KAUNO TECHNOLOGIJOS UNIVERSITETAS

ELEKTROS IR ELEKTRONIKOS FAKULTETAS

Povilas Čižauskas

MOBILAUS ROBOTO MARŠRUTO TIKSLINIMO ALGORITMO SUKŪRIMAS IR TYRIMAS

Baigiamasis magistro projektas

Vadovas
Doc. dr. Virginijus Baranauskas

KAUNAS, 2015

KAUNO TECHNOLOGIJOS UNIVERSITETAS
ELEKTROS IR ELEKTRONIKOS FAKULTETAS
AUTOMATIKOS KATEDRA

MOBILAUS ROBOTO MARŠRUTO TIKSLINIMO ALGORITMO SUKŪRIMAS IR TYRIMAS

Baigiamasis magistro projektas
Valdymo technologijos (621H66001)

Vadovas
(parašas) Doc. dr. Virginijus Baranauskas
(data)

Recenzentas
(parašas) Lekt. dr. Kęstas Rimkus
(data)

Projektą atliko
(parašas) Povilas Čižauskas
(data)

KAUNAS, 2015

KAUNO TECHNOLOGIJOS UNIVERSITETAS

ELEKTROS IR ELEKTRONIKOS

(Fakultetas)

Povilas Čižauskas

(Studento vardas, pavardė)

Valdymo technologijos, 621H66001

(Studijų programos pavadinimas, kodas)

Baigiamojo projekto „Mobilaus roboto maršruto tikslinimo algoritmo sukūrimas ir tyrimas“

AKADEMINIO SĄŽININGUMO DEKLARACIJA

20 ____ . _____ _____ .

Kaunas

Patvirtinu, kad mano **Povilo Čižausko** baigiamasis projektas tema „Mobilaus roboto maršruto tikslinimo algoritmo sukūrimas ir tyrimas“ yra parašytas visiškai savarankiškai, o visi pateikti duomenys ar tyrimų rezultatai yra teisingi ir gauti sąžiningai. Šiame darbe nei viena dalis nėra plagijuota nuo jokių spausdintinių ar internetinių šaltinių, visos kitų šaltinių tiesioginės ir netiesioginės citatos nurodytos literatūros nuorodose. Įstatymų nenumatytų piniginių sumų už šį darbą niekam nesu mokėjęs.

Aš suprantu, kad išaiškėjus nesąžiningumo faktui, man bus taikomos nuobaudos, remiantis Kauno technologijos universitete galiojančia tvarka.

_____ _____

(vardą ir pavardę įrašyti ranka) *(parašas)*

Čižauskas, P. Mobilaus roboto maršruto tikslinimo algoritmo sukūrimas ir tyrimas. *Magistro* baigiamasis projektas / vadovas doc. dr. Virginijus Baranauskas; Kauno technologijos universitetas, Elektros ir elektronikos fakultetas, automatikos katedra.

Kaunas, 2015. 50 psl.

SANTRAUKA

Darbo tikslas: sudaryti mobilaus roboto pasisukimo kampo, nuklydus nuo užduotos kelio trajektorijos, tikslinimo algoritmą bei jį ištirti.

Tiksliai žinoti mašinos poziciją yra viena iš pagrindinių problemų mobiliųjų robotų valdymo programoje. Ieškodami sprendimo, mokslininkai ir inžinieriai sukūrė daugybę sistemų, jutiklių ir metodų mobilaus roboto pozicionavimui. Yra sukurta daug pozicionavimo sistemų kategorijų, kurių keletas yra šios: nuvažiuoto atstumo ir greičio stebėjimas, inercinė navigacija, magnetiniai kompasai, aktyvūs švyturiai, navigacija pasitelkiant orientyrus, modelių atitikimas. Tačiau reliame gyvenime atsiranda įvairių nenumatytų sąlygų ir aplinkybių, kurios gali atsirasti robotui judant, pavyzdžiui ratų prasisukimas ar jutiklių duomenų dreifas. Todėl dažniausiai roboto navigacijai naudojama keletas skirtingų jutiklių, kurie gali kompensuoti vienas kito netikslumus ir, duomenų apdorojimui pritaikius atitinkamus algoritmus, tiksliai nustatyti mobilaus roboto poziciją ir ją koreguoti.

Šiame darbe aprašomi ir nagrinėjami skirtingi mobilaus roboto pozicijos nustatymo metodai. Darbe apžvelgiami skirtingi jutikliai bei aprašomi keletas naudojamų algoritmų mobilaus roboto pozicijos tikslinimui.

Ištyrus skirtingus metodus buvo pasirinktas vienas iš jų, kurio principu sudarytas mobilaus roboto pasisukimo kampo tikslinimo algoritmas. Algoritmas realizuotas „Centaurus CPN“ programiniame pakete. Atliktų eksperimentų ir aprašyto algoritmo programos bandymų analizės duomenys pateikti lentelėse bei grafiniu pavidalu.

Reikšminiai žodžiai: robotas, akcelerometras, algoritmas, koordinatė, ašis.

Čižauskas, Povilas. Development and Investigation of Algorithm for Mobile Robot Path Planning Correction. Final project of *master degree* / supervisor doc. dr. Virginijus Baranauskas; Kaunas University of Technology, Faculty of Electrical and Electronics Engineering, department of automation.

Kaunas, 2015. 50 p.

SUMMARY

The aim of this project: to create and examine mobile robot rotation angle correctionalgorithm used when robot is strayed from given pathway.

To know exactly the position of machines is one of the main problems of mobile robot control program. While searching for a solution, scientists and engineers have developed a variety of systems, sensors and methods for mobile robot positioning. There are a lot of positioning system categories, some of which are as follows: traveled distance and speed monitoring, inertial navigation, magnetic compasses, active beacons, navigation using landmarks, pattern matching. However, in real life different conditions and unforeseen circumstances may occur while moving the robot, such as the over rotation of wheels or sensor data drift. Therefore, in most cases several different sensors are being used for mobile robot navigation that can offset each other's inaccuracies and, by using appropriate algorithms for data processing, precisely adjust mobile robot position.

In this paper different mobile robot positioning methods are being analysed and described. In addition to this, paper renders reviews of different sensors and algorithms which describe mobile robot position correction.

After examination of different methods one of them was chosen for creating mobile robot rotation angle adjustment algorithm. The algorithm was developed by using "Centaurus CPN" programming package. Data from conducted experiments and algorithm tests is being analysed and presented by using tables and graphs.

Keywords: robot, accelerometer, algorithm, coordinate, axis.

TURINYS

Įvadas

Viena iš problemų roboto lokalizacijoje - kaip nustatyti roboto būseną (vietą ir orientaciją) atsižvelgiant į aplinka. Ši problema yra viena iš pagrindinių problemų mobilioje robotikoje. Mobilaus roboto lokalizacijos problema iškyla su daug įvairių niuansų. Pati paprasčiausia lokalizacijos problema, kuri susilaukė daugiausiai dėmesio, yra vietos stebėjimas. Čia pradinė roboto poza yra žinoma, tačiau susiduriama su problema kaip kompensuoti papildomas klaidas roboto nuvažiuotame atstume. Pozicijos sekimo algoritmai dažnai padaro ribojančias prielaidas dėl klaidos dydžio ir roboto netikrumo formos.

Primityviai roboto lokalizacijos problema gali būti apibendrinama trimis klausimais: „kur aš?“, „kur aš einu?“ ir „kaip man ten patekti?“. Pirmis klausimas yra vienas iš lokalizacijos problemų: kaip man išsiaiškinti, kur aš esu esamoje aplinkoje, pasiremiant tuo, ką matau ir kas man ankščiau buvo pasakyta? Antrasis ir trečiasis klausimai yra esminiai nurodant tikslą ir kaip jį pasiekti suplanuojant maršrutą. Pirmiausia yra susitelkiama į pirmą lokalizacijos klausimą, kurį teisingai atsakius būtų padėtas svarbus pagrindas kitiems dviem klausimams įgyvendinti. Ypač svarbu sekti, ar mobilus robotas nenuklydo nuo jam užduotos kelio trajektorijos. Nuklystama dėl technologinių priežasčių (pvz. prasisuko vienas iš roboto ratų) ar piktavalių kėslų (pvz. fiziškai uždengė roboto atstumo jutiklius ar apsuko visą robotą). Siekiant išspresti šią problemą tarpiniuose kelio trajektorijos taškuose būtina atlikti koordinačių tikrinimą. Būtina žinoti, ar mobilus robotas realiai pajudėjo iš savo vietos, ar važiuoja ta kryptimi. Šiai problemai spręsti siūloma naudoti papildomus elektroninius komponentus, fiksuojančius mobilaus roboto posūkio kampus.

1 Literatūros apžvalga

1.1 Roboto pozicijos nustatymas naudojantis realiais jutiklių duomenimis

1.1.1 Pozicionavimo sistemos:

Nuvažiuoto atstumo ir greičio matavimas yra plačiausiai paplitęs navigacijos metodas mobilaus roboto pozicionavime. Jis suteikia gerą trumpalaikį tikslumą, yra nebrangus ir leidžia labai aukštą diskretizavimo dažnį. Tačiau pagrindinė idėja yra per tam tikrą laiką didėjančios judesio informacijos integracija kuri neišvengiamai veda prie duomenų kaupimo klaidų. Tiksliau, orientacijos klaidos sukels didelį pozicijos neatitikimą, kuris proporcingai augs su nuvažiuotu atstumu. Neskaitant šių trūkumų, dauguma mokslininkų sutinka, kad odometrija yra svarbi dalis roboto navigacijos sistemoje, o navigacijos uždaviniai būtų daug paprastesni jeigu išaugtų atstumo ir greičio matavimų tikslumas.

Odometrija remiasi paprastomis lygtimis, kai ratų santykis gali būti tiksliai paverčiamas į linijinį poslinkį pagrindo atžvilgiu. Tačiau, esant ratų buksavimo atvejui ir kitiem trikdžiams, ratų sukimasis negali būti tiksliai išverstas į linijinį judesį. Gautos klaidos gali būti skirstomos į dvi grupes: sistemingos klaidos ir nesistemingos klaidos. Sistemingos klaidos atsiranda dėl kinematinių priežasčių, pavyzdžiui, nevienodo ratų skersmens arba tikslios ratų bazės nežinojimo. Nesistemingos klaidos yra tos, kurios atsiranda salytyje su grindimis, pavyzdžiui, ratų buksavimas ar iškilimų ar įtrūkimų sąveika.

