,j}^{-1} Q_t (H_{m,j}^{-1})^T$
        $w_t^{[p]} = p_0$
      else
        for $i = 0$ to $2n$ **do**
          $z_{t,j}^{[i][p]} = h(x_{t|t-1}^{[i][p]}, \mu_{t-1,j}^{[p]}) + w_t^{[i][p]}$
        end for
        $\hat{z}_t^{[p]} = \sum_{i=0}^{2n} W_m^{[i]} z_{t,j}^{[i][p]}$
        $S_t^{[p]} = \sum_{i=0}^{2n} W_c^{[i]} (z_{t,j}^{[i][p]} - \hat{z}_t^{[p]})(z_{t,j}^{[i][p]} - \hat{z}_t^{[p]})^T$
        $\Sigma_t^{x,z[p]} = \sum_{i=0}^{2n} W_c^{[i]} (x_{t|t-1}^{[i][p]} - \bar{x}_t^{[p]})(z_{t,j}^{[i][p]} - \hat{z}_t^{[p]})^T$
        $K_t^{[p]} = \Sigma_t^{x,z[p]} (S_t^{[p]})^{-1}$
        $x_t^{[p]} = \bar{x}_t^{[p]} + K_t^{[p]} (z_t^{[k]} - \hat{z}_t^{[p]})$
        $\Sigma_t^{[p]} = \Sigma_t^{x[p]} - K_t^{[p]} S_t^{[p]} K_t^{[p]T}$
        $w_t^{[p]} = \left|2\pi S_t^{[p]}\right|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(z_t^{[k]} - \hat{z}_t^{[p]})^T (S_t^{[p]})^{-1} (z_t^{[k]} - \hat{z}_t^{[p]})\right)$
        $\hat{z}_{t,j}^{[p]} = h(\mu_{t-1,j}^{[p]}, x_t^{[p]})$
        $H_j = \left.\dfrac{\partial h}{\partial m_j}\right|_{\mu_{t-1,j}^{[p]}, x_t^{[p]}}$

$S_{t,j}^{[p]} = H_j \Sigma_{t-1,j}^{[p]} H_j^T + Q_t$
$K_{t,j}^{[p]} = \Sigma_{t-1,j}^{[p]} H_j^T (S_{t,j}^{[p]})^{-1}$
$\mu_{t,j}^{[p]} = \mu_{t-1,j}^{[p]} + K_{t,j}^{[p]}(z_t^{[k]} - \hat{z}_{t,j}^{[p]})$
$\Sigma_{t,j}^{[p]} = (I - K_{t,j}^{[p]} H_j)\Sigma_{t-1,j}^{[p]}$
end if
end for
for all landmarks j not observed at time t **do**
$\mu_{t,j}^{[p]} = \mu_{t-1,j}^{[p]}$
$\Sigma_{t,j}^{[p]} = \Sigma_{t-1,j}^{[p]}$
end for
end for
$w_t^{[p]} \leftarrow \dfrac{w_t^{[p]}}{\sum_{\ell=1}^{N} w_t^{[\ell]}}, \quad p = 1, \ldots, N$
$\chi_t = \emptyset$
for $i = 1$ to N **do**
draw random p with probability $\propto w_t^{[p]}$
add $\langle x_t^{[p]}, (\mu_{t,j}^{[p]}, \Sigma_{t,j}^{[p]})_{j=1}^{M} \rangle$ to χ_t
end for
return χ_t
end function

Algoritam 13 Ballistic drift noise

function BALLISTIC_DRIFT_NOISE(x, P, n)
  $S = \text{chol}(P)$
  $\ell = \text{length}(x)$
  $Y = 0_{\ell \times n}$
  $b = \text{false}$
  $d = 0_\ell$
  $v_{\text{bias}} = 0_\ell, \quad p_{\text{bias}} = 0_\ell$
  for $t = 1$ to n **do**
    if $\neg b \ \wedge \ \text{rand}() < p_{\text{start}}$ **then**
      $b = \text{true}$
      $d = S\,\mathcal{N}(0, I_\ell)$
      $d = d \odot \text{mask}$
      if $\|d\| < \varepsilon$ **then**
        $d = \text{mask}$
      end if
      $d = d/\|d\|$
    else if $b \ \wedge \ \text{rand}() > p_{\text{stay}}$ **then**
      $b = \text{false}$
      $d = 0_\ell$
    end if
    $a = 0_\ell$
    if b **then**
      $a = a_{\text{mag}}\, d$
    end if
    $v_{\text{bias}} = \rho_v\, v_{\text{bias}} + a$
    $p_{\text{bias}} = \rho_p\, p_{\text{bias}} + v_{\text{bias}}$
    $w = S\,\mathcal{N}(0, I_\ell)$
    if b **then**
      $w = w + \text{cov}_{\text{scale}}\, S\,\mathcal{N}(0, I_\ell)$
    end if
    $Y_{:,t} = x + w + p_{\text{bias}}$
  end for
  return Y
end function

Bibliografija

[1] K. M. Lynch and F. C. Park, *Modern Robotics: Mechanics, Planning, and Control*, pp. 513–564. Cambridge University Press, 2017.

