UNIVERZA V LJUBLJANI
FAKULTETA ZA MATEMATIKO IN FIZIKO

Matematika – 1. stopnja

Luka Avbreht

Algoritmi za pozicioniranje v brezžičnih mrežah

Delo diplomskega seminarja

Mentor: prof. dr. Sergio Cabello Justo

Ljubljana, 2018

Kazalo

1. Uvod 4
2. Definicije in teorija 4
2.1. Jacobijeva matrika 4
2.2. Minimum na premici 4
2.3. Gradientni spust 5
2.4. Metoda največjega verjetja 6
2.5. Gramova matrika 7
3. Ozadje sistema 8
3.1. Ozadje sistema 8
3.2. Modeli, neodvisni od razdalje 9
3.3. Modeli odvisni od razdalje 9
4. Algoritmi 11
4.1. Klasični multilateracijski algoritem 11
4.2. Algoritem s standardno metodo največjega verjetja 13
4.3. Algoritem z modificirano metodo največjega verjetja 16
5. Implementacija in primerjava algoritmov 18
5.1. Test 4 fiksne točke 18
5.2. Test različno generirani vhodni podatki 19
5.3. Test 6 fiksnih točk 20
5.4. Test 8 fiksnih točk 21
5.5. Vpliv števila točk na metodo največjega verjetja 22
5.6. Kolinearne fiksne točke 23
6. Primeri uporabe 24
Slovar strokovnih izrazov 25
Literatura 25

Algoritmi za pozicioniranje v brezžičnih mrežah

Povzetek

V diplomskem delu obravnavamo problem lociranja objekta v brezžičnem prostoru. Pregledamo ozadje problema ter ga dobro definiramo. Opišemo nekaj osnovnih načinov merjenja razdalje v brezžičnih mrežah. Podrobno opišemo tri algoritme ter implementiramo dva najpogostejša algoritma. V enem uporabimo predefiniran sistem ter rešitev poiščemo s pomočjo Gramove matrike, v drugem pa rešitev poiščemo s pomočjo metode največjega verjetja. Algoritma implementiramo ter s pomočjo naključno generiranih testnih primerov analiziramo njuno delovanje.

Positioning algorithms for wireless networks

Abstract

In the work, we address the problem of localisation in wireless networks. For that we analyse the background of the problem and provide precise definitions. Some standard ways of measuring distance are also described. We thoroughly describe three algorithms and implement two of them. One calculates a position in space using classical overdetermined system of equations, and the other one uses the maximal likelihood estimation model. In the end we analyse them both using some randomly generated test cases.

Math. Subj. Class. (2010): 68U01,68U20,62P30,90B18

Ključne besede: metoda največjega verjetja, lokalizacija, predoločen sistem, gradientni spust

Keywords: maximum likelihood estimation, localization, overdefined sistem, gradient descent

1. Uvod

V sodobnem svetu so brezžična omrežja postala del vsakdana. Dandanes skoraj vsak v žepu nosi mobilni telefon, vsakodnevno uporablja računalnik, celo avtomobili se dandanes povezujejo v brezžična omrežja. Vse več ljudi se poslužuje navigacije, ko se odpravijo na pot, svojim ljubljenčkom na ovratnice namestijo čipe namenjene uporabi v brezžičnih mrežah, na kolo si namestijo oddajnik, ki preprečuje krajo le tega ipd.

V diplomskem delu bomo raziskali ozadje algoritmov za ugotavljanje pozicije v brezžičnih mrežah, pregledali potrebne tehnološke rešitve, ki jih omogočajo, ter raziskali prednosti in slabosti dveh bolj pogostih algoritmov s pomočjo simulacij.

V drugem poglavju se bomo spomnili nekaterih ključnih vsebin, ki bodo prišle prav za nadaljno razumevanje diplomskega dela. V tretjem poglavju se bomo poglobili v različne oblike sistemov, pregledali grobe delitve tehnologij ter kako le te vplivajo na izbiro algoritma, ki ga uporabimo. Naslednji del diplomskega dela je namenjen izpeljavi ozadja in implementaciji najpogostejših algoritmov. V petem poglavju bomo naredili analizo dveh algoritmov, običajnega multilateracijskega in pa statističnega, ki uporablja metodo največjega verjetja. V zadnjem delu pa bomo našteli nekaj primerov uporabe algoritmov v vsakdanjem življenu.

Osnovne reference za delo so bile [4, 5, 6, 7, 9].

2. Definicije in teorija

Za razumevanje in lažje sledenje diplomski nalogi, bomo v tem poglavju pogledali nekaj algoritmov ter izrekov. Večino izrekov, algoritmov in metod, naštetih v tem poglavju smo sicer spoznali že tekom študija. Pri nekaterih ne najbolj osnovnih pa smo dodali tudi primer.

2.1. **Jacobijeva matrika.** Jacobijeva matrika $J_F(x_1, \dots x_n)$ je matrika parcialnih odvodov prvega reda velikosti $m \times n$ [10]. Naj bo $F = (f_1, \dots, f_m)$ funkcija n spremenljivk $(x_1, \dots, x_n)$, tedaj je Jacobijeva matrika funkcije $F(x_1, \dots, x_n)$ oblike:

$$J_F = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

2.2. **Minimum na premici.** Pri tem algoritmu iščemo lokalni minimum funkcije na zaprtem intervalu. Da to lahko storimo, mora biti funkcija dovolj "lepa" . Funkcija mora biti odvedljiva na $[a, b]$. Da na intervalu najdemo minimum, pa mora prav tako veljati, da je $f^{'}(a) < 0$ in $f^{'}(b) > 0$. Če funkcija ustreza zahtevam, lahko uporabimo postopek, ki je precej podoben bisekciji.

Bisekcija je ena izmed bolj zanesljivih in ena najbolj enostavnih metod za iskanje ničel realnih funkciji. Pri tem algoritmu je ideja zelo podobna. Iskanje minimuma funcije f prav tako omejimo na interval $[a, b]$. Izračunamo vrednost odvoda v srednji točki $c = \frac{a+b}{2}$, od kjer dobimo naslednjo informacijo:

$$f^{'}(c) = \begin{cases} 0 & c \text{ je ekstrem funkcije } f \\ > 0 & \text{minimum funkcije } f \text{ bomo iskali na intervalu } [a, c] \\ < 0 & \text{minimum funkcije } f \text{ bomo iskali na intervalu } [c, b] \end{cases}$$

Tak postopek ponavljamo, dokler ni razlika med dvema zaporednima izračunanima vrednostima manjša od določene tolerance. V diplomskem delu bomo uporabili naslednji algoritem 1.

Algoritem 1 Bisekcija Minimum

Vhod: funkcija f, krajišči intervala $[a, b]$ in pa željena toleranca tol
Izhod: približek za minimum funkcije f

```
1: procedure BisekcijaMin(f, a, b, tol)
2:     korak = tol * 10^-2
3:     while b − a > tol do
4:         točka = (a + b)/2
5:         vred = (f(točka+korak) − f(točka))/ korak
6:         if abs(vred) < tol then
7:             return točka
8:         else
9:             if vred ≤ 0 then
10:                a = točka
11:            else
12:                b = točka
13:            end if
14:        end if
15:    end while
16:    return točka
17: end procedure
```

Ker odvoda funkcije v našem primeru v splošnem ne poznamo, v algoritmu uporabimo diferenčni količnik. Za približek za odvod funkcije torej uporabimo:

$$f'(x) \approx \frac{f(x+h) - f(x)}{h},$$

kjer je h dovolj majhen v primerjavi s toleranco, v našem primeru $h = tol * 10^{-2}$.

2.3. **Gradientni spust.** Gradientni spust je iterativna metoda prvega reda za iskanje lokalnega minimuma funkcije okoli začetne točke [8]. Za nas je zanimiva, ker jo lahko uporabimo tudi za iskanje ničel sistema nelinearnih enačb, pri čemer moramo poznati le lokalno obliko sistema. Obstaja tudi sorodna metoda za iskanje lokalnega maximuma, ki pa je v delu ne bomo omenjali.

Metoda sloni na predpostavki, da je funkcija $F(x)$, katere minimum iščemo, odvedljiva v okolici začetne točke a_0. Vemo, da bomo minimum v naslednjem koraku iskali v točkah, ki so v smeri negativnega odvoda F v trenutni točki a. Sledi torej, da bomo za naslednji približek izbrali kar točko

$$a_{n+1} = a_n - \gamma \nabla F(a_n)$$

za dovolj majhne γ, tako da velja

$$F(a_n) > F(a_{n+1}).$$

V iteracijskem koraku se torej od trenutne točke a_n premaknemo v smeri negativnega odvoda proti minimumu. Izbira vrednosti γ se lahko spremeni v vsakem koraku. Za izračun te vrednosti smo v delu uporabili zgoraj opisan algoritem za iskanje minimuma na premici s pomočjo bisekcije.

Nas bo zanimalo iskanje ničel funkcije $G\colon \mathbf{R}^n \to \mathbf{R}^n$, se pravi takšne vrednosti x, da velja $G(x) = 0$. Če definiramo funkcijo $F(x)$ na naslednji način

$$F(x) = \frac{1}{2} \cdot G(x)^T G(x),$$

pri čemer oznaka $G(x)^T$ pomeni transponiranje matrike $G(x)$, potem je iskanje ničel funkcije $G(x)$ ekvivalentno iskanju minimuma funkcije $F(x)$. Za ta problem pa lahko uporabimo gradientni spust.