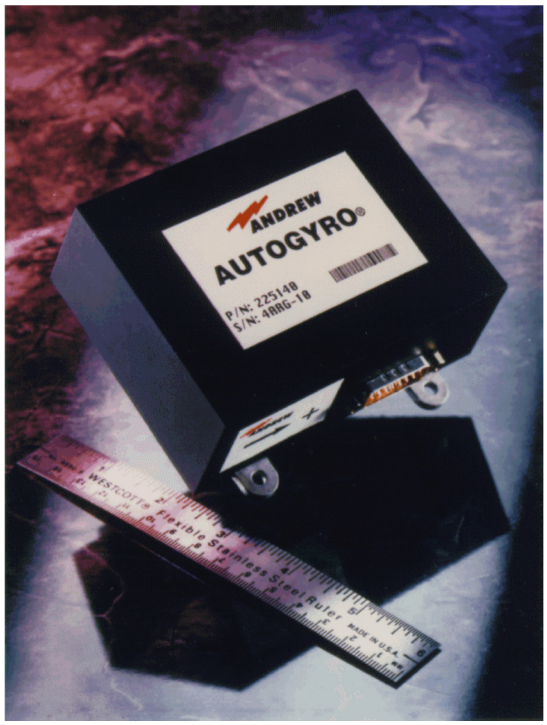
Inercinė navigacija naudoja giroskopus ir akcelerometrus atitinkamai matuoti sukimosi greitį ir pagreitį. Matavimai yra integruojami vieną arba du kartus (akcelerometrui), kad būtų gauta poziciją. Inercinės navigacijos sistemo privalumas yra tas, kad ji yra savarankiška, t.y. jai nereikia išorinių nuorodų. Tačiau inercinio jutiklio duomenys dreifuoja bėgant laikui dėl poreikio integruoti duomenis pozicijos nustatymui. Bet kokia maža pastovi paklaida didėja po integravimo, todėl inerciniai davikliai dažniausiai netinkami tiksliam pozicijos nustatymui per ilgą laiko tarpą.

Akcelerometrų naudojimas. Bandymų rezultatai naudoti akcelerometrus mobilių robotų navigacijai buvo prasti. Buvo atrasta, kad jie turi labai mažą signalas – triukšmas santykį esant mažiems pagreičiams, pavyzdžiui, lėtam pasisukimui. Akcelerometrai taip pat kenčia nuo plataus dreifo ir jie yra jautrūs nelygiam pagrindui, kadangi betkoks nukrypimas nuo horizontalios pozicijos privers jutiklį reaguoti į gravitacijos pagreičio G komponentę. Viena pigi inercinės navigacijos sistema siekia įveikti pastarąją problemą įtraukiant pokrypio jutiklį. Pokrypio jutiklio duomenys buvo naudojami akcelerometrui, kad kompensuoti gravitacijos komponenčių

projekcijas ant kiekvienos akcelerometro ašies. Nepaisant to, duomenys gauti iš pokrypio kompensavimo sistemos rodo pozicijos dreifą nuo 1 iki 8 cm/s, priklausomai nuo pagreičio pasikeitimo dažnio. Tai yra nepriimtinas klaidų lygis daugeliui mobilių robotų sistemų.

Giroskopai yra ypač svarbūs mobilių robotų pozicionavimui, kadangi jie gali padėti kompensuoti svarbiausius silpnumo taškus odometrija paremtuose pozicionavimo sistemose. Bet kokia maža, trumpalaikė orientavimosi klaida įššaukia pastoviai augantį šoninės pozicijos nuokrypį. Dėl šios priežasties būtų labai naudinga, jei orientavimosi klaidos būtų aptiktos ir ištaisytos nedelsiant.

Dar visai neseniai labai tikslūs giroskopai buvo per brangūs mobilių robotų programoms, tačiau visai neseniai atsirado optinio pluošto giroskopai, vadinami lazeriniais giroskopais, kurie yra žinomi kaip labai tikslūs. Jie labai atpigo ir tapo labai patraukliu sprendimu mobilių robotų navigacijoje. Vienas iš komercinių lazerinių giroskopų yra „Autogyro Navigarot“, pagamintas „Andrew Corp.“, pavaizduotas 1.1 paveiksle. Tai vienos ašies interferometrinis šviesolaidinis giroskopas, paremtas poliarizaciją palaikančio pluošto ir precizinio šviesolaidžio giroskopo technologija. Šis giroskopas kainuoja mažiau kaip 1000$ ir yra tinkamas mobilaus roboto navigacijos sistemoms. [1]

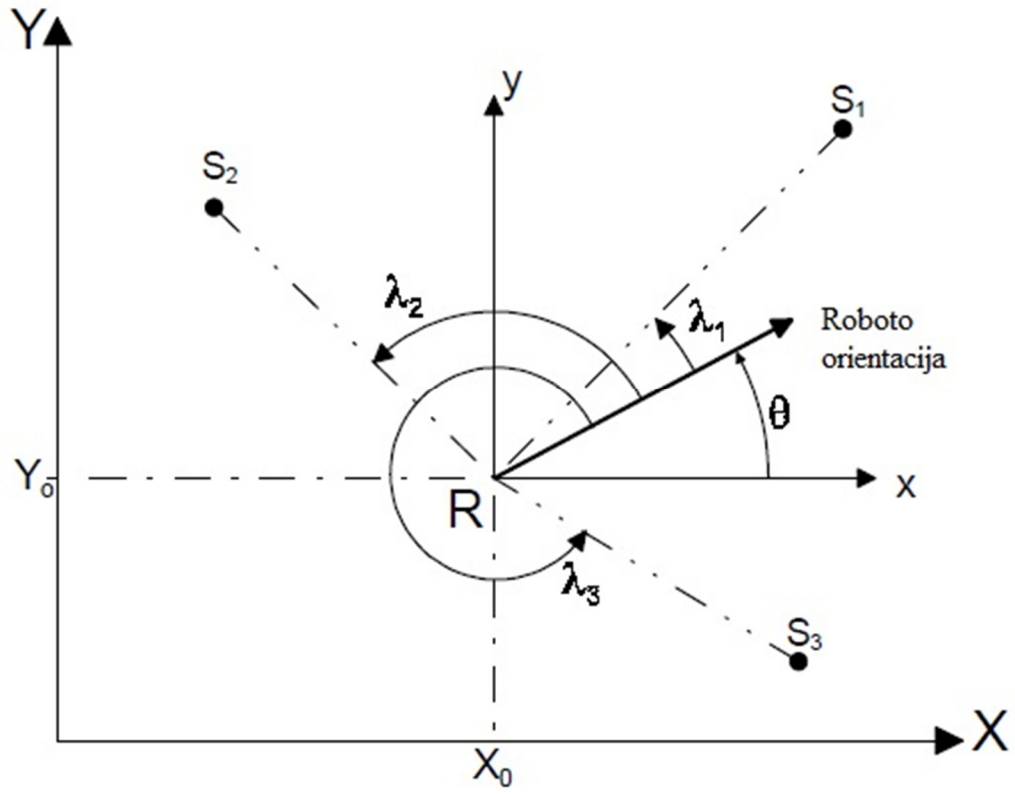


1.1 pav. „Autogyro“ giroskopas [1]

Aktyvių švyturių navigacinė sistema yra dažniausiai naudojama laivų ir lėktuvų, bei komercinių mobilių robotų sistemose. Aktyvūs švyturiai gali būti patikimai aptinkami ir pateikti tikslią pozicionavimo informaciją su minimaliu apdorojimu. Kaip rezultatas, šis metodas leidžia didelį dikretizavimo dažnį ir patikimumą, bet jis taip pat reikalauja didelių išlaidų montavime ir priežiūroje. Tikslus švyturių sumontavimas yra reikalingas tiksliam pozicionavimui. Galima išskirti du skirtingus aktyvių švyturių tipus: trilateriacijos ir trianguliacijos.

Trilateriacija – mašinos pozicijos nustatymas pasiremiant matavimais iki žinomų švyturių šaltinių. Trilateriacijos navigacijos sistemose paprastai yra trys arba daugiau siųstuvų, sumontuotų žinomose aplinkos padėtyse, ir vienas imtuvas, sumontuotas robote. Taip pat gali būti vienas siųstuvas robote ir imtuvai sumontuoti sienose. Naudojant realaus laiko informaciją sistema suskaičiuoja atstumą tarp stacionarių siųstuvų ir imtuvo.

Trianguliacija – yra trys arba daugiau aktyvių siųstuvų, sumontuotų žinomose vietose. Besisukantis jutiklis ant roboto korpuso registruoja kampus λ_1, λ_2 ir λ_3, kuriais jis mato siųstuvo švyturius S_1, S_2, S_3, atsižvelgiant į išilginę roboto ašį. Ši sistema pavazduota 1.2 paveiksle.