[2] S. Thrun, M. Montemerlo, D. Koller, B. Wegbreit, J. Nieto, and E. Nebot, "{FastSLAM}: {An} efficient solution to the simultaneous localization and mapping problem with unknown data association," *Journal of Machine Learning Research*, 2004.

[3] M. Weiss, *Data Structures & Algorithm Analysis in C++*. Pearson, 4th ed., 2013.

[4] R. Chatila and J. Laumond, "Position referencing and consistent world modeling for mobile robots," in *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, pp. 138–145, Institute of Electrical and Electronics Engineers, 1985.

[5] R. C. Smith and P. Cheeseman, "On the representation and estimation of spatial uncertainty," *The International Journal of Robotics Research*, vol. 5, pp. 56–68, 12 1986.

[6] H. F. Durrant-Whyte, "Sensor models and multisensor integration," *The International Journal of Robotics Research*, vol. 7, pp. 97–113, 12 1988.

[7] S. Rahman, R. DiPietro, D. Kedarisetti, and V. Kulathumani, "Large-scale indoor mapping with failure detection and recovery in slam," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 12294–12301, IEEE, 10 2024.

[8] X. Yan, W. Guo, Y. Wang, and S. Li, "Application of visual slam technology in military intelligent unmanned systems," in *Proceedings of the 2024 4th International Conference on Artificial Intelligence, Big Data and Algorithms*, pp. 206–211, ACM, 6 2024.

[9] J. S. Willners, Y. Carreno, S. Xu, T. Łuczyński, S. Katagiri, J. Roe, Èric Pairet, Y. Petillot, and S. Wang, "Robust underwater slam using autonomous relocalisation," *IFAC-PapersOnLine*, vol. 54, pp. 273–280, 2021.

[10] S. Thrun, B. Wolfram, and D. Fox, *Probabilistic robotics*. MIT Press, 2005.

[11] A. Josep, P. Yvan, S. Joaquim, and L. Xavier, *The SLAM problem: a survey*. 2008.

[12] S. Thrun and M. Montemerlo, "The graph slam algorithm with applications to

large-scale mapping of urban structures," in *International Journal of Robotics Research*, vol. 25, pp. 403–429, 5 2006.

[13] F. Dellaert and M. Kaess, "Factor graphs for robot perception," *Foundations and Trends in Robotics*, vol. 6, pp. 1–139, 2017.

[14] A. Juric, F. Kendes, I. Markovic, and I. Petrovic, "A comparison of graph optimization approaches for pose estimation in slam," in *2021 44th International Convention on Information, Communication and Electronic Technology (MIPRO)*, pp. 1113–1118, IEEE, 9 2021.

[15] G. Grisetti, R. Kummerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based slam," *IEEE Intelligent Transportation Systems Magazine*, vol. 2, pp. 31–43, 2010.

[16] Y. Zhang, F. Tosi, S. Mattoccia, and M. Poggi, "Go-slam: Global optimization for consistent 3d instant reconstruction," *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 3704–3714, 2023.

[17] Álvaro Parra, T.-J. Chin, A. Eriksson, and I. Reid, "Visual slam: Why bundle adjust?," 2019.

[18] M. Kaess, A. Ranganathan, and F. Dellaert, "isam: Incremental smoothing and mapping," *IEEE Transactions on Robotics*, vol. 24, pp. 1365–1378, 12 2008.

[19] T. Zheng, F. Wang, and Z. Xu, "An improved gtsam-based nonlinear optimization algorithm in orbslam3," in *2022 International Conference on Intelligent Transportation, Big Data & Smart City (ICITBS)*, pp. 30–33, IEEE, 3 2022.

[20] H. Saleem, R. Malekian, and H. Munir, "Neural network-based recent research developments in slam for autonomous ground vehicles: A review," *IEEE Sensors Journal*, vol. 23, pp. 13829–13858, 7 2023.

[21] Z. Zhao, C. Wu, X. Kong, Z. Lv, X. Du, and Q. Li, "Light-slam: A robust deep-learning visual slam system based on lightglue under challenging lighting conditions," 5 2024.