Metoda uporablja algoritem Gradientni spust prikazan v točki 2. V algoritmu uporabimo zgoraj opisano Jacobijevo matriko funkcije $G = (g_1 \dots g_n)$ oblike:

$$Jacobijanka(G, X_0) = \begin{bmatrix} \frac{\partial g_1(X_0)}{\partial x_1} & \frac{\partial g_1(X_0)}{\partial x_2} & \cdots & \frac{\partial g_1(X_0)}{\partial x_n} \\ \frac{\partial g_2(X_0)}{\partial x_1} & \frac{\partial g_2(X_0)}{\partial x_2} & \cdots & \frac{\partial g_2(X_0)}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial g_n(X_0)}{\partial x_1} & \frac{\partial g_n(X_0)}{\partial x_2} & \cdots & \frac{\partial g_n(X_0)}{\partial x_n} \end{bmatrix}$$

Algoritem 2 Ničle funkcije $G\colon \mathbf{R}^n \to \mathbf{R}^n$

Vhod: funkcija G, začetni približek X_0 in željena toleranca tol
Izhod: približek za ničlo funkcije G v okolici X_0

```
1: procedure MINIMUM FUNKCIJE(G, x_0, tol)
2:     prib = x_0
3:     razlika = tol * 10
4:     prejšnja = 1/2 · G(prib)^T G(prib)
5:     while razlika > tol do
6:         deltaf = Jacobijanka(G, X_0)^T × G(X_0)          //Gradient F(X_0)
7:         opt_delta = BisekcijaMin(λγ : prib −γ deltaf , −10, 10, tol^2)
8:         prib = prib - opt_delta·deltaf
9:         mini = 1/2 · G(prib)^T G(prib)
10:        razlika = abs(prejšna - mini)
11:        prejšna = mini
12:    end while
13:    return prib
14: end procedure
```

2.4. **Metoda največjega verjetja.** Metoda največjega verjetja je statistična metoda za ocenjevanje parametrov statističnega modela [3]. Uporabna je pri reševanju mnogih problemov v statistiki, v tem diplomskem delu pa jo bomo uporabili za reševanje problema lokalizacije točke v prostoru.

Naj bodo $X_1, X_2, \dots, X_n$ slučajne spremenljivke, ki imajo skupno gostoto podano kot $f_X(X_1, X_2, \dots, X_n|\theta)$ in so med seboj neodvisne. Potem funkcijo verjetja $L(\theta)$ za parameter θ definiramo kot funkcijo odčitkov $x_1, x_2, \dots, x_n$:

$$L(\theta) = f_X(x_1, x_2, \dots, x_n|\theta).$$

Rezultat po metodi največjega verjetja bo maksimum funkcije $L(\theta)$.

Če pa poznamo gostoto vsakega parametra posebej, lahko do komulativne porazdelitve pridemo z množenjem funkcij gostote. Torej za vsak X_n je njegova gostota

definirana kot $f_i(x_i|\theta)$. Funkcija verjetja tega sistema v odvisnoti od parametra θ bo torej

$$L(\theta) = \prod_{i=1}^{n} f_i(x_i|\theta).$$

Brez škode za rešitev problema lahko namesto ekstrema funkcije $L(\theta)$, iščemo ekstrem funkcije $\log(L(\theta))$. Dobimo torej problem oblike

$$\log(L(\theta)) = \sum_{i=1}^{n} \log(f_i(x_i)).$$

Da poiščemo maksimum funkcije $\log(L(\theta))$, kjer je $\theta = (\theta_1, \theta_2, \ldots, \theta_n)$, pa ponavadi rešujemo sistem enačb

$$\begin{aligned}
\frac{\partial \log(L(\theta))}{\partial \theta_1} &= 0, \\
\frac{\partial \log(L(\theta))}{\partial \theta_2} &= 0, \\
&\vdots \\
\frac{\partial \log(L(\theta))}{\partial \theta_n} &= 0.
\end{aligned}$$

2.5. **Gramova matrika.** Naj bo matrika A matrika sistema m enačb z n neznankami, pri čemer je $m > n$. Torej matrika velikosti $m \times n$. Naj bo b vektor dimenzije m. Potem lahko približek za rešitev sistema

$$Ax = b$$

iščemo po metodi najmnajših kvadratov, kot je opisano v knjigi [2].

Iščemo torej vektor $x \in \mathbb{R}^n$ ki bo minimiziral vsoto kvadratov razlik

$$\sum_{i=1}^{m} (A_i x - b_i)^2$$

oziroma tak vektor x, da bo norma

$$||Ax - b||_2$$

minimalna. Predpostaviti moramo tudi to, da je matrika A polnega ranga, se pravi da velja $rang(A) = n$.

Ta problem rešimo tako, da obema stranema sistema primnožimo matriko A^T, tako dobimo primer običajnega sistema:

$$A^T Ax = A^T b.$$

Matriki $A^T A$ pravimo tudi Gramova matrika.

Se pravi, da rešitev predoločenega sistema $Ax = b$ prevedemo na običajen sistem za Gramovo matriko $A^T A$. Le to rešimo na enega izmed standardnih načinov. Najenostavnejši je s pomočjo iskanja inverza Gramove matrike.

Rešitev lahko torej zapišemo kot

$$x = (A^T A)^{-1} A^T b.$$

3. Ozadje sistema

3.1. **Ozadje sistema.** Z razvojem brezžičnih omrežji in brezžičnih komunikacij, se je pojavila potreba po podatku o lokacij posameznih uporabnikov sistema. Zametki takšnih tehnologij sodijo v prvo polovico prejšnega stoletja pri uporabi radarjev, sprva le za vojaške namene. Kasneje pa tudi z razširitvijo na komercialni trg, predvsem letališča in pristanišča. Največji razcvet je sistem doživel v začetku devedesetih let prejšnjega stoletja, ko je obrambno ministrstvo ZDA zasnovalo sistem GPS. Dandanes se lahko v omrežje GPS priključi vsak uporabnik s primernim sprejemnikom. Vendar pa zaradi raznih omejitev sistem ni popoln, predvsem zaradi cene vzrdrževanja, slabšega delovanja v zaprtih prostorih itd.

V ta namen je bilo zato v zadnjem času razvitih in predlaganih mnogo sistemov in algoritmov za reševanje težave pozicioniranja v brezžičnih prostorih. Najbolj groba delitev takih sistemov je na tiste, ki se za svoje delovanje zanašajo na odčitek izmerjene razdalje, in tiste, ki tega podatka ne potrebujejo, ampak pri delovanju uporabljajo druge podatke. V obeh primerih za delovanje potrebujemo nekaj točk (oddajnikov), ki imajo v prostoru fiksno lokacijo oz. lahko to lokacijo izračunamo z uporabo višje-nivojske lokalizacije, običajno GPS. Od števila fiksnih točk je odvisna natančnost in zanesljivost obeh sistemov. Minimalno število fiksnih točk pri reševanju tovrstnega problema na ravnini, se pravi dvo-dimenzionalnem prostoru, je tri, pri lokalizaciji na sferi oz kakšnem drugem tridimenzionalnem prostoru pa štiri, včasih celo več.

Najpogosteje nas zanima lokacija, včasih pa tudi hitrost ter smer, v katero se iskana točka ali množica le teh v prostoru premika. Sistem je običajno sestavljen iz senzorjev, ki so strateško razvrščeni v prostoru ter iskanega objekta, opremljenega s potrebno tehnologijo, ki omogča njegovo delovanje. Le to lahko grobo razdelimo na 6 najpogostejših korakov opisanih v viru [5].

Zaznavanje iskanega objekta v prostoru je prvi in običajno ključen korak za delovanje sistema. Preden sploh lahko začnemo z reševanje problema lokalizacije, moramo namreč vzpostaviti potrebno komunikacijo med objektom in senzorji.

Naslednji korak je komunikacija med senzorji v prostoru in sporočanje potrebnih informaciji o obliki prostora glavnemu računalniku. V kolikor točke v prostoru niso fiksne, v tem koraku senzor izračuna svojo lokacijo s pomočjo višjenivojske lokalizacije. Senzorji in programska oprema v tem koraku pripravijo primerno obliko informacije za uporabo v naslednjem koraku.

Nato sledi uporaba primernega algoritma za izračun lokacije v prostoru glede na obliko prostora. Na ta korak se bomo v nadaljevanju nabolj osredotočili.

V kolikor sistem omogoča delno izklaplanje posameznih senzorjev in s tem varčevanje z energijo, v tem koraku s pomočjo prejšnih izračunov lokacije, sistem predvidi nadaljevanje poti objekta v prostoru, ter s tem njegovo naslednjo lokacijo, ter aktivira senzorje v njeni okolici.

Varčevanje z energijo sistema, kjer na podlagi izračuna iz prejšnega koraka, v delovanju ohranimo le tisti del sistema, ki pokriva prostor, kjer naj bi se objekt nahajal v naslednjem trenutku.

In ponovno iskanje objekta v prostoru, v kolikor prejšnji korak odpove. V tem koraku se nekako vrnemo nazaj na prvega, ter celoten postopek ponovimo. V kolikor objekta v prostoru ne izgubimo, pa nadaljujemo z drugim korakom.