1.2 pav. Roboto pozicijos nustatymas trianguliacijos metodu [2]

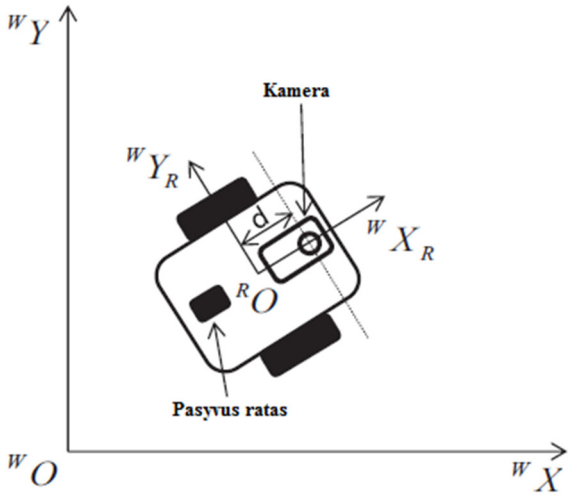
Iš šių trijų matavimų, nežinomos „X“ ir „Y“ kordinatės ir roboto orientacija gali būti apskaičiuojama. Vienintelė problema su šia konfigūracija yra ta, kad norint jog švyturiai būtų matomi iš atstumo didesnio kaip 20 ir daugiau metrų, jie turi būti sutelkti į kūgio formos sklidimo modelį. Kaip rezultatas, švyturiai nėra matomi daugelyje vietų. Problema yra ypač svarbi, kadangi bent trys švyturiai turi būti matomi vienu metu. [2]

Orientyrai yra atskiros funkcijos kurias robotai gali atpažinti iš jo sensorinės įvesties. Orientyrai gali būti geometrinės figūros, pavyzdžiui, stačiakampiai, linijos, apskritimai, ir jie gali talpinti papildomą informaciją, pavyzdžiui, brūkšninius kodus. Apskritai, orientyrai turi fiksuotas ir žinomas pozicijas, atitikmenius, į kurios robotas gali pozicionuoti save. Orientyrai yra kruopščiai atrenkami, kad būtų lengvai atpažystami. Pavyzdžiui, turi būti pakankamas kontrastas palyginti su fonu. Prieš robotui pradedant naudotis orientyrais, orientyrų charakteristikos turi būti žinomos ir išsaugotos roboto atmintyje. [3]

1.1.2 Roboto pozicijos nustatymas remiantis judesio ir vaizdo sensoriais

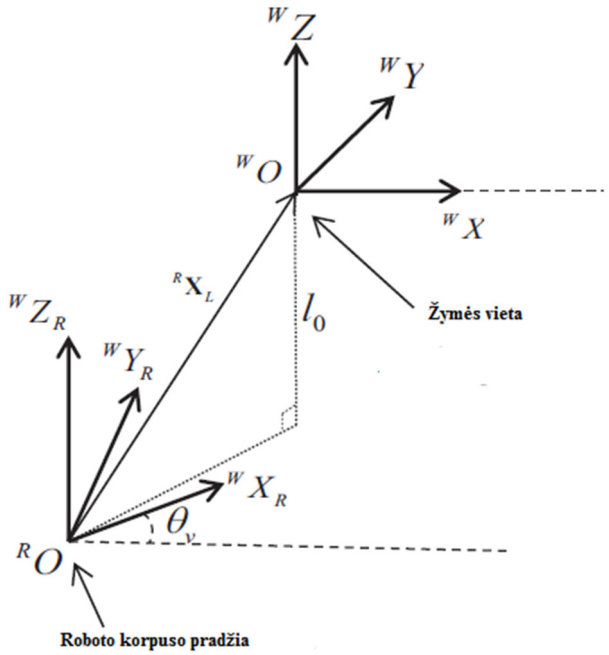
Šiame metode naudojami duomenys gauti iš vaizdo duomenų sintezės ir judesio jutiklių matavimų. Vaizdo jutiklio duomenys gaunami naudojantis viena kamera, o judesio jutiklių duomenys gaunami iš enkoderių, giroskopo ir akcelerometro, esančių mobiliam robote. Iš šios informacijos sukurta sistema paskaičiuoja orientaciją, padėtį ir roboto greitį naudodama išplėstinį Kalmano filtrą (toliau EKF). Norint tiksliai įvertinti roboto poziciją ir greitį, sukurta sistema turi aptikti ratų praslydimą lygindama suskaičiuotus greičius naudojant enkoderius ir akcelerometrą. Siūlomas lokalizacijos metodas įvertina roboto būvį įvairiomis dinaminėmis aplinkybėmis, pavyzdžiui, ratų praslydimo arba roboto pagrobimo atveju.

Norint įvertinti roboto orientaciją ir poziciją naudojantis vaizdo informacija reikalingas mobilaus roboto modeliavimas. 1.3 paveiksle pavaizduotas plokštuminis tipinio ratuoto mobilaus roboto vaizdas. Jis susideda iš transporto priemonės su dviem varančiaisiais ratais, pasyviuoju ratu ir į viršų žiūrinčia kamera.



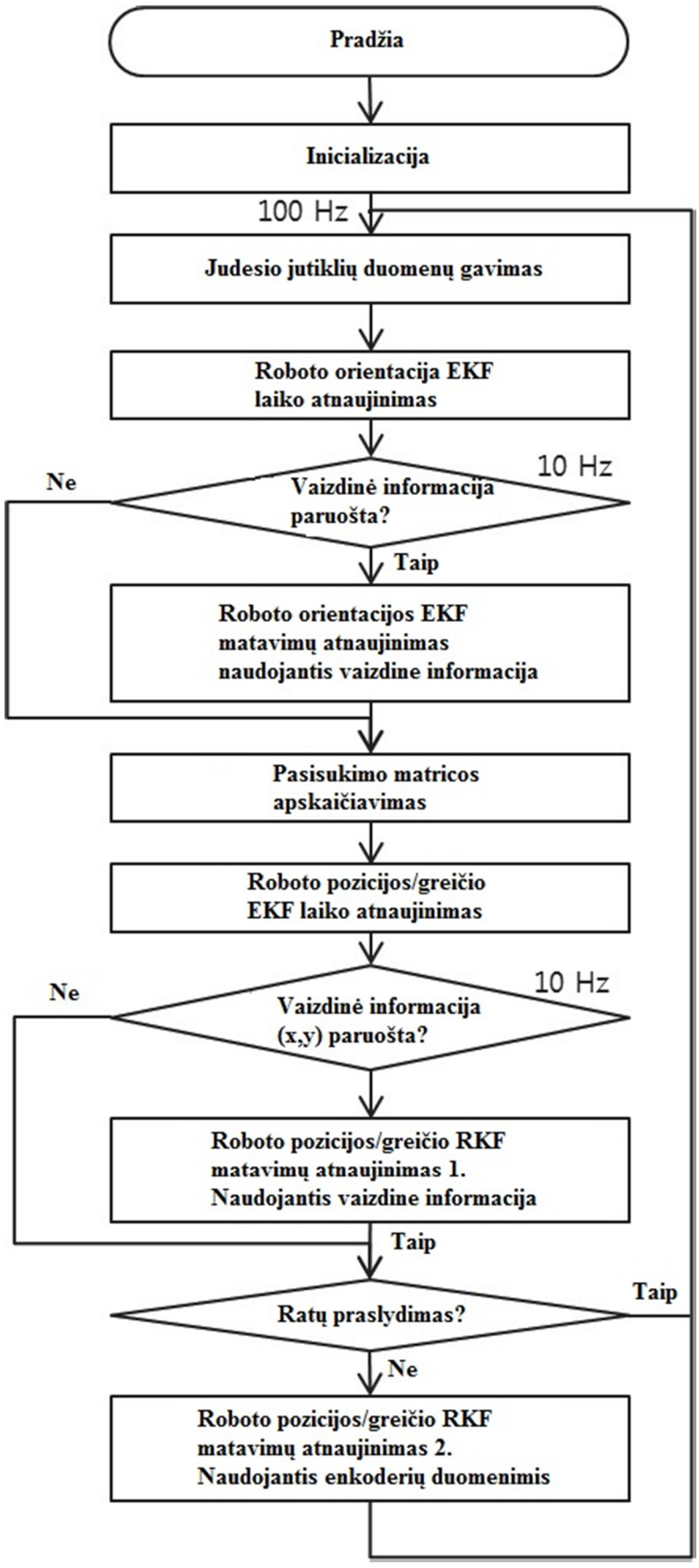
1.3 pav. Plokštuminis mobilaus roboto sistemos vaizdas [4]

Norint apskaičiuoti mobilaus roboto orientaciją ir poziciją naudojantis vaizdo apdorojimo duomenimis naudojama SURF (speeded up robust feature) atpažinimo programa. Programos inicializacijos metu vartotojas gali pasirinkti orientyrų šabloną. Orientyrų vieta ir orientacija nuotraukos plane yra apskaičiuojama kiekviename žingsnyje. 3D trimatis roboto kinematikos ir koordinačių sistemos vaizdas atvaizduojamas 1.4 paveiksle. [4]



1.4 pav. Mobilaus roboto kinematikos ir koordinačių sistemos 3D vaizdas [4]

Šiame metode naudojamas roboto lokalizacijos algoritmas pavaizduotas 1.5 paveiksle.



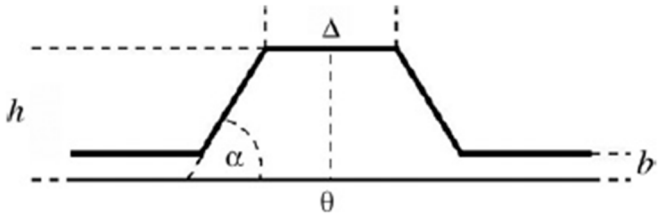
1.5 pav. Mobilaus roboto lokalizacijos algoritmas [4]

1.2 Roboto navigacijos algoritmų realizavimas programiniuose paketuose

Mobilaus roboto lokalizacijai aplinkoje galima pasitelkti ir pritaikyti įvairius lokalizacijos metodus bei jų mišinius. Populiarūs yra „Fuzzy“, „Gauso“, bei „Trilanguliacijos“ metodai, kurie gali būti sujungti su kitais metodais taip gaunant hibridinius lokalizacijos metodus. Toliau šie metodai aprašyti skirtingų mokslininkų kurie savaip pritaikė šiuos metodus.