[22] Y. Zhang, P. Shi, and J. Li, "3d lidar slam: A survey," *The Photogrammetric Record*, vol. 39, pp. 457–517, 6 2024.

[23] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: A versatile and accurate monocular slam system," *IEEE Transactions on Robotics*, vol. 31, pp. 1147–1163, 10 2015.

[24] J. Engel, T. Schöps, and D. Cremers, *LSD-SLAM: Large-Scale Direct Monocular SLAM*, pp. 834–849. 2014.

[25] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," 10 2016.

[26] J. Mccormac, R. Clark, M. Bloesch, A. Davison, and S. Leutenegger, "Fusion++: Volumetric object-level slam," in *2018 International Conference on 3D Vision (3DV)*, pp. 32–41, IEEE, 9 2018.

[27] L. Huang, "Review on lidar-based slam techniques," in *2021 International Conference on Signal Processing and Machine Learning (CONF-SPML)*, pp. 163–

168, IEEE, 11 2021.

[28] A. Concha, G. Loianno, V. Kumar, and J. Civera, "Visual-inertial direct slam," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1331–1338, IEEE, 5 2016.

[29] X. Zhou and S. Roumeliotis, "Multi-robot slam with unknown initial correspondence: The robot rendezvous case," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1785–1792, IEEE, 10 2006.

[30] S. Thrun, B. Wolfram, and D. Fox, *Probabilistic robotics*. MIT Press, draft ed., 2000.

[31] F. Dekking, C. Kraaikamp, H. Lopuhaä, and L. Meester, *A Modern Introduction to Probability and Statistics*. Springer, 2005.

[32] D. C. Montgomery and G. C. Runger, *Applied Statistics and Probability for Engineers*. Wiley, 6th ed., 2013.

[33] R. G. Brown and P. Y. C. Hwang, *Introduction to Random Signals and Applied Kalman Filtering with Matlab Exercises*. IEEE Press, 3rd ed., 1996.

[34] C. Stachniss, G. D. Tipaldi, and W. Burgard, "Extended kalman filter," 2014.

[35] E. Wan and R. V. D. Merwe, "The unscented kalman filter for nonlinear estimation," in *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373)*, pp. 153–158, IEEE, 2000.

[36] G. Campion and W. Chung, *Springer Handbook of Robotics*, pp. 391–410. Springer, 1st ed., 2008.

[37] C. Stachniss, G. D. Tipaldi, and W. Burgard, "Ekf slam," 2014.

[38] T. Bailey, *Mobile Robot Localisation and Mapping in Extensive Outdoor Environments*. PhD thesis, The University of Sydney, 2002.

[39] J. Solà, "Ekf-slam: A very quick guide," 2014.

[40] G. Huang, A. I. Mourikis, and S. I. Roumeliotis, "Analysis and improvement of the consistency of extended kalman filter based slam," tech. rep., University of Minnesota, 2007.

[41] M. Dorvash, A. Eslamian, and M. R. Ahmadzadeh, "Enhanced unscented kalman filter-based slam in dynamic environments: Euclidean approach," tech. rep., Faculty, 2023.

[42] M. S. Bahraini, M. Bozorg, and A. B. Rad, "New adaptive ukf algorithm to improve the accuracy of slam," *International Journal of Robotics*, vol. 5, pp. 35–46, 2019.

[43] X. Wang and H. Zhang, "A upf-ukf framework for slam," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2007.

[44] R. Martinez-Cantin and J. A. Castellanos, "Unscented slam for large-scale outdoor environments," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, 2005.

[45] C. Stachniss, G. D. Tipaldi, and W. Burgard, "Unscented kalman filter," 2014.

[46] C. Stachniss, G. D. Tipaldi, and W. Burgard, "Short introduction to particle filters and monte carlo localization," 2014.

[47] Y. Norhidayah and B. Norlida, "Particle filter in simultaneous localization and mapping (slam) using differential drive mobile robot," *Jurnal Teknologi*, vol. 77, 2015.

[48] P. Abbeel, "Rao-blackwellized particle filtering," 2012.

[49] C. Stachniss, G. D. Tipaldi, and W. Burgard, "Fastslam – feature-based slam with particle filters," 2014.

[50] C. Kim, R. Sakthivel, and W. K. Chung, "Unscented fastslam: A robust and efficient solution to the slam problem," *IEEE Transactions on Robotics*, vol. 24, 2008.

[51] M. Raitoharju and R. Piche, "On computational complexity reduction methods for kalman filter extensions," *IEEE Aerospace and Electronic Systems Magazine*, vol. 34, pp. 2–19, 10 2019.