3.2. **Modeli, neodvisni od razdalje.** V diplomskem delu se sicer sistemom, ki za izračun lokacije ne uporabljajo odčitka razdalj, ne bomo podrobneje posvečali, vendar pa so po svoje zelo zanimivi, ter zato vredni omembe.

Modeli take vrste običajno za svoje delovanje uporabljajo le podatek o tem, katere fiksne točke znotraj sistema so objekt v prostoru zaznale. Namesto razdalje tako vsak senzor posebej pridobi podatek $P(s_i)$ oblike:

$$P(s_i) = \begin{cases} 1 & ; \text{če je senzor } s_i \text{ zaznal objekt,} \\ 0 & ; \text{drugače.} \end{cases}$$

Tak model za izračun lokacije uporabi zgoraj opisno binarno funkcijo, in s pomočjo oblike prostora in lokacij senzorjev izračuna razdaljo. Najpogostejši algoritem, uporabljen v takšnem modelu, je algoritem, imenovan DV-Hop, katerega podrobnejši opis lahko najdemo v [6].

3.3. **Modeli odvisni od razdalje.** Bolj popularni in običajno tudi bolj natančni pa so modeli, ki za svoje delovanje uporabljajo izmerjeno razdaljo med senzorjem v prostoru, in pa objektom ki ga locirajo, ali pa podatek o azimutu (kotu) med senzorjem in iskanim objektom. Osredotočili se bomo predvsem na sisteme, ki se zanašajo na izmerjeno razdaljo.

3.3.1. *Merjenje razdalje.* Razdaljo je moč izmeriti in izračunati na več različnih načinov. Ker je ta korak ponavadi najmanj natančen, in zato za seboj nosi največjo napako, je pametno da se spoznamo z nekaterimi od bolj pogostih metod za izračun razdalje, zato jih bomo nekaj opisali. Opisi principov so izvelček iz vira [4, strani 2-5].

3.3.2. *RSSI.* RSSI (angleško: Received signal strength Indicator) je definirana kot napetost oddanega signala na prejemniku. Prednost takega sistema je, da sam signal ne nosi nobenega podatka, in je zato lahko bolj natančen, vendar pa ne omogoča lociranja večih objektov istočasno. Oddaljenost izračunamo s pomočjo podatka o hitrosti širjenja valovanja v prostoru (odvisno s kakšnim valovanjem imamo opravka).

Izračun razdalje deluje na principu naslednje enačbe:

$$P_{rx} = c \times \frac{P_{tx}}{d^{\alpha}}.$$

Torej je razdalja med prejemnikom in oddajnikom enaka

$$d = \sqrt[\alpha]{c \times \frac{P_{tx}}{P_{rx}}},$$

pri čemer so:

- P_{tx} — moč oddanega signala,
- α — koeficient izgube moči signala,
- P_{rx} — moč prejetega signala,
- c — hitrost valovanja,
- d — razdalja.

V praksi se je izkazalo, da je podatek, izmerjen na ta način, precej nenatančen. Tudi če sta oba, oddajnik in prejemnik, negibna. Eden izmed razlogov je problem simetričnosti/sferičnosti antene, ki signal oddaja. Običajne antene signala ponavadi namreč ne oddajajo enako močno v vse smeri. Velik problem pa so prav tako senčne cone, ki nastanejo zaradi za signal neprodušnih ovir. Prav tako je za natančnost

potrebna kalibracija vezij, kar pa je v praksi precej zahtevno in zato lahko precej drago.

Ta princip običajno uporabljajo sistemi v katerih med objektom in senzorjem ni velikih razdalj. Naprimer lokalizacija objekta znotraj stavb s pomočjo Bluetooth ali WIFI tehnologije.

3.3.3. *ToA.* ToA (angleško: Time of Arrival) je princip merjenja razdalje med oddajnikom in prejemnikom glede na časovno razliko med oddanim signalom in prejetim signalom, ki s pomočjo hitrosti valovanja v mediju določi razdaljo med senzorjema. To stori na naslednji način:

$$d = (T_2 - T_1) * v_p,$$

pri čemer je:

- T_1 — čas, ko je bil signal poslan,
- T_2 — čas, ko je bil signal prejet,
- v_p — hitrost valovanja.

V praksi imamo najpogosteje opravka z elektromagnetnim valovanjem v zraku (cca 1 bar), kjer ima valovanje hitrost približno $V_c = 2,99792458 \cdot 10^{-8}$ m/s. Za to potrebujemo ure z zelo veliko natančnostjo, kar je sicer precej zahtevno in drago, ampak v praksi najbolj zanesljivo. Obstaja tudi metoda dvosmernega časa prihoda (angleško: Two Way Time of Arrival), ki ne zahteva sinhronizacije ur. Več o principu v naslednjem odstavku.

Tak pristop uporablja najbolj znan sistem za določanje lokacije GPS. Sistem je namreč zelo natančen in omogoča merjenje ogromnih razdalj med sprejemnikom in oddajnikom, v tem primeru prejemnikom na zemlji in geostacionarnim satelitom v orbiti.

3.3.4. *TDoA in TWToA.* Tako TDoA (angleško: Time Difference of Arrival) in TWToA (angleško: Two Way Time of Arrival) v svojem delovanju uporabljata principe kot prej opisani ToA, le da za svoje delovanje potrebujeta zelo natančno kalibriran oddajnik, ne pa tudi prejemnika (včasih tudi obratno).

TDoA pri svojem delovanju uporablja dve vrsti valovanja v prostoru, ki se širita z različno hitrostjo. Običajno sta to elektro-magnetno valovanje in zvočno valovanje. S pomočjo razlike v času prejetega signala, ki je posledica različnih hitrosti valovanja, izračuna razdaljo med senzorjem in objektom. Hitrost elektromagnetnega valovanja je namreč, kot že rečeno $V_c = 2,99792458 \cdot 10^{-8}$ m/s, hitrost zvočnega valovanja pa $V_s = 331.2$ m/s.

TWToA pa deluje na podoben način kot ToA, vendar pa le ta namesto sinhronizacije ur na obeh koncih, zahteva le dovolj natančno uro na oddajniku, ter podatek o času, ki ga sprejemnik potrebuje za odgovor. Oddajnik namreč pošlje signal, in ko le ta pripotuje do prejemnika, ta nanj odgovori, ter nato oddajnik s pomočjo časovne razlike, ki zaradi prepotovane poti nastane, izračuna prepotovano pot.

3.3.5. *Merjenje kotov.* Merjenje kotov oz AoA (angleško: Angle of Arrival) je metoda, ki določi smer, iz katere prihaja signal. Običajno se pri teh odčitkih uporablja razlika o času prihoda signala na dve vzporedni anteni, kar omogoča izračun kota, pod katerim je signal anteni zadel.

4. Algoritmi

Naslednji korak v reševanju problema je predpriprava podatkov, ki jih bomo kasneje uporabljali v algoritmu. Le ta je običajno tesno povezana in prilagojena samemu algoritmu. Podatki so običajno razporejeni v logično podatkovno strukturo. Prav tako je potrebno predhodno poznavanje lokacij senzorjev in običajno tudi oblike prostora, v katerem sistem rešujemo. Primerna predpriprava podatkov lahko namreč precej olajša pisanje algoritma in pogosto omogoča hitrejše izvananje le tega.

Običajno pred začetkom izvajanja algoritma predpostavimo, da se nahajamo v ravnini, ali pa v nekem tridimenzionalnem prostoru. Poznati moramo lokacije n fiksnih točk $P_i = (x_i, y_i), i = 1 \ldots, n$, kjer mora biti n primerno velik, da rešitev sploh lahko določimo. Kot vhodni podatek prav tako dobimo odčitke razdalj r_i med i-to fiksno točko in objektom v prostoru $P = (x, y)$. Predpostavke od tu naprej se ponavadi razlikujejo od algoritma do algoritma, zato bomo več o tem povedali pri vsakem opisanem algoritmu posebej.

Za izračun lokacije v prostoru, kot že rečeno, obstaja mnogo algoritmov. Ti so zelo odvisni od zgoraj opisanih metod merjenja in predpriprave podatkov, prav tako pa se med seboj razlikujejo v predpostavkah, ki jih pred začetkom izvajanja privzamemo. V delu si bomo podrobneje ogledali več algoritmov, enega deterministočnega, in dva ki temeljita na metodah iz statistike.

4.1. **Klasični multilateracijski algoritem.**

Prvi izmed algoritmov, ki jih bomo pogledali podrobneje, je običajni multilateracijski algoritem, ki za podatke o razdaljah uporablja izmerjen podatek kot točno vrednost. Torej na ta podatek v algoritmu ne gledamo kot na slučajno spremenljivko, več o tem pa kasneje. Pred začetkom izvajanja algoritma vemo, da se nahajamo na dvodimenzionali ravnini. Na njej poznamo lokacije n fiksnih točk $P_i, i = 1, 2 \ldots n$, kjer mora biti $n \geq 3$. Na ravnino si lahko postavimo običajni koordinatni sistem, ter pozicije fiksnih točk opišemo s koordinatami $P_i = (x_i, y_i), i = 1, 2 \ldots n$. Vsaka od teh fiksnih točk ima možnost merjenja razdalje med seboj, in neznano točko v prostoru, katere lokacijo iščemo. To točko označimo s $P = (x, y)$. Za vhodni podatek v algoritmu torej dobimo koordinate fiksnih točk, ter odmerke $r_i, i = 1, 2 \ldots n$, ki predstavljajo razdaljo izmerjeno med i-to fiksno točko ter iskano točko P.