1.2.1 Fuzzy - Markov logika paremtas lokalizacijos metodas

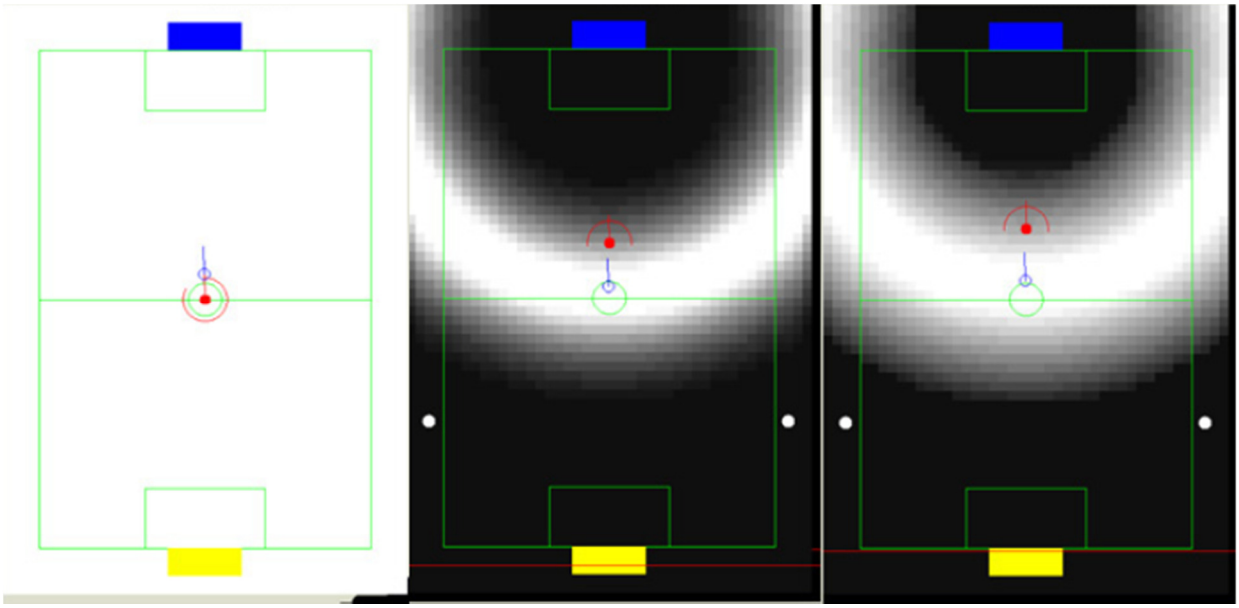
Pagrindinis šio metodo tikslas yra pateikti robotui konkrečius duomenis apie jo poziciją. Šiame metode laukas yra padalintas į tinklą su x ir y kordinatėmis. Kiekviena šio tinklo pozicija yra konfigūruojamos dimensijos celė. Kiekviena celė turi informaciją apie tikimybę, kad robotas yra šioje padėtyje, ir labiausiai tikėtiną roboto orientaciją. Ši informacija yra perduodama difuziniu trapezoidu, 1.6 paveikslas. Jeigu *h* yra žemas lygis, tikimybė, kad robotas bus šioje padėtyje, yra maža. Jei *h* yra aukštas lygis, labai tikėtina, kad robotas yra būtent šioje pozicijoje. Jeigu trapezoidas yra platus (Δ - yra didelė), roboto orientacija nėra tiksliai žinoma, jeigu Δ - mažas arba artimas nuliui, tuomet galime teigti, kad roboto orientacija yra lygi θ.



1.6 pav. Difuzinis trapezoidas [5]

Roboto lokalizacijos procesas naudojamas šiame metode yra pasikartojantis, kiekvienas ciklas turi prognozę ir atsinaujinantį žingsnį. Prognozės žingsnis yra atliekamas naikinant tikimybių tinklelį judesio kryptimi naudojant odometriją. Atsinaujinimo žingsnyje pridedama vaizdinė informacija. Šis metodas remiasi atstumų informacija nuo 6 išdėstytų žymių aikštelėje. Robotas laikomas lokalizuotu, kai kokybė yra aukšta. Kokybė priklauso nuo aikštės dydžio ir skaičiaus celių su didele tikimybe (*h* yra aukštas lygis), tuomet robotas yra sukoncentruotas į sumažintą erdvę. Roboto pozicija yra nustatoma celių su didele tikimybe centre. [5]

Šio metodo taikymo pavyzdys yra pavaizduotas 1.7 paveiksle. Balta spalva žymi celes su didele tikimybe ir juoda spalva žymi celes su maža tikimybe. Raudomas apskritimas žymi roboto poziciją, o raudona rodyklė – orientaciją.



1.7 pav. Roboto lokalizacijos vertinimas pasitelkiant „Fuzzy“ logiką [6]

Robotas laikomas lokalizuotu kai yra kelios celės su didele tikimybe. Kairiausiame paveiksle robotas pradeda judėti kai visos celės turi tokią pačią tikimybę. Kai robotas juda, celės pakeičia spalvą pagal tikimybę, kad jis ten yra. Tikimybė išmatuojama pagal atstumus nuo ženklų. Šio metodo tikslumas pagrinde priklauso nuo tinklelio celių dydžio, kuo didesnė celė, tuo tikslumas mažesnis ir atvirkščiai. Bendrai paėmus galima būtų išskirti šiuos pagrindiniu šio metodo bruožus:

- Greitas atsistatymas po „pagrobimo“;
- Daug greitesnis nei klasikinis „Markov“ metodas;
- Šį metodą sudėtinga derinti;
- Labai jautrus jutiklių paklaidoms, kas padaro šį metodą labai nestabilų triukšmingose aplinkose. [6]

1.2.2 Integruota Fuzzy logika

Kiti mokslininkai tiria išmatuotas mobilaus roboto nuvažiuoto atstumo ir greičio klaidas pasiteldami „Fuzzy“ logiką. Odometrija pateikia duomenis apie roboto esamą poziciją po tam tikros judesio komandos įvykdymo. Čia egzistuoja skaičiai ir faktai, kurie padaro odometriją nepakankamai patikimą. Odometrijos klaidos turi dvi komponentes: sistemines ir nesistemines klaidas. Sisteminės klaidos yra sukeliamos tam tikrų specifinių sistemos parametrų. Kita vertus, kintamieji, kurie gali sukelti roboto nesistemines klaidas, yra šie: roboto greitis (linijinis ir kampinis), paviršiaus tipas ir nuvažiuotas atstumas. Roboto odometrijos klaidos gali būti apibendrinamos taip:

Orientacijos klaidos: kai orientacijos klaidos patenka į sistemą, jos auga nestabdomai ir sukelia šoninio nuokrypio klaidą. Orientacijos klaidos paprastai atsiranda iš roboto slydimo, komandų įvykdymo netikslumo ir sisteminių komponentų.

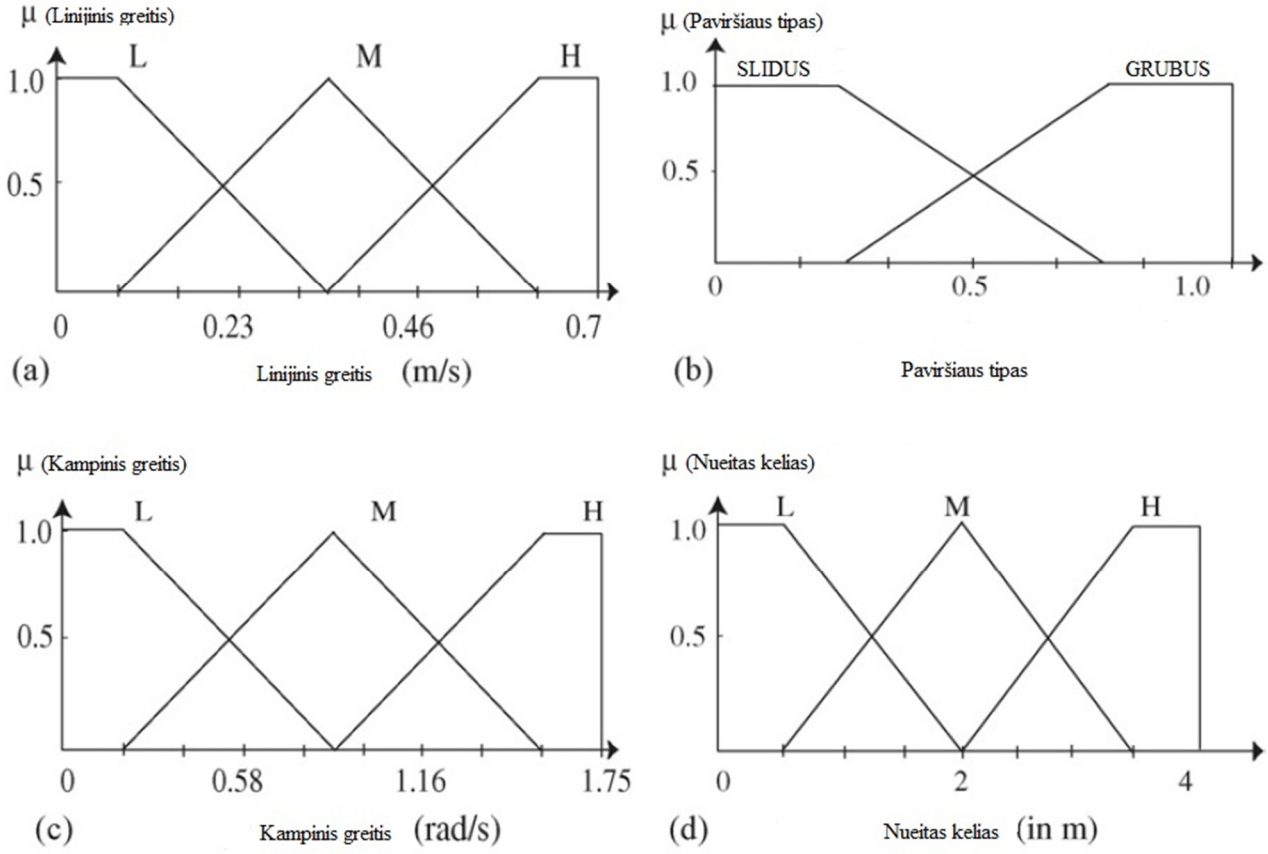
Roboto padėties nuskaitymo klaidos sukelia roboto šoninės padėties klaidas. Taip pat skirtinga aplinka sukelia papildomas klaidas roboto padėties nustatyme.

Realus odometrijos klaidų modelis turi ištirti visus šiuos faktorius siekiant atspindėti sudėtingą mobilaus roboto lokalizaciją.