[52] A. E. Al-Tarras, M. I. Yacoub, M. S. Asfoor, and A.-H. Sharaf, "Computation complexity evaluation of fastslam algorithm for unmanned ground vehicles," in *2019 24th International Conference on Methods and Models in Automation and Robotics (MMAR)*, pp. 390–395, IEEE, 8 2019.

[53] G. Kock, "Performance evaluation of several slam algorithms in a feature-based vslam framework," tech. rep., The University of Twente, 5 2021.

[54] S. Samsuri, H. Zamzuri, M. A. A. Rahman, S. Mazlan, and A. H. A. Rahman, "Computational cost analysis of extended kalman filter in simultaneous localization and mapping (ekf-slam) problem for autonomous vehicle," vol. 10, pp. 7764–7768, 1 2015.

[55] G. Huang, A. Mourikis, and S. Roumeliotis, "On the complexity and consistency of ukf-based slam," in *2009 IEEE International Conference on Robotics and Automation*, pp. 4401–4408, IEEE, 5 2009.

[56] T. Sebastian, L. Yufeng, K. Daphne, N. Andrew, G. Zoubin, and D.-W. Hugh, "Simultaneous localization and mapping with sparse extended information filters," *I. J. Robotic Res.*, vol. 23, pp. 693–716, 2004.

[57] H. Shoudong, "A review of optimisation strategies used in simultaneous localisation and mapping," *Journal of Control and Decision*, vol. 6, pp. 1–14, 2018.

[58] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "Fastslam: A factored solution to the simultaneous localization and mapping problem," *Proceedings of the National Conference on Artificial Intelligence*, 2003.

[59] T. Bailey, J. Nieto, and E. Nebot, "Consistency of the fastslam algorithm," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pp. 424–429, IEEE, 5 2006.

[60] Y. Bar-Shalom, X. Li, and T. Kirubarajan, *Estimation with Applications to*

Tracking and Navigation. Wiley, 1 2002.

[61] L. H. A. Co., "Specifications utm-30lx," tech. rep., Hokuyo Automatic Co., Ltd., 12 2018.

[62] L. H. A. Co., "Specifications ust-30lx," tech. rep., Hokuyo Automatic Co., Ltd., 12 2018.

[63] A. Doucet, N. D. Freitas, and N. Gordon, *Sequential Monte Carlo methods in practice*. Springer, 2011.

[64] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "Fastslam: A factored solution to the simultaneous localization and mapping problem," in *Proceedings of the AAAI National Conference on Artificial Intelligence*, pp. 593–598, 2002.

Izjava o istovjetnosti štampane i elektronske verzije master rada

Ime i prezime autora: Miljan Golubović
Broj indeksa/upisa: 1/22
Studijski program: Automatika i industrijska elektrotehnika
Naslov rada: Uporedna analiza algoritama za simultanu lokalizaciju i mapiranje u realnom vremenu
Mentor: Žarko Zečević

Potpisani/a: Miljan Golubović

Izjavljujem

da je štampana verzija mog master rada istovjetna elektronskoj verziji koju sam predao/la za objavljivanje u Digitalni arhiv Univerziteta Crne Gore.

Istovremeno izjavljujem da dozvoljavam objavljivanje mojih ličnih podataka u vezi sa dobijanjem akademskog naziva master nauka, kao što su ime i prezime, godina i mjesto rođenja, naslov master rada i datum odbrane rada.

U Podgorici, 23.12.2025. godine

Potpis magistranda

Miljan Golubović

IZJAVA O KORIŠĆENJU

Ovlašćujem Univerzitetsku biblioteku da u Digitalnom arhivu Univerziteta Crne Gore pohrani moj master rad pod nazivom:

„Uporedna analiza algoritama za simultanu lokalizaciju i mapiranje u realnom vremenu“

koji je moje autorsko djelo.

Master rad sa svim prilozima predao/la sam u elektronskom formatu pogodnom za trajno arhiviranje.

Moj master rad pohranjen u Digitalnom arhivu Univerziteta Crne Gore mogu da koriste svi koji poštuju odredbe sadržane u odabranom tipu licence Kreativne zajednice (*Creative Commons*) za koju sam se odlučio/la.

1. Autorstvo
2. Autorstvo – nekomercijalno
3. Autorstvo – nekomercijalno – bez prerade
4. Autorstvo – nekomercijalno – dijeliti pod istim uslovima
5. Autorstvo – bez prerade
6. Autorstvo – dijeliti pod istim uslovima

(Molimo da zaokružite samo jednu od šest ponuđenih licenci, kratak opis licenci dat je na poleđini lista).

U Podgorici, 23.12.2025. godine

Potpis magistranda

Musab Tokydalebt