Problema se lotimo tako, da rešimo sistem nelinearnih enačb:

$$\begin{aligned}
(x_1 - x)^2 + (y_1 - y)^2 &= r_1^2, \\
(x_2 - x)^2 + (y_2 - y)^2 &= r_2^2, \\
&\vdots \\
(x_n - x)^2 + (y_n - y)^2 &= r_n^2.
\end{aligned}$$

V splošnem bomo torej dobili predefiniran sistem. Pri izpeljavi je najbolj preprosto rešiti sistem, v katerem nastopajo le 3 enačbe, ki ga bomo lahko kasneje posplošili na predefiniran sistem.

Rešujemo torej sistem enačb:

$$\begin{aligned}
(x_1 - x)^2 + (y_1 - y)^2 &= r_1^2, \\
(x_2 - x)^2 + (y_2 - y)^2 &= r_2^2, \\
(x_3 - x)^2 + (y_3 - y)^2 &= r_3^2.
\end{aligned}$$

Vsako enačbo posebej lahko zapišemo kot:

$$\begin{aligned} r_i^2 &= (x_i - x)^2 + (y_i - y)^2 \\ &= x_i^2 - 2x_i x + x^2 + y_i^2 - 2y_i y + y^2 \\ &= (x^2 + y^2) - 2x_i x - 2y_i y + (x_i^2 + y_i^2) \end{aligned}$$

oziroma v vektorski obliki:

$$||P||^2 - 2 \cdot P \cdot P_i + ||P_i||^2 = r_i^2$$

Če sedaj v sistemu treh enačb od prve in druge odštejemo tretjo, dobimo sistem:

$$\begin{aligned} ||P||^2 - ||P||^2 - 2 \cdot P \cdot P_1 + 2 \cdot P \cdot P_3 + ||P_1||^2 - ||P_3||^2 &= r_1^2 - r_3^2 \\ ||P||^2 - ||P||^2 - 2 \cdot P \cdot P_2 + 2 \cdot P \cdot P_3 + ||P_2||^2 - ||P_3||^2 &= r_2^2 - r_3^2 \end{aligned}$$

Enačbi okrajšamo ter obrnemo, da dobimo:

$$\begin{aligned} 2 \cdot P \cdot (P_1 - P_3) &= ||P_1||^2 - ||P_3||^2 - r_1^2 + r_3^2 \\ 2 \cdot P \cdot (P_2 - P_3) &= ||P_2||^2 - ||P_3||^2 - r_2^2 + r_3^2 \end{aligned}$$

Iz zgornjega sistema lahko izrazimo vektor P kot stolpec, ki predstavlja iskano točko $P = (x, y)$ na naslednji način:

$$AP = b,$$

pri čemer sta matrika A in vektor b oblike:

$$\begin{aligned} A &= \begin{bmatrix} 2(x_1 - x_3) & 2(y_1 - y_3) \\ 2(x_2 - x_3) & 2(y_2 - y_3) \end{bmatrix}, \\ b &= \begin{bmatrix} x_1^2 - x_3^2 + y_1^2 - y_3^2 + r_3^2 - r_1^2 \\ x_2^2 - x_3^2 + y_2^2 - y_3^2 + r_3^2 - r_2^2 \end{bmatrix}, \end{aligned}$$

Če problem posplošimo na n enačb, pridemo do predefiniranega sistema:

$$AP = b$$

kjer so

$$\begin{aligned} A &= \begin{bmatrix} 2(x_1 - x_n) & 2(y_1 - y_n) \\ \vdots & \vdots \\ 2(x_{n-1} - x_n) & 2(y_{n-1} - y_n) \end{bmatrix}, \\ b &= \begin{bmatrix} x_1^2 - x_n^2 + y_1^2 - y_n^2 + r_n^2 - r_1^2 \\ \vdots \\ x_{n-1}^2 - x_n^2 + y_{n-1}^2 - y_n^2 + r_n^2 - r_{n-1}^2 \end{bmatrix}, \\ P &= \begin{bmatrix} x & y \end{bmatrix}^T \end{aligned}$$

Rešitev za sistem lahko poiščemo z metodo, opisano v razdelku 2.5. Matriki A z leve primnožimo matriko A^T, ter tako dobimo običajen sistem za Gramovo matriko $A^T A$, katere rešitev poiščemo kot

$$P^T = (A^T A)^{-1} A^T b.$$

Podrobnejši opis postopka je opisan v algoritmu 3.

Algoritem 3 Predoločen Sistem

Vhod: Seznam velikosti $2 \times n$ koordinat n fiksnih točk, seznam velikosti $1 \times n$ odmerkov razdalj od fiksnih točk do iskane točke

Izhod: Koordinati x in y iskane točke P

procedure PredoločenSistem($koordinate, odmerki$)
    $n = len(koordinate)$
    $A = int[n][2]$
    $B = int[n][1]$
    $x_n = koordinate[n-1][0]$
    $y_n = koordinate[n-1][1]$
    $r_n = odmerki[n-1]$
    for $index = 0; index < n \leftarrow$
        to $x_i = koordinate[index][0]$ **do**
        $y_i = koordinate[index][1]$
        $r_i = odmerki[index]$
        $A[index] = [2(x_i - x_n), 2(y_i - y_n)]$
        $B[index] = [x_i^2 - x_n^2 + y_i^2 - y_n^2 - r_i^2 + r_n^2]$
    end for
    $P = (A^T A)^{-1} A^T B$
    return P^T
end procedure

4.2. **Algoritem s standardno metodo največjega verjetja.** V algoritmu bomo uporabili prej opisano metodo največjega verjetja. Prav tako kot prejšnji algoritem, tudi ta rešuje podoben problem. Pred začetkom izvajanja prav tako predpostavimo, da se nahajamo na dvodimanzionalni ravnini. Tako kot v prejšnjem primeru poznamo lokacije n fiksnih točk, opremljenih z merilci razdalje. Spet nas zanima lokacija neznane točke $P = (x, y)$, ki jo bomo izračunali s pomočjo odmerkov razdalje $r_i, i = 1, 2 \ldots n$, ki predstavljajo izmerjeno razdaljo med fiksnimi točkami in neznano točko. Za razliko od prejšnega primera, pa na te odčitke gledamo kot na nepristranske cenilke. Predpostavimo, da je njihova porazdelitev normalna, ter da je varianca vedno enaka in neodvisna od razdalje. Za naš primer so razdalje torej porazdeljene s porazdelitvijo $N(r_i, \sigma^2)$. Prav tako privzamemo, da so izmerjeni odčitki r_i med seboj neodvisni.

Rešitev bomo poiskali z prej opisano metodo največjega verjetja. Najprej se spomnimo gostote slučajne spremenljivke X, porazdeljene normalno s parametroma $N(\mu, \sigma^2)$:

$$f_x(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp \frac{-(x-\mu)^2}{2\sigma^2}.$$

Za vsak odčitek torej poznamo njeno gostoto porazdelitve:

$$f_i(r_i|x, y) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp \frac{-(r_i - \mu_i)^2}{2\sigma^2},$$

kjer je

$$\mu_i = \sqrt{(x - x_i)^2 + (y - y_i)^2},$$

x_i, y_i pa koordinati fiksne točke.

Funkcija verjetja za ta sistem je torej

$$L(x,y) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp \frac{-(\mu_i - r_i)^2}{2\sigma^2}.$$

Zaradi monotonosti logaritemske funkcije lahko rešitev za cenilko vektorja (x, y) iščemo kot ekstrem logaritma funkcije verjetja. Iščemo torej ekstreme funkcije:

$$\begin{aligned}
\log(L(x,y)) &= \sum_{i=1}^{n} \log(\frac{1}{\sqrt{2\pi\sigma^2}} \exp \frac{-(\mu_i - r_i)^2}{2\sigma^2}) \\
&= n \log(\frac{1}{\sqrt{2\pi\sigma^2}}) + \sum_{i=1}^{n} \log(\exp \frac{-(\mu_i - r_i)^2}{2\sigma^2})) \\
&= n \log(\frac{1}{\sqrt{2\pi\sigma^2}}) + \sum_{i=1}^{n} \frac{-(\mu_i - r_i)^2}{2\sigma^2} \\
&= n \log(\frac{1}{\sqrt{2\pi\sigma^2}}) + \frac{1}{2\sigma^2} \sum_{i=1}^{n} -(\mu_i - r_i)^2
\end{aligned}$$

Sistem bomo reševali tako, da bomo poiskali stacionarne točke funkcije $\log(L(x, y))$. Rešitev torej dobimo kot ničle parcialnih odvodov funkcije $\log(L(x, y))$ po x in y. Rešujemo torej sistem enačb:

$$\frac{\partial \log(L(x,y))}{\partial x} = \sum_{i=1}^{n} (x - x_i) \frac{(\mu_i - r_i)}{\mu_i} = 0$$
$$\frac{\partial \log(L(x,y))}{\partial y} = \sum_{i=1}^{n} (y - y_i) \frac{(\mu_i - r_i)}{\mu_i} = 0$$

Ta sistem bomo rešili s pomočjo algoritma Gradientni spust, opisanega v razdelku 2.3. Še enkrat se spomnimo vhodnega podatka. Imamo seznam fiksnih točk $P_i = (x_i, y_i), i = 1, \ldots, n$ in pa izmerjene razdalje $r_i, i = 1 \ldots, n$. Za reševanje problema, si bomo predhodno pripravili naslednje pomožne funkcije.