Fuzzy sistema yra sukurta pateikti ekspertų žinias apie mobilaus roboto nuvažiuotą atstumą ir greitį priimtina forma. Kitaip tariant, siūloma Fuzzy sistema, pavadinta Fuzzy spėjimu, pateikia lingvistinius kintamuosius apie mobilaus roboto nueito kelio ir greičio klaidas kaip priešpriešą tradiciniam klaidų modeliui. Siūlomas Fuzzy spėjimas turi keturis lingvistinius įėjimo kintamuosius: roboto linijinis greitis (v_t), roboto kampinis greitis (ω_t), paviršiaus tipas (F_t), nueitas kelias (d_t), ir penkis išėjimo lingvistinius kintamuosius: slydimas, vertimo klaida, pasisukimo klaida, roboto specifinė klaida orientacijoje ir roboto specifinė klaida erdvinėje padėtyje. Fuzzy logikos jeigu (IF) – tada (THEN) taisyklės atrodo taip:

- Jeigu F_t yra C_k ir d_t yra D_l ir v_t yra A_i tada poslinkis yra E_ω;
- Jeigu F_t yra C_k ir v_t yra A_i ir ω_t yra B_j tada pasisukimo klaida yra G_z;
- Jeigu F_t yra C_k ir d_t yra D_l ir v_t yra A_i tada vertimo klaida yra F_v;
- Jeigu d_t yra D_l tada roboto specifinė orientacijos klaida yra H_{1q};
- Jeigu d_t yra D_l tada roboto specifinė erdvinės padėties klaida yra H_{2q}.

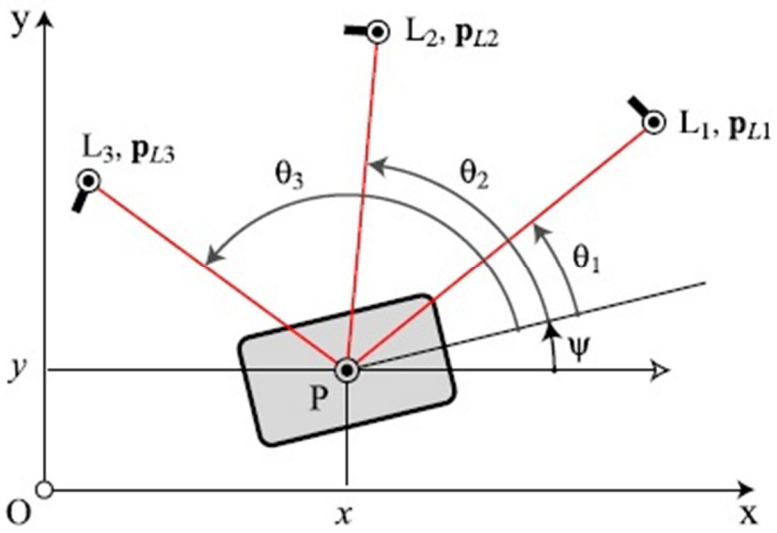
Jeigu norima robotą pritaikyti važiuoti ant skirtingų paviršių, naujos Fuzzy logikos sąlygos gali būti įterptos, taip pat reikalingas pagrindą atpažįstantis jutiklis. Fuzzy logikos motininės įėjimo kintamųjų funkcijos pavaizduotos 1.8 paveiksle. [7]



1.8 pav. Motininės Fuzzy logikos įėjimo funkcijos: (a) – linijinis greitis, (b) – paviršiaus tipas, (c) – kampinis greitis, (d) – nueitas kelias [7]

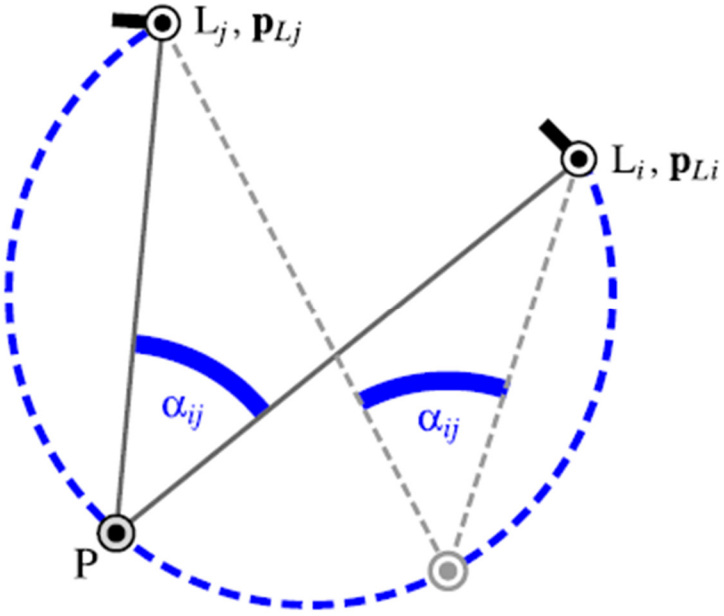
1.2.3 Aktyvių švyturių, trianguliacijos metodas

Šis metodas naudojamas nustatyti mobilaus roboto padėtį ir orientaciją aplinkoje, atliekant kampinius matavimus. Šie matavimai atliekami matuojant kampus tarp roboto centro linijos ir bent trijų švyturių, esančių žinomoje pozicijoje. Trianguliacijos problema gali būti apibūdinama taip: Duotos trys žymių pozicijos P_{Li} (i = 1, 2, 3), kurios nusako vektorių nuo duotos absoliutinės atraminės pozicijos L_i. Problema atsiranda nustatant poziciją $p = \{x,y\}^T$ nuo roboto pozicijos P ir orientacijos kampo ψ iš išmatuotų kampų Θ_1, Θ_2, Θ_3. Šie kampai yra tarp tiesios linijos, einančios per P ir žymes. Objektų išsidėstymas erdvėje pavaizduotas 1.9 paveiksle.



1.9 pav. Objektų išsidėstymas erdvėje [8]

Šiuo metu yra keletas labiau paplitusių algoritmų spręsti trianguliacijos uždavinius. Vienas iš labiausiai naudojamų yra apskritimo arkos algoritmas. Pagal 1.10 paveiksle pateiktą pavyzdį matyti, kad šis metodas yra paremtas tuo, jog kampas $\alpha_{ij} = \Theta_j - \Theta_i$ ($i, j = 1, 2, 3, i \neq j$) tarp linijų, jungiančių P ir dvi žymes L_i ir L_j, suformuoja roboto pozicijos vektorių p, kuris yra ant apskritimo arkos.



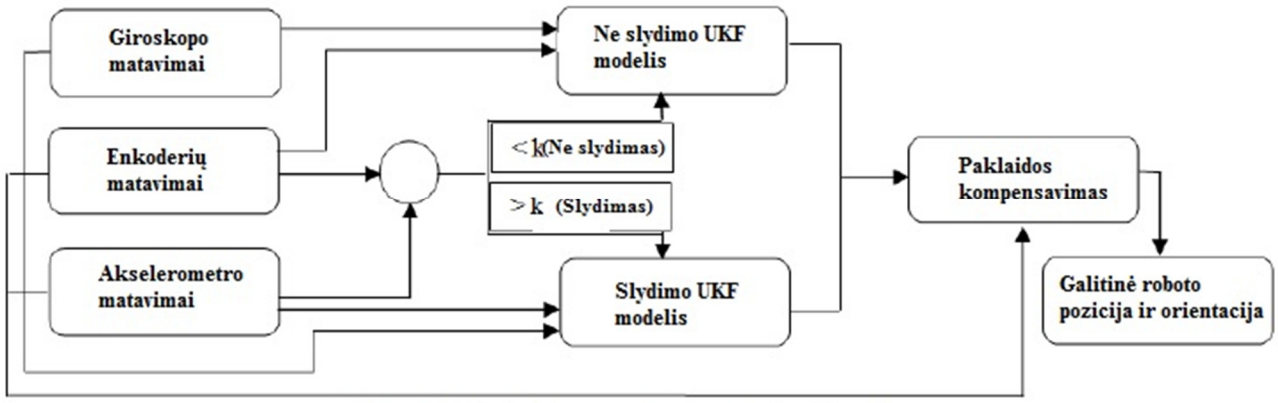
1.10 pav. Trianguliacijos apskritimo arka [8]

Trianguliacijos problemos ypatumai atsiranda kai roboto taškas P atsiduria perimetre, apibrėžtame trijų žymių, šiuo atveju abu perimetrai yra vienodi ir jų susikirtimas negali būti nustatytas. Tai yra svarbu paminėti, kadangi tai - būdingas trianguliacijos ypatumas, kuris negali būti išvengiamas panaudojant vieną iš esamų grafinių arba analitinių metodų. Be to, kai roboto

taškas P yra arti kritinio perimetro, bet koks mažas nuokrypis kampų matavime iššaukia svarbią klaidą p matavime. [8]

1.2.4 Roboto pozicijos nustatymas remiantis nejautriu (unscented) Kalmano filtru

Mobilaus roboto pozicijos nustatymui naudojami duomenys gauti iš akcelerometro, giroskopo ir enkoderių. Nejautrus Kalmano filtras (toliau UKF) naudojamas kaip veiksmingas jutiklių sintezės algoritmas, kuris pasižymi patobulinta filtravimo technika, kuri pralenkia plačiai naudojamą praplėstą kalmano filtrą (Extended Kalman Filter) daugelyje programų. Sistema yra pajėgi kompensuoti praslydimo klaidas, persijungiant tarp dviejų skirtingų UKF modelių, sukurtų slydimo ir ne slydimo atvejams. Laikui bėgant atsiranda akcelerometro paklaida dėl dvigubos integracijos, todėl akcelerometro duomenys šiuo metodu naudojami tik UKF slydimo modelyje. Rezultatai gauti iš UKF sensorių sintezės yra lyginami su rezultatais gautais iš tikslių lazerinių atstumo jutiklių. Apžvelgti eksperimentiniai rezultatai rodo, kad tokia sistema yra pajėgi tiksliai sekti roboto judėjimo trajektoriją įvairiais roboto judesių scenarijais, netgi ir tokiais, kai roboto enkoderių duomenys nėra patikimi dėl įvykusio ratų praslydimo. 1.11 paveiksle pavaizduota šiame metode naudojama blokinė jutiklių duomenų sintezės diagrama.



1.11 pav. Jutiklių sintezės blokinė diagrama [9]

Du skirtingi UKF modeliai pavadinti „Slydimo UKF modelis“ ir „Neslydimo UKF modelis“, naudojami atsiradus slydimo sąlygoms. Duomenys iš akcelerometro ir enkoderių pirmiausia yra lyginami tarpusavyje, kad nustatyti slydimo atsiradimą. Remiantis šio detektoriaus išėjimu, bet kuris iš dviejų UKF modelių yra naudojamas. Slydimo UKF modelis integruoja tik inercinių jutiklių duomenimis, o neslydimo UKF modelis integruoja duomenis iš enkoderių ir giroskopo.