Prva funkcija nam bo omogočala izračun vektorja parcialnih odvodov funkcije $\log(L(x, y))$ v točki (x_0, y_0). Rezultat bo torej oblike $[\frac{\partial \log(L(x_0,y_0))}{\partial x}, \frac{\partial \log(L(x_0,y_0))}{\partial y}]$. To storimo s pomočjo algoritma 4.

Algoritem 6 torej poišče ničlo funkcije $G(x, y) = (\frac{\partial \log(L(x,y))}{\partial x}, \frac{\partial \log(L(x,y))}{\partial y})$.

Naslednja funkcija, ki jo bomo uporabili v algoritmu, izračuna skalarni produkt funkcije $G(x, y)$ same z seboj, ter dobljeno vrednost pomnoži z $\frac{1}{2}$. Ta funkcija je namreč del algoritma za gradientni spust:

$$F(x,y) = \frac{1}{2} G(x,y)^T G(x,y).$$

Pri iskanju optimalnega parametra *optdelta* v vsakem koraku uporabimo funkcijo $funkcf(delta)$, ki je definirana kot:

$$funkcf(\xi) = F(x_k - \xi \cdot \text{deltaf}[0], y_k - \xi \cdot \text{deltaf}[1])$$

Še zadnja funkcija, ki si jo pripravimo pred začetkom izvajanja algoritma, pa je funkcija, ki vrne Jacobijevo matriko funkcije $G(x, y)$. Le to si pripravimo na podoben način, kot smo si pripravili funkcijo G. Pripravimo si torej matriko oblike

Algoritem 4 G(x,y)

Vhod: točki x in y v prostoru, seznam koordinat fiksnih točk $fiksne\_tocke$ in seznam odmerkov *odmerki*

Izhod: Vektor velikosti 2×1, ki predtavljajo vrednosti parcialnega odvoda $\log(L(x,y))$ po x in pa po y. Torej vektor oblike $[\frac{\partial \log(L(x,y))}{\partial x}, \frac{\partial \log(L(x,y))}{\partial y}]$

procedure G($x, y, fiksne\_tocke, odmerki$)
    resX = 0
    resY = 0
    for $index = 0$; $index < len(fiksne\_tocke) \leftarrow$
        to $x_i = fiksne\_tocke[index][0]$ **do**
        $y_i = fiksne\_tocke[index][1]$
        $r_i = odmerki[index]$
        resX += $(x - x_i)(\sqrt{(x_i - x)^2 + (y_i - y)^2} - r_i)/(\sqrt{(x_i - x)^2 + (y_i - y)^2})$
        resY += $(y - y_i)(\sqrt{(x_i - x)^2 + (y_i - y)^2} - r_i)/(\sqrt{(x_i - x)^2 + (y_i - y)^2})$
    end for
    return [resX, resY]
end procedure

$$\begin{bmatrix} \frac{\partial G_1(x,y)}{\partial x} & \frac{\partial G_1(x,y)}{\partial y} \\ \frac{\partial G_2(x,y)}{\partial x} & \frac{\partial G_2(x,y)}{\partial y} \end{bmatrix}.$$

To storimo s pomočjo algoritma 5.

Algoritem 5 jacobian_G(x,y)

Vhod: točki x in y v prostoru, seznam koordinat fiksnih točk $fiksne\_tocke$ in seznam odmerkov *odmerki*

Izhod: Matrika velikosti 2×2, ki predstavlja vrednosti parcialnih odvodov $G(x,y)$ po x in pa po y

procedure JACOBIAN_G($x, y, fiksne\_tocke, odmerki$)
    resX = 0
    resY = 0
    resXY = 0
    for $index = 0$; $index < len(fiksne\_tocke) \leftarrow$
        to $x_i = fiksne\_tocke[index][0]$ **do**
        $y_i = fiksne\_tocke[index][1]$
        $r_i = odmerki[index]$
        $mii = \sqrt{(x_i - x)^2 + (y_i - y)^2}$
        resX += $((2(x_i - x)^2 \frac{1}{mii} - mii + r_i)mii - (2(x_i - x)^2(mii - r_i)\frac{1}{mii}))/mii^2$
        resY += $((2(y_i - y)^2 \frac{1}{mii} - mii + r_i)mii - (2(y_i - y)^2(mii - r_i)\frac{1}{mii}))/mii^2$
        resXY += $((2(x_i - x)(y_i - y)) - (2(x_i - x)(y_i - y)(mii - r_i)\frac{1}{mii}))/mii^2$
    end for
    return [[resX, resXY], [resXY, resY]]
end procedure

S tako pripravljenimi pomožnimi funkicjami se lotimo glavnega algoritma. Pred začetkom izvajanja se moramo odločiti za željeno toleranco (v diplomskem delu v kasnejšem poglavju z implementacijo algoritmov uporabimo 10^{-6}). Ker gre za

iterativno metodo in obstaja možnost, da začetni približek ne privede do primerne rešitve, omejimo tudi maksimalno število korakov iteracije. Da poiščemo rešitev $P = (x, y)$ po metodi največjega verjetja, uporabimo sledeči algoritem 6.

Algoritem 6 Metoda največjega verjetja

Vhod: Seznam velikosti $2 \times n$ koordinat n fiksnih točk $fiksnetocke$, seznam velikosti $1 \times n$ odmerkov razdalj od fiksnih točk do iskane točke *odcitki*, začetni približek (x_0, y_0), toleranca *tol* in maksimalno število korakov *maxit*
Izhod: Koordinati x in y iskane točke P

```
1: procedure METODA_NAJVEČJEGA_VERJETJA(fiksne_tocke, odcitki, x_0, y_0, tol, maxit)
2:     x_k = x_0
3:     y_k = y_0
4:     razlika = 3 · tol
5:     prejšnja = (1/2) G(x_k, y_k)^T G(x_k, y_k)
6:     koraki = 0
7:     while razlika > tol & koraki < maxit do
8:         nabla = jacobian_G(x_k, y_k)^T · G(x_k, y_k)
9:         opt_delta = BisekcijaMin((λγ : (x_k, y_k) − γ nabla , −10, 10, tol ∗ 10^−2)
10:        x_k = x_k − opt_delta · nabla[0]
11:        y_k = y_k − opt_delta · nabla[1]
12:        mini = (1/2) G(x_k, y_k)^T G(x_k, y_k)
13:        razlika = abs(prejšnja - mini)
14:        prejšna = mini
15:        koraki += 1
16:    end while
17:    return [x_k, y_k]
18: end procedure
```

V algoritmu o pomožnih funkcijah parametra $fiksne\_tocke$ in *odcitki* ne pišemo, saj se skozi sam algoritem ne spreminjajo, in jih lahko uporabimo kot privzete argumente.

4.3. **Algoritem z modificirano metodo največjega verjetja.** V diplomskem delu si bomo poleg prejšnih dveh algoritmov podrobneje ogledali še enega, ki pa ga kasneje ne bomo implementirali in testirali.

Tudi ta algoritem sloni na metodi največjega verjetja, le da pri tem za razliko od standardnega modela upoštevamo napake meritev v odvisnosti od izmerjene razdalje. V algoritmu na razdalje torej gledamo kot na slučajne spremenljivke, katerih varianca ni več konstantna, ampak je odvisna od odčitka na senzorju. Tako kot v prejšnih algoritmih tudi tukaj predpostavimo, da se nahajamo na dvo-dimenzionalni ravnini, v kateri imamo n fiksnih točk $P_i = (x_i, y_i), i = 1, \ldots, n$, opremljenih s senzorji, ki omogočajo odmerek razdalje do iskane točke $P = (x, y)$. Odčitki pa so za razliko od prejšnega algoritma tukaj oblike $r_i = d_i(1 + \xi)$, kjer je ξ porazdeljen normalno s parametri $N(0, \sigma^2)$, d_i pa so dejanske razdalje med iskano točko $P(x, y)$ in fiksnimi točkami P_i, torej $d_i = \sqrt{(x - x_i)^2 + (y - y_i)^2}$.

Vsak odčitek r_i ima torej funkcijo gostote oblike

$$f_r(r_i) = \frac{1}{\sqrt{2\pi\sigma^2}d_i} \exp \frac{-(r_i - d_i)^2}{2\sigma^2 d_i^2}.$$

Ker predpostavimo, da so odčitki medseboj neodvisni, je njihova skupna gostota oblike

$$f_R(r_1, \ldots, r_n) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}d_i} \exp \frac{-(r_i - d_i)^2}{2\sigma^2 d_i^2}.$$

Funkcija verjetja, katere maximum bomo iskali, bo torej oblike

$$L(x, y) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}d_i} \exp \frac{-(r_i - d_i)^2}{2\sigma^2 d_i^2}.$$

Tako kot v prejšnih primerih lahko zaradi monotonosti logaritma iščemo ekstrem funkcjie

$$\begin{aligned}
\log(L(x, y)) &= \sum_{i=1}^{n} \log(\frac{1}{\sqrt{2\pi\sigma^2}d_i} \exp \frac{-(r_i - d_i)^2}{2\sigma^2 d_i^2}) \\
&= n \log(\frac{1}{\sqrt{2\pi\sigma^2}}) + \sum_{i=1}^{n} \left(\log(\frac{1}{d_i}) + \frac{-(r_i - d_i)^2}{2\sigma^2 d_i^2} \right) \\
&= -n \log(\sqrt{2\pi}\sigma) - \sum_{i=1}^{n} \log(d_i) - \sum_{i=1}^{n} \frac{(r_i - d_i)^2}{2\sigma^2 d_i^2}.
\end{aligned}$$

Prav tako kot v prejšnjem primeru, poiščemo ekstrem funkcije verjetja tako, da rešimo sistem enačb parcialnih odvodov zgornjega izraza, odvajanega po x in po y.

$$\frac{\partial \log(L(x, y))}{\partial x} = 0$$

$$\frac{\partial \log(L(x, y))}{\partial y} = 0$$

Oziroma bolj natančno:

$$\sum_{i=1}^{n} \left(\frac{(x - x_i)}{(x - x_i)^2 + (y - y_i)^2} + \frac{r_i(x - x_i)(\sqrt{(x - x_i)^2 + (y - y_i)^2} - r_i)}{\sigma^2((x - x_i)^2 + (y - y_i)^2)^2} \right) = 0$$

$$\sum_{i=1}^{n} \left(\frac{(y - y_i)}{(x - x_i)^2 + (y - y_i)^2} + \frac{r_i(y - y_i)(\sqrt{(x - x_i)^2 + (y - y_i)^2} - r_i)}{\sigma^2((x - x_i)^2 + (y - y_i)^2)^2} \right) = 0$$

Zaradi oblike sistema pa metoda, ki smo jo uporabili v prejšnem primeru, ne deluje dobro, zato je v diplomskem delu nismo implementirali in preizkusili.

5. Implementacija in primerjava algoritmov

V diplomskem delu smo, standardni algoritem za predoločen sistem opisan v razdelku 4.1, in algoritem z metodo največjega verjetja iz razdelka 4.2 implementirali, in preizkusili na nekaj testnih primerih.

Za implementacijo smo uporabili programski jezik Python 3. Znotraj programa za testiranje smo uporabili knjižnico random za generiranje naključnih števil, ki smo jih uporabili kot odčitek razdaljah med fiksno točko in iskano točko v prostoru. Za prikazovanje izračunanih simulaciji smo uporabili knjižnico matplotlib, za numerične izračune in linearno algebro pa knjižnico numpy. Prav tako smo uporabili nekaj osnovnih matematičnih funkcij, ki so implementirane v knjižnici math.

Pri vseh testnih primerih smo si določili ravnino velikosti $n \times m$ enot ter v njej izbrali pozicije nekaj fiksnih točk. V vseh primerih smo določili točko, kjer se naš objekt nahaja. Nato smo med iskano točko in fiksnimi točkami izmerili razdaljo d_i, ter iz te izračunane razdalje naredili naključno razdaljo porazdeljeno normalno s parametroma $r_i = N(d_i, \sigma)$, pri čemer smo parameter σ spreminjali pri vsakem testiranju posebej.

Pri vsakem testiranju bomo prav tako priložili sliko histograma, ki prikazuje vrednosti napake. Le te dobimo kot absolutno vrednost razdalje med izračunano lokacijo in lokacijo objekta v prostoru.

5.1. **Test 4 fiksne točke.** Pri prvem testu smo primerjali eno in drugo metodo na ravnini velikosti 100×100 enot. Na ravnini smo določili štiri fiksne točke na robovih območja. Za lokacijo fiksne točke smo določili koordinato blizu središča ravnine natančneje na koordinatah $(30, 30)$. Razdaljo smo izračunali na 1000 različnih vhodnih podatkih o razdalj med fiksno točko in iskano točko, ter na vsakem vhodnem podatku uporabili prvi in drugi algoritem po enkrat. Za generiranje naključnih podatkov smo v tem primeru uporabljali funkcijo iz pythonove knjižnice random, ki je razdalje generirala kot slučajne spremenljivke:

$$r_i \sim N(d_i, 0.08 \cdot d_i),$$

kjer je d_i realna izračunana razdalja med objektom in fiksno točko.

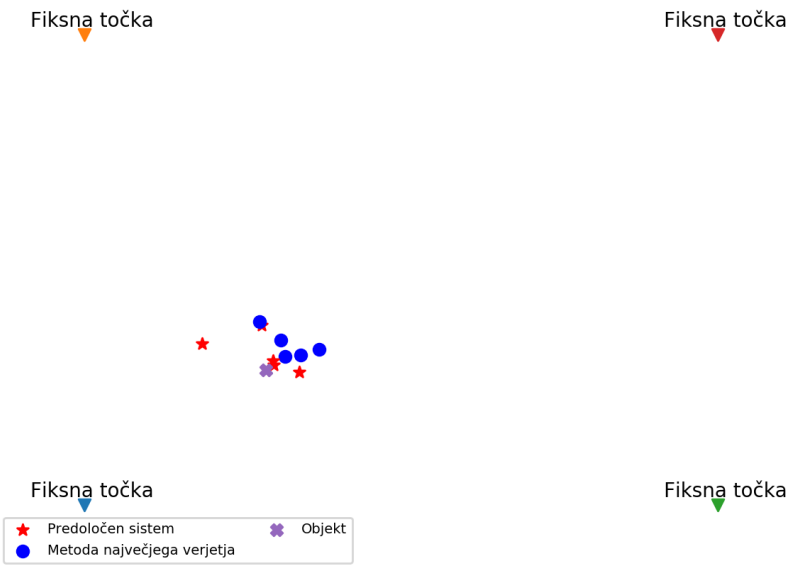
Slika 1 prikazuje območje, na katerem so označene fiksne točke ter nekaj izračunanih rezultatov. Le teh je zaradi preglednosti manj, kot jih uporabimo v testu.

Za vsako meritev smo izračunali oddaljenost od izračunane točke ter realne točke, ter iz izmerjenih algoritmov dobili podatke, ki jih bomo predstavili na naslednjih histogramih.

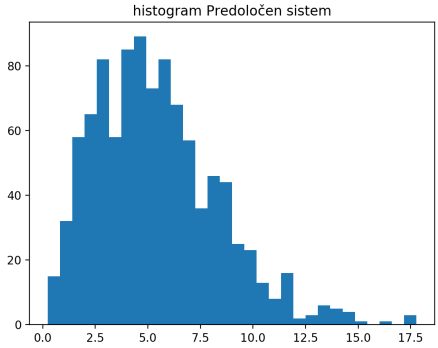
Slika 2 predstavlja histogram napak odmerkov, izračunanih po metodi s predoločenim sistemom, slika 3 pa histogram odmerkov, izračunanih po drugi metodi. V tabeli 1 pa so predstavljeni numerični rezultati meritve.

Tabela 1. Parametri testa primerjave različnih vhodnih podatkov

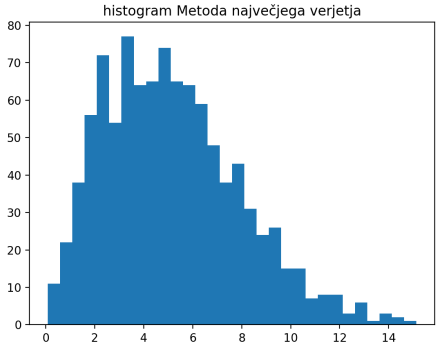
	Predoločen sistem	Največje verjetje
povprečna vrednost	5.42592189	5.18230497
maksimalna napaka	17.76305641	15.12772467
minimalna napaka	0.23956991	0.08480078
standardni odklon	2.92349831	2.77232031
relativni standardni odklon	0.53880214	0.53495893



Slika 1. Testni primer 1, Območje testa



Slika 2. Histogram metode 1



Slika 3. Histogram metode 2

Vse številke so v enotah, ki jih uporabljamo pri velikosti tabele. V tem primeru spomnimo, da je ravnina velikosti 100×100 enot. Povprečna napaka pri prvi metodi je po pričakovanju nekoliko večja od napake v drugi metodi. Tudi vsi ostali parametri kažejo na to, da je algoritem, ki uporablja metodo največjega verjetja, bolj natančen od algoritma, ki rešuje predefiniran sistem.