Roboto pozicija ir orientacija nustatoma atlikus matavimus, iš kurių pasitelkiant sekančias formules galima gauti roboto poziciją ir orientaciją:

$$p_k = \begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} = \begin{bmatrix} x_{k-1} + v_{k-1} \cdot \Delta t \cdot \cos(\theta_{k-1}) \\ y_{k-1} + v_{k-1} \cdot \Delta t \cdot \sin(\theta_{k-1}) \\ \theta_{k-1} + w_{k-1} \cdot \Delta t \end{bmatrix}$$

$$v = \frac{v_r + v_l}{2} \qquad w = \frac{v_r - v_l}{d}$$

Apžvelgus eksperimentinius duomenis, kurių rezultatai pateikti 1.1 lentelėje, matome, kad naudojant aprašytą UKF sistemą, paklaidos galutiniame trajektorijos taške yra ženkliai mažesnės negu remiantis akcelerometrų arba giroskopo duomenimis.

5.1 lentelė. Akcelerometro išėjimo vertės esant skirtingiems posūkio kampams [9]

Eksperimentas	Sensoriai	Galutinės pozicijos/kampo paklaida
Tiesinis judesys	Enkoderiai	100 mm
	UKF sistema	5 mm
Tiesinis judesys + pasisukimas	Enkoderiai	2.33°
	Giroskopas	8.2°
	UKF sistema	-1.2°
Praslydimas (tiesinis judesys)	Enkoderiai	238 mm
	UKF sistema	96 mm

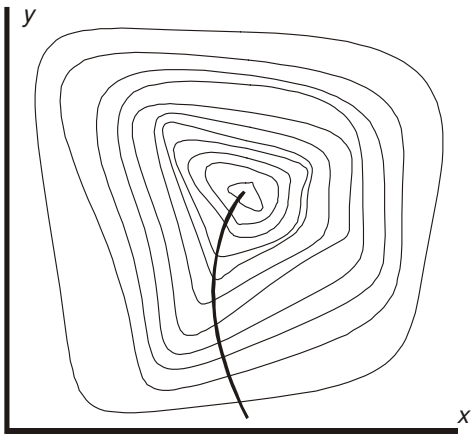
Šie eksperimentiniai duomenys gauti juos lyginant su atstumu, išmatuotu tiksliu lazeriniu jutikliu. Šie rezultatai parodo, kad sistema yra pajėgi sekti roboto judesį įvairiuose roboto judesio scenarijuose, tokiuose kaip ratų prasisukimo, kai duomenys gauti iš enkoderių yra nepatikimi ir klaidinantys. [9]

1.2.5 Roboto orientacijos paieška naudojant greičiausio nusileidimo metodus

Dauguma daugiamatės optimizacijos metodų vienaip ar kitaip naudoja informaciją apie gradientą. Gradiento fizikinę esmę geriausiai galime įsivaizduoti trimatėje erdvėje. Jei paimti bet kurį tašką šiame paviršiuje, tai pati didžiausia įkalnė iš šio taško, atitiks gradiento kryptį, ir priešingai, pačio didžiausio nuolydžio kryptis, bus priešinga gradiento krypčiai. Paviršiaus konfigūracija priklauso nuo ją aprašančių lygčių ar kitokios informacijos, ir gradiento kryptis ir dydis skirtingiems paviršiaus taškams yra skirtingi. Galima pateikti labai paprastą pavyzdį, sakykime, kad mes plona srovele pilame vandenį konkrečiame paviršiaus taške. Vanduo visada tekės kryptimi priešinga didžiausiam gradientui, t.y. ta kryptimi kur paviršius yra stačiusias, o

tekėjimo greitis priklausys nuo gradiento dydžio: kuo gradientas bus didesnis, tuo ir vanduo tekės greičiau.

1.12 paveiksle pateikta greičiausio nusileidimo linija, tai linija kiekviename paviršiaus taške statmena lygių kriterijų reikšmes jungiančiai linijai. Ši linija sutampa su gradiento kryptimi tik greičiausias nusileidimas vyksta priešinga kryptimi gradiento vektoriui.



1.12 pav. Greičiausio nusileidimo linija sutampanti su gradientu [10]

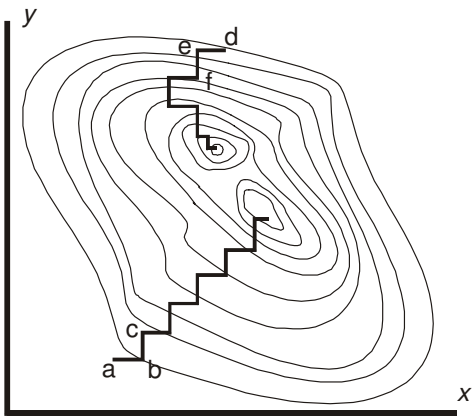
Dauguma metodų, kurių veikimas bazuojasi gradiento pagrindu, yra vadinami greičiausio nusileidimo metodais. Kadangi skaičiavimai vykdomi ne išvestinėmis, o pokyčiais, gradientas yra paskaičiuojamas tik artimas realiajam, tad yra įvairūs skaičiavimo algoritmai ir tuo pagrindu sukurta eilė skaičiavimo metodų: [10]

Rosenbroko metodas turi tą ypatumą, kad skirtingai gradientiniam metodui, judėjimas vyksta ne tiksliai pagal gradientą, o kiekvienos koordinatės kryptimi eilės tvarka, taip organizuojant skaičiavimo algoritmą, kad atliktas žingsnis visuomet būtų sėkmingas. Kitaip sakant, pradiniam taškui paskaičiuojama funkcionalo reikšmė, o sekanti skaičiavimo seka būtų tokia:

- Atliekamas žingsnis pirmąja koordinate ir paskaičiuojama funkcionalo reikšmė. Jei ši reikšmė yra geresnė už buvusią, fiksuojama kryptis kaip sėkminga ir numatoma, kad sekantis žingsnis ta koordinatės kryptimi bus didesnis. Jei reikšmė buvo blogesnė, kryptis keičiama priešinga. Jei ir tai neduoda rezultato, žingsnis yra mažinamas, jei reikia - keičiamos kryptys ir žingsnis dar mažinamas tol, kol bus nustatyta, kad funkcionalo paskutinioji reikšmė bus geresnė už reikšmę prieš pradedant paiešką šiai koordinatei.
- Atliekamas žingsnis antrąja koordinate prisilaikant tų pačių reikalavimų, kaip kad ir pirmajai koordinatei.

- Atliekami žingsniai sekančiomis koordinatėmis, užbaigiant paskutiniąja.
- Sekantis paieškos ratas vėl pradedamas eilės tvarka nuo pirmosios koordinatės, ir t.t. iki pasiekiamas norimas optimizavimo tikslumas.

Paieškos algoritmo funkcionavimas pateiktas 1.13 pav. Paieška pradedama iš taško *a* ar taško *d*. Atliekamas žingsnis funkcionalo mažėjimo kryptimi pagal ašį *x*. Sekančiu ėjimu ta pati operacija vykdoma pagal *y* koordinatę. Po to žingsnis vėl atliekamas koordinate *x*, o toliau *y* koordinatei ir t.t iki bus pasiektas optimumas. Paveiksle aiškiai matome, kad, priklausomai nuo paieškos pradžios taško, priklauso kuris paviršiaus ekstremumas bus pasiektas. Tokiu būdu šis metodas negarantuoja, kad paieškos rezultate bus rastas globalinis optimumas. [11]



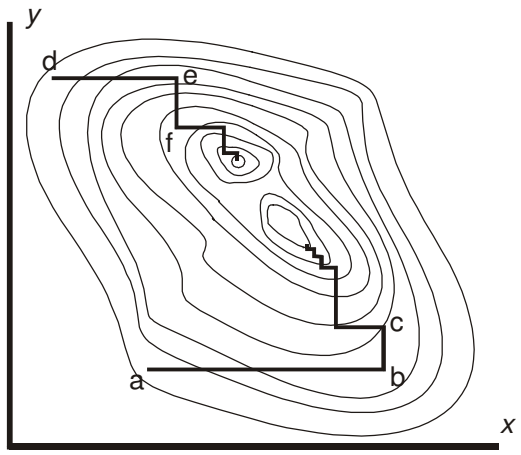
1.13 pav. Funkcionalo minimumo paieška Rosenbroko metodu [11]

Gauso-Zaidelio metodas turi tą ypatumą, kad skirtingai Rosenbroko metodui, judėjimas vyksta kiekvienos koordinatės kryptimi eilės tvarka, taip organizuojant skaičiavimo algoritmą, kad pagal tą koordinatę būtų pasiektas funkcionalo minimumas. Kitaip sakant, pradiniam taškui paskaičiuojama funkcionalo reikšmė, o sekanti skaičiavimo seka būtų tokia:

- Atliekamas žingsnis pirmąja koordinate, ir paskaičiuojama funkcionalo reikšmė. Jei ši reikšmė yra geresnė už buvusią, atliekamas sekantis žingsnis ta pačia kryptimi ir t.t iki pasiekiamas funkcionalo minimumas. Jei reikšmė buvo blogesnė, kryptis keičiama priešinga ir funkcionalo minimumas ieškomas ta kryptimi.
- Atliekamas žingsnis antrąja koordinate, ir paskaičiuojama funkcionalo reikšmė. Jei ši reikšmė yra geresnė už buvusią, atliekamas sekantis žingsnis ta pačia kryptimi ir t.t iki pasiekiamas funkcionalo minimumas. Jei reikšmė buvo blogesnė, kryptis keičiama priešinga ir funkcionalo minimumas ieškomas ta kryptimi.
- Atliekami žingsniai sekančiomis koordinatėmis, užbaigiant paskutiniąja.

- Sekantis paieškos ratas vėl pradedamas eilės tvarka nuo pirmosios koordinatės ir t.t. iki pasiekiamas norimas optimizavimo tikslumas.

Paieškos algoritmo funkcionavimas pateiktas 1.14 paveiksle. Metodo funkcionavimas turi tam tikro panašumo su Rosenbroko metodu. Šis metodas pradėjęs paiešką nuo tašo *a* ar *d* pagal *x* koordinatę, vykdo ją iki tol, kol randa funkcionalo minimumą šios koordinatės kryptimi (atitinkamai *b* ar *e* tašką). Toliau nuo taško *b* ar *e* vykdoma paieška pagal *y* koordinate iki tol, kol randa funkcionalo minimumą šios koordinatės kryptimi (atitinkamai *c* ar *f* tašką) ir t.t. Užbaigus paskutiniąją koordinate, paieška toliau pradedama nuo pirmosios koordinatės (taškai *c* ar *f*). [11]



1.14 pav. Funkcionalo minimumo paieška Gauso-Zaidelio metodu [11]

2 Tyrimų dalis

2.1 Roboto Khepera II įranga

Roboto duomenų perdavimui panaudotas „Zigbee“ bevielio ryšio modulis, tai Institute of Electrical and Electronic Engineer (toliau IEEE) 802.15.4 standartu pagrįstas bevielio duomenų perdavimo protokolas, skirtas bevielių jutiklių tinklui valdyti. Šis strandartas priklauso fiziniam lygiui ir vidutiniam informacijos valdymo sublygiui su mažais greičiais bevieliuose tinkluose, kurie palaiko paprastus prietaisus išeikvojant minimalias energija sąnaudass ir tipiškai atlieka operacijas iki 50 metrų spinduliu. Pastarajam tinklui pasiekti šiame standarte naudojamas ZigBee bevielis tinklas. Naujas ZigBee standartas – vienintelė standartinė bevielė tinklo technologija, realizuojanti tokius reikalavimus kaip:

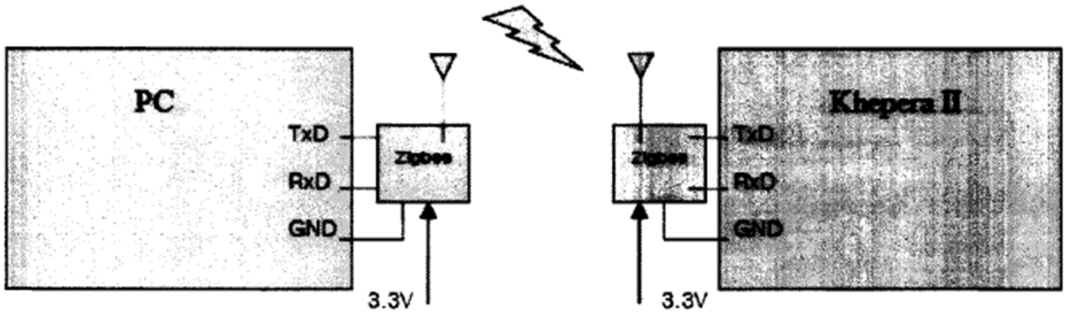
- *Patikimumas* (CRC klaidų kontrolė);
- *Mažos energijos sąnaudos* (Naudojama energijos taupymo sistema, kuri ZigBee tinklo įrenginius perveda į budintį režimą, kurio metu naudojama mažai energijos. Atsiradus poreikiui naudotis šiuo įrenginiu, jis „pažadinamas“ per 15 ms);
- *Maža kaina* (kaina apie 1 JAV dolerį);
- *Paprasta priežiūra, administravimas bei veiksmų stebėjimas;*
- *Integracija su kitais standartais* (palaikomas glaudus ryšys su IEEE standartais, kad būtų užtikrinta šio naujo standarto pritaikomumo galimybė);
- *Saugumas* (128bitų asinchroninis kodas)

Šis standartas veikia 2.4 GHz radio dažniu, tačiau pasižymi sąlyginai maža perdavimo sparta, nuo 20 iki 250 Kbps. Šį ZigBee tinklo trūkumą kompensuoja mažos energijos sąnaudos, kas užtikrina nepertraukiamą ir taupų sistemos darbą gana ilgą laiką bei taupo sistemos administravimui skirtą laiką ir pinigus. [12]

2.1 lentelė. ZigBee modulio techniniai parametrai

Duomenų perdavimo sparta	Iki 250kbps
Sąsaja	Naudojant UART
Signalizavimo sparta	Nuo 9600bps iki 115200bps
Maitinimo įtampa	Nuo 2.7V iki 3.6V
Matmenys	26.5 x 19 x 12mm

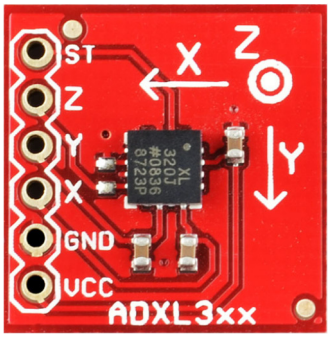
Duomenų perdavimas tarp kompiuterio ir Khepera II roboto vykdomas pasitelkus suporuotus ZigBee modulius. Šiam tikslui būtina užtikrinti sąsajas: ZigBee – kompiuteris ir ZigBee – Khepera II robotas. 2.1 paveikslas.



2.1 pav. Duomenų perdavimas tarp kompiuterio ir Khepera II roboto [12]

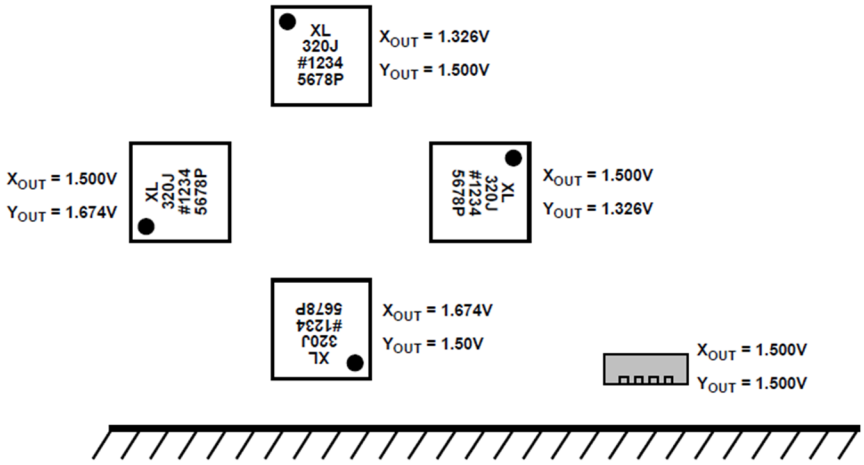
Mobilaus roboto nuvažiuotam keliui ir posūkio kampui fiksuoti naudojami roboto ratų enkoderių bei akcelerometro duomenys. Šie duomenys apdorojami programineme pakete Centaurus CPN, o perduodami bevieliu ryšiu naudojant suporuotus „ZigBee“ modulius.

Naudojamas firmos „Sparkfun“ ADXL320 akcelerometras. Šis modulis maitinamas 2.4-5.25V įtampa, matuojami „X“ bei „Y“ ašių pokyčiai. Duomenys, naudojant analoginius ryšio kanalus, perduodami į duomenų apdorojimui naudojamą programinį paketą. 2.2 paveikslas.



2.2 pav. ADXL320 akcelerometras.

Priklausomai nuo akcelerometro pasisukimo „X“ bei „Y“ ašimis kampo, akcelerometro išėjime formuojamos skirtingos įtampų vertės. Kaip matoma iš gamintojo pateiktos specifikacijos, akcelerometrui nejudant X_{OUT} bei Y_{OUT} gaunama 1.5V įtampa. 2.3 paveikslas.

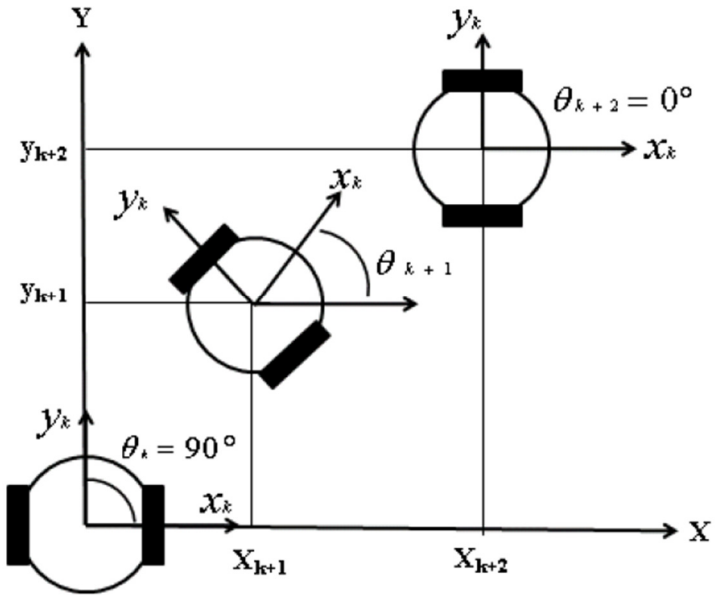


2.3 pav. Akcelerometro išėjimo duomenys priklausomai nuo pasisukimo kampo.

2.2 lentelė. Akcelerometro techniniai parametrai

Modelis	ADXL320
Išmatavimai	18 x 18mm
Maitinimo įtampa	2,4V – 5,25V DC
Matavimo ribos	±5g
Ašių skaičius	2 (X ir Y)
Darbinė temperatūra	−55°C iki +125°C
Netiesiškumas	±2%

Roboto nuvažiuotą kelią ir pasisukimo kampą galima apskaičiuoti remiantis enkoderių duomenimis. Tačiau duomenys gali būti netikslūs jeigu įvyko ratų praslydimas. Todėl tinkamam padėties nustatymui naudojami ir akcelerometro duomenys, pagal kuriuos nustatoma roboto judėjimo kryptis ir tinkamai įvertintinamas roboto pasisukimo kampas. 2.4 paveikslas. [13]



2.4 pav. Roboto pozicijos ir orientacijos nustatymas naudojant enkoderių duomenis [13]

2.2 Maršruto tikslinimo algoritmo sudarymas ir realizavimas

Daugelis mobilaus roboto maršruto tikslinimo ir padėties nustatymo metodų remiasi ne tik enkoderių, akcelerometrų ar giroskopų duomenimis, bet ir papildomomis vaizdo apdorojimo priemonėmis bei kitais jutikliais. Šiame darbe apžvelgiami keletas roboto pozicijos nustatymo metodų, kurie pasitelkia papildomas priemonęs pozicijos nustatymui. Remiantis šiais pavyzdžiais sudaromas algoritmas, kuris nustatys roboto pasisukimo poziciją naudodamas akcelerometro duomenis.