5.2. **Test različno generirani vhodni podatki.** V tem testu bomo primerjali algoritem, ki uporablja metodo največjega verjetja, s samim seboj. V tem primeru bomo prav tako kot prej postavili enako ravnino in fiksne točke posadili na isto mesto. Razlika bo le v generiranju testnih podatkov. V prvem primeru bomo podatke o razdalji pokvarili podobno kot prej. Njihova porazdelitev bo torej normalna s parametri:

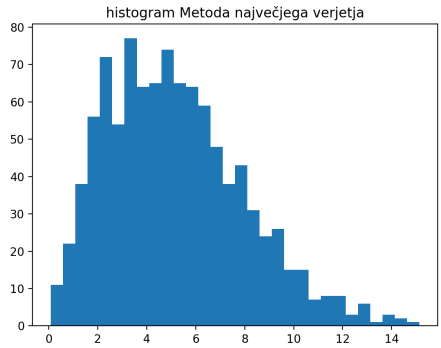
$$r_i \sim N(d_i, 0.08 \cdot d_i).$$

V drugem primeru pa bomo podatke o razdalji pokvarili z drugačnimi parametri. V tem primeru bodo razdalje porazdeljene s parametri:

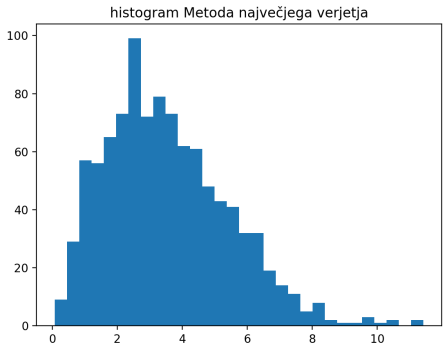
$$r_i \sim N(d_i, 4).$$

V obeh primerih je d_i dejanska razdalja med objektom in fiksno točko.

Tako kot prej smo vsako meritev ponovili na 1000 vhodnih podatkih. Histrogrami napake meritve so prestavljeni v slikah histogramov 4 in 5.



Slika 4. Histogram porazdelitve 1



Slika 5. Histogram porazdelitve 2

Obdelani podatki obeh meritev, so tako kot v prejšnjem primeru prikazani v tabeli 2.

Tabela 2. Parametri testa primerjave različnih vhodnih podatkov

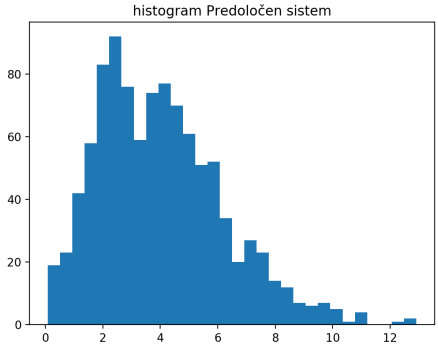
	$N(d_i, 0.1 \times d_i)$	$N(d_i, 4)$
povprečna vrednost	5.18230497	3.54004837
maksimalna napaka	15.12772467	11.41578845
minimalna napaka	0.08480078	0.07736554
standardni odklon	2.77232031	1.89347292
relativni standardni odklon	0.53495893	0.53487204

Primerjava te metode na različnih vhodnih podatkih je zanimiva, ker pri algoritmu, ki koordinate točke izračuna s pomočjo metode največjega verjetja predpostavimo, da so odčitki razporejeni z parametroma $r_i = N(d_i, C)$, kjer je C konstantna. Iz tega razloga so rezultati, ki smo jih pridobili s tem testom pričakovani. Metoda očitno deluje bolje, če podatek pokvarimo na isti način, kot smo to predpostavili v metodi.

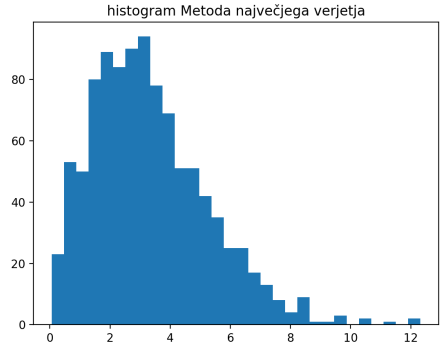
5.3. **Test 6 fiksnih točk.** V tem testu bomo primerjali obe metodi na testni plošči s šestimi fiksnimi točkami. Plošča bo spet velikosti 100×100, testne točke pa bodo vse znotraj lika, ki ga fiksne točke definirajo, nekje blizu središča. Vse meritve bomo ponovili 1000 krat. Vsako metodo bomo na podatkih uporabili po enkrat. Podatke o razdaljah bomo pridobili kot slučajne spremenljivke $r_i = \sim N(d_i, 0.1 \cdot d_i)$, pri čemer je d_i spet razdalja med objektom in fiksno točko. Slika 6 prikazuje skico območja, na katerem algoritem testiramo.

V tabeli 3 so prikazani numerični rezultati tega testa. Sliki 7 in pa 8 predstavljata histograma absolutnih napak obeh algoritmov.

Slika 6. Testni primer 3, območje testa



Slika 7. Histogram metode 1



Slika 8. Histogram metode 2

Tabela 3. Parametri testa primerjave različnih vhodnih podatkov

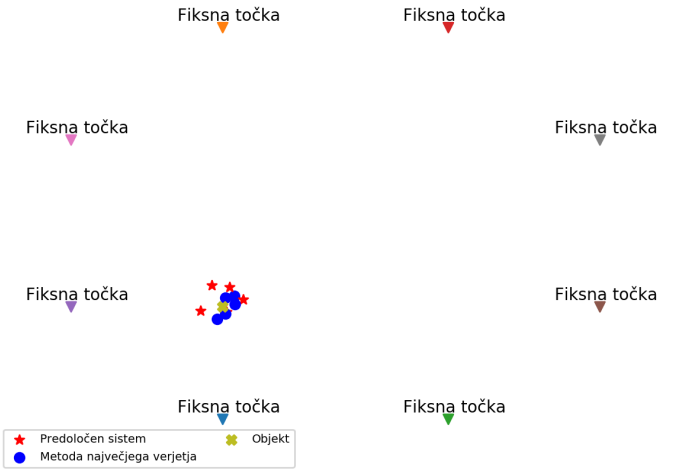
	Predoločen sistem	Največje verjetje
povprečna vrednost	4.02544213	3.33272914
maksimalna napaka	12.91022723	12.31462629
minimalna napaka	0.08730603	0.06545813
standardni odklon	2.19201867	1.92374751
relativni standardni odklon	0.54454110	0.57722888

Opazimo lahko, da je tu razlika med obema metodama precej večja. Standardni odklon in relativni standardni odklon sta sicer precej podobna, vendar pa je razlika med povprečno napako med metodama precej večja in sicer kar 0.7 enote. Prav tako lahko opazimo, da je povprečna napaka v tem primeru v primerjavi z primerom kjer smo imeli le štiri fiksne točke manjša.

5.4. **Test 8 fiksnih točk.** Pri tem testu bomo nadalje povečali število fiksnih točk. Spet bomo test naredili na ravnini velikosti 100×100 enot. Vse izmerjene meritve bodo v teh istih enotah. Razdalje med fiksnimi točkami in objektom bodo tako kot

prej oblike $r_i \sim N(d_i, 0.08 \cdot di)$, kjer je d_i resnična razdalja med objektom in fiksno točko.

Slika 9 prikazuje testno ravnino za ta primer s fiksnimi točkami, objektom in pa le nekaj izračunanimi točkami, za bolšo preglednost.



Slika 9. Testni primer 4, Območje testa

Pri testiranju smo lokacijo s pomočjo obeh algoritmov izračunali na 1000 vhodnih podatkih. V tabeli 4 so prikazani rezultati testa.

Tabela 4. Parametri testa primerjave različnih vhodnih podatkov

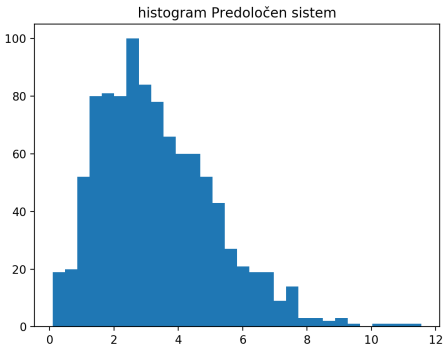
	Predoločen sistem	Največje verjetje
povprečna vrednost	3.36983117	2.96446719
maksimalna napaka	11.54965043	10.78595106
minimalna napaka	0.10056098	0.15406786
standardni odklon	1.80656849	1.57231621
relativni standardni odklon	0.53610059	0.53038745

Podobno kot pri prejšnji primerjavi lahko opazimo, da algoritem, ki uporablja metodo največjega verjeta, pride do boljših rezultatov. Sliki 10 in 11 prikazujeta histograma absolutnih vrednosti napake izračuna.

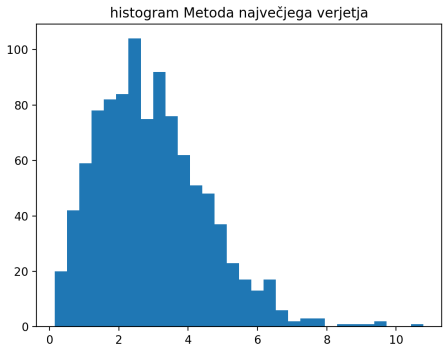
5.5. **Vpliv števila točk na metodo največjega verjetja.** Pri tem testu nas je zanimal vpliv števila točk na metodo, ki za svoje delovanje uporablja metodo največjega verjetja. Vse teste smo naredili na plošči velikosti 100×100. Spreminjali smo le število fiksnih točk. Podatki o razdalji so bili v vseh primerih pokvarjeni na enak način. Se pravi $r_i \sim N(d_i, 0.08 \cdot di)$, tako kot v prejšnjih primerih.

Teste smo ponovili na plošči s štirimi, šestimi in osmimi fiksnimi točkami, na enakih pozicijah kot pri testih 5.1, 5.3 in pa 5.4. V tabeli 5 lahko vidimo razultate testov.