2.2.1 Eksperimentiniai duomenys

Buvo atlikta keletas skirtingų eksperimentų, kurių metu buvo fiksuojami akcelerometro ašių duomenys. Pirmu eksperimentu robotas buvo sukamas nuo 0° iki 360° laipsnių, visą ratą padalinus į dalis po 45° ir kiekvienoje padėtyje nuskaitant „X“ ir „Y“ koordinates. Iš viso buvo atlikti 5 tokie bandymai, o gauti rezultatai kaip duomenys buvo suvesti į 2.3 lentelę. Taip pat buvo apskaičiuoti duomenų vidurkiai, kurie bus naudojami kaip etaloniniai duomenys vėlesniems bandymams.

2.3 lentelė. Eksperimentiniai duomenys

Ašis / Kampas	Sukta ranka										vidurkis	
	x	y	x	y	x	y	x	y	x	y	x	y
0°	852	772	860	776	852	780	864	780	860	784	858	778
45°	836	832	836	828	832	832	836	836	828	834	834	832
90°	772	848	784	840	776	844	768	846	780	842	776	844
135°	728	828	716	824	720	820	724	822	722	826	722	824
180°	700	768	696	772	704	770	708	764	702	770	702	769
225°	724	720	728	708	720	716	726	712	722	714	724	714
270°	772	688	776	700	784	696	780	692	778	688	778	693
315°	836	716	832	720	828	708	824	724	830	722	830	718
360°	852	772	854	768	856	780	852	784	858	776	854	776

2.4 lentelėje suvesti duomenys iš sekančio eksperimento, kuris buvo atliktas tokiu pačiu principu, tik robotas buvo ne pastatomas tiksliu kampu, bet sukosi pats, pagal užduotą komandą, į kiekvieną padėtį, pradedant nuo 0°.

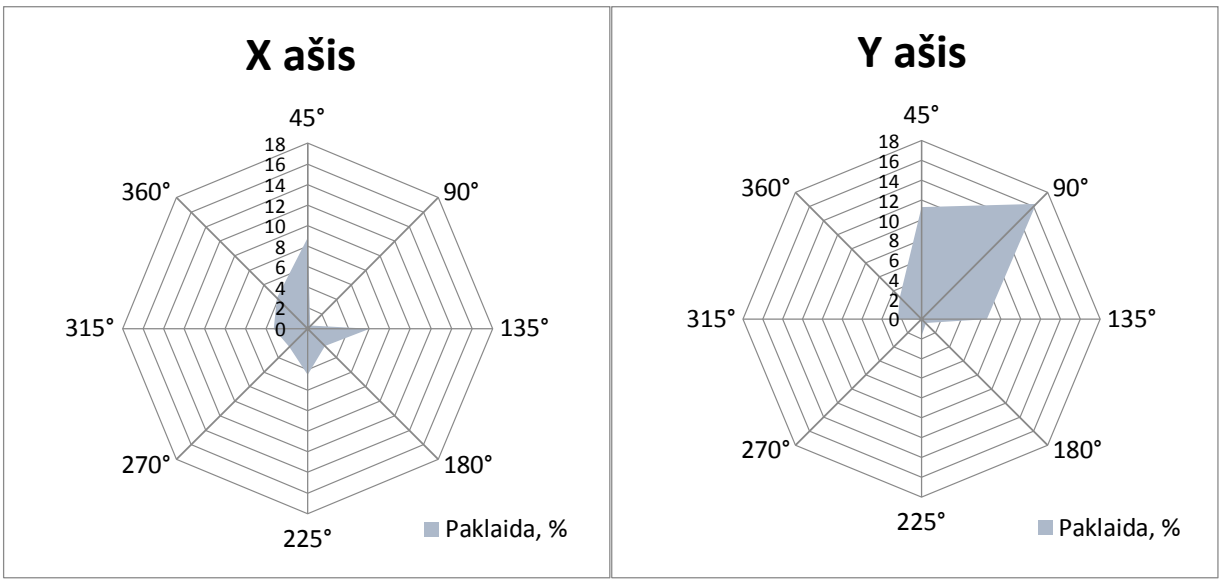
2.4 lentelė. Eksperimentiniai duomenys

Ašis / Kampas	Sukosi pats										Vidurkis	
	x	y	x	y	x	y	x	y	x	y	x	Y
0°	852	772	860	776	852	780	864	780	860	784	858	778
45°	764	744	760	760	756	728	752	740	768	720	760	738
90°	788	688	772	676	756	680	768	700	780	786	773	706
135°	680	768	676	764	688	760	664	780	684	778	678	770
180°	692	772	676	756	684	760	688	764	686	768	685	764
225°	680	708	692	704	696	712	704	688	684	696	691	702
270°	760	692	764	688	768	700	756	676	744	704	758	692
315°	800	704	812	712	804	708	796	684	802	696	803	701
360°	816	752	822	764	814	748	818	750	824	746	819	752

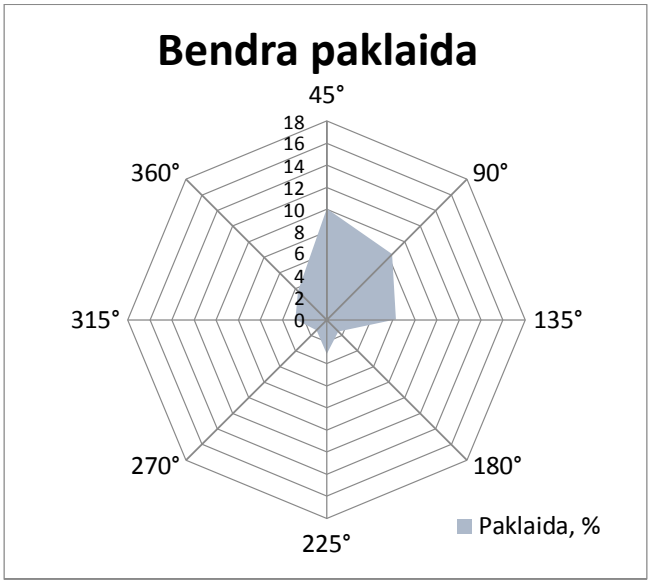
Šie duomenys buvo palyginti su gautais roboto pasisukimo kampą keičiant ranka ir apskaičiuoti nuokrypiai procentais nuo etaloninės vertės „X“ ir „Y“ ašimis. Taip pat buvo paskaičiuotas bendras „X“ ir „Y“ ašių nuokrypio vidurkis, 2.5 lentelė. Gauti duomenys grafiškai atvaizduoti 2.5 – 2.6 paveiksluose, kuriuose matyti „X“, „Y“ ir bendra paklaida procentais prie skirtingų roboto pasisukimo kampų.

2.5 lentelė. Procentinė „X“ ir „Y“ ašių paklaida

Ašis / Kampas	Paklaida nuo vidurkio, %		
	x	y	Bendra
0°	0.0	0.0	0.0
45°	8.8	11.3	10.1
90°	0.4	16.4	8.4
135°	6.0	6.6	6.3
180°	2.4	0.6	1.5
225°	4.5	1.7	3.1
270°	2.5	0.1	1.3
315°	3.3	2.4	2.8
360°	4.2	3.1	3.6



2.5 pav. Procentinė „X“ ir „Y“ ašių paklaida atvaizduota grafiškai

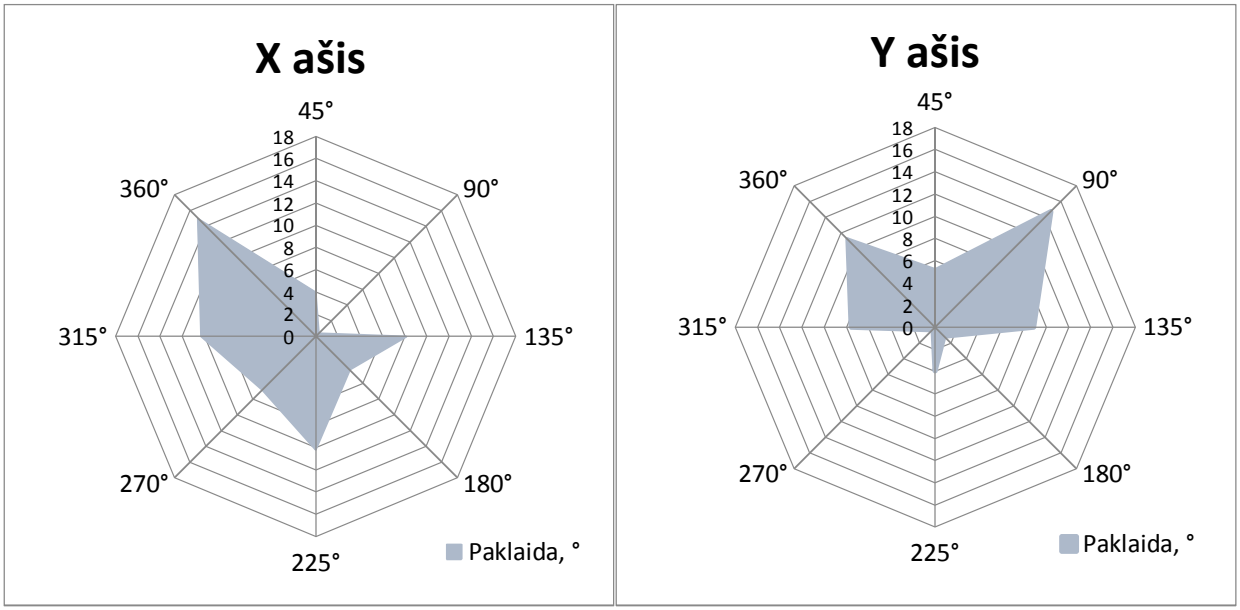


2.6 pav. Bendra procentinė „X“ ir „Y“ ašių paklaida atvaizduota grafiškai