Opazimo lahko, da število fiksnih točk močno vpliva na delovanje algoritma. Prav-tako je lahko razvidno, da je napredek ,kar se tiče povprečne napake, največji, ko imamo namesto štirih šest fiksnih točk. Prav tako vidimo, da se standardni odklon manjša, ko dodajamo fiksne točke.



Slika 10. Histogram metode 1

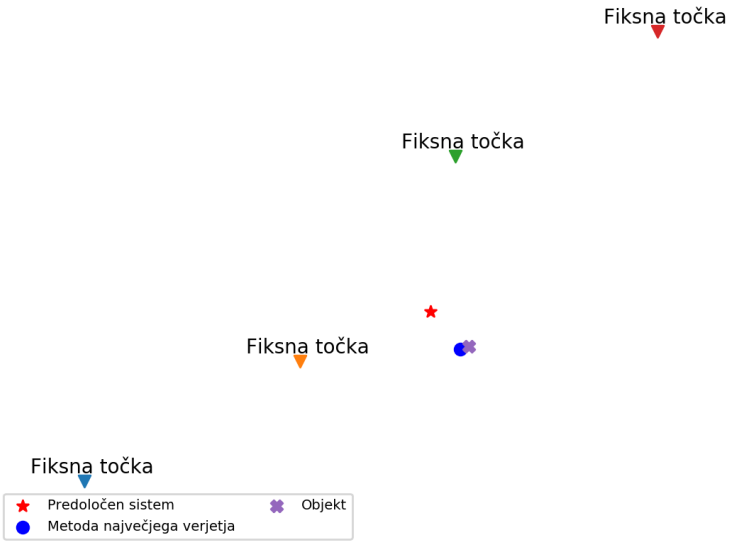


Slika 11. Histogram metode 2

Tabela 5. Parametri testa primerjave različnih vhodnih podatkov

	štiri	šest	osem
povprečna vrednost	5.18230497	3.33272914	2.96446719
maksimalna napaka	15.12772467	12.31462629	10.78595106
minimalna napaka	0.08480078	0.06545813	0.15406786
standardni odklon	2.77232031	1.92374751	1.57231621
relativni standardni odklon	0.53495893	0.57722888	0.53038745

5.6. **Kolinearne fiksne točke.** Pri tem testu nas je zanimalo delovanje algoritmov v primeru, ko fiksne točke niso v prostoru niso razpršene. Za fiksne točke smo vzeli pet točk na 100×100 ravnini, ki so skoraj kolinearne, iskano lokacijo objekta pa smo naključno spreminjali. Takšno meritev smo ponovili 3000 krat. Slika 12 prikazuje lokacijo fiksnih točk na ravnini in izračun lokacije enega primera.

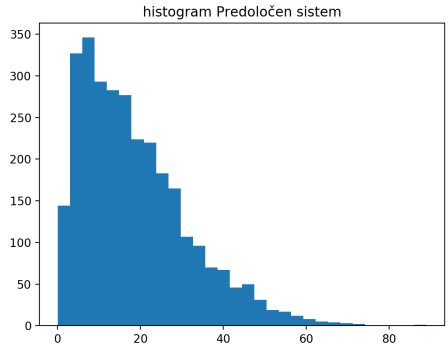


Slika 12. Testni primer 6, Območje testa

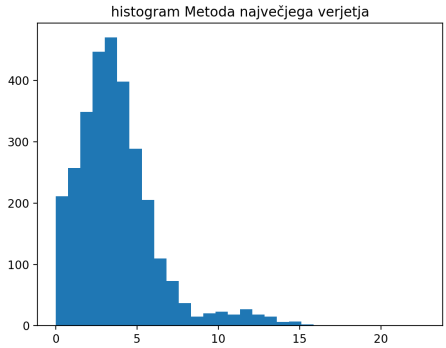
Tabela 6 prikazuje rezultate zgoraj opisanega testa, sliki 13 in 14 pa prikazujeta histograma napak meritev.

Tabela 6. Parametri testa primerjave različnih vhodnih podatkov

	Predoločen sistem	Največje verjetje
povprečna vrednost	18.65469406	3.77543953
maksimalna napaka	89.00009894	22.65535818
minimalna napaka	0.07059485	0.00811294
standardni odklon	12.97046594	2.52527286
relativni standardni odklon	0.69529234	0.66886857



Slika 13. Histogram metode 1

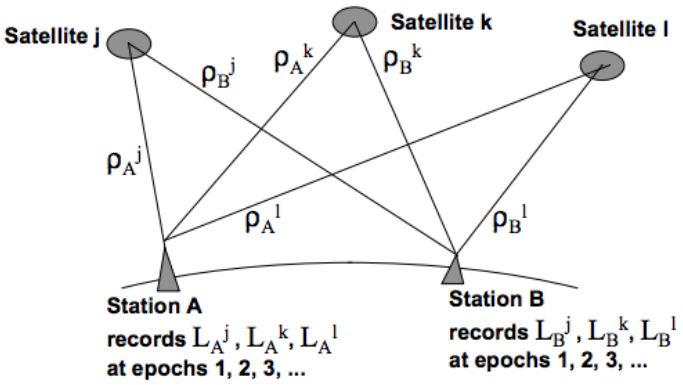


Slika 14. Histogram metode 2

Opazimo lahko, da pri takšni, manj optimalni postavitvi fiksnih točk, deluje algoritem, ki uporablja metodo največjega verjetja, precej bolje od tistega, ki rešuje predoločen sistem. Razlika v povprečni napaki je tu približno 15 enot, kar je res veliko. Prav tako je rešitev s pomočjo predoločenega sistema precej bolj razpršena (ima velik standardni odklon). V primerjavi s prejšnjimi meritvami opazimo, da pozicija fiksnih točk močno vpliva na zanesljivost rešitve sistema.

6. Primeri uporabe

Tovrstni algoritmi se, kot že omenjeno, vse bolj pogosto pojavljajo v sodobnem svetu. Med nami verjetno ni osebe, ki se v življenju še ni poslužila GPS naprave, uporabljala pametnega telefona itd. Vendar pa to še zdaleč ni vse. Tovrstni sistemi se pojavljajo v najrazličnejših okoljih. V proizvodniji, kjer lahko podjetja s pomočjo sledenja premičnih strojev in vozičkov optimizirajo delovni proces. V transportni industriji, kjer lociranje posameznih vozil omogoča večji nadzor nad dejavnostjo, prisili voznike tovornih vozil k potrebnim počitkom in tako posledično zmanjšuje število prometnih nesreč. Podvodne oblike takih sistemov pristaniščem omogočajo kontrolo prometa. Avtobusnim in transportnim podjetjem lahko sistem za lociranje vozil znotraj garaž in servisnih služb omogoča hitrejši in bolj nadzorljiv delovni porces, ter s tem prepreči marsikatero mehansko napako, še preden se ta zgodi. Prav tako so ti sistemi vse bolj popularni v zaščitenih naravnih območjih. S pomočjo pozicioniranja živali in predvidevanjem njihovega premikanja lahko človek dobi še več informaciji o življenju le teh, ter prepreči zatavanje živali v naseljena območja.



Slika 15. Delovanje sistema GPS, iz vira [1]

Slovar strokovnih izrazov

RSSI Moč radijskega signala

ToA Princip merjenja razdalje s pretečenim časom, ki ga signal potrebuje med točkama

TDoA Princip merjenja razdalje z razliko v času, ki ga signal potrebuje med točkama

TWToA Princip dvosmirnega merjenja razdalje s pretečenim časom, ki ga signal potrebuje med točkama

Literatura

[1] G. Blewitt, *Basics of the GPS technique: observation equations*, Department of Geomatics, University of Newcastle, Newcastle upon Tyne, NE1 7RU, United Kingdom,1997

[2] B. Plestenjak, *Razširjen uvod v numerične metode*, DMFA – založništvo, Ljubljana 2010

[3] J. A. Rice, *Mathematical statistics and data analysis*, Third Edition, Duxbury Press, 2007

[4] F. Santos, *Localization in wireless sensor networks* preprint (2008) [ogled 4. 9. 2018], dostopno na `http://web.ist.utl.pt/ist150077/doc/localization.pdf`

[5] E. L. Souza, E. F. Nakamura, R. W. Pazzi *Target tracking for sensor networks: a survey*, ACM Computing Surverys, Vol. 49 No. 2, Article 30 (2016)

[6] H. Tang, Q. Wang, Z. Liu, *A wireless sensor network DV-Hop localization algorithm*, ISCA (2013)

[7] Y. Zhang, Lichun Bao, Shih-Hsien Yang, Max Welling, Di Wu: *Localization algorithms for wireless sensor retrival*, The Computer Journal, Vol. 53 No. 10, 2010

[8] *Gradient descent*, v: Wikipedia: The Free Encyclopedia, [ogled 30. 4. 2018], dostopno na `https://en.wikipedia.org/wiki/Gradient_descent`

[9] *Indoor positioning system*, v: Wikipedia: The Free Encyclopedia, [ogled 25. 4. 2018], dostopno na `https://en.wikipedia.org/wiki/Indoor_positioning_system`

[10] *Jacobijeva matrika*, v: Wikipedia: The Free Encyclopedia, [ogled 1. 5. 2018], dostopno na `https://sl.wikipedia.org/wiki/Jacobijeva_matrika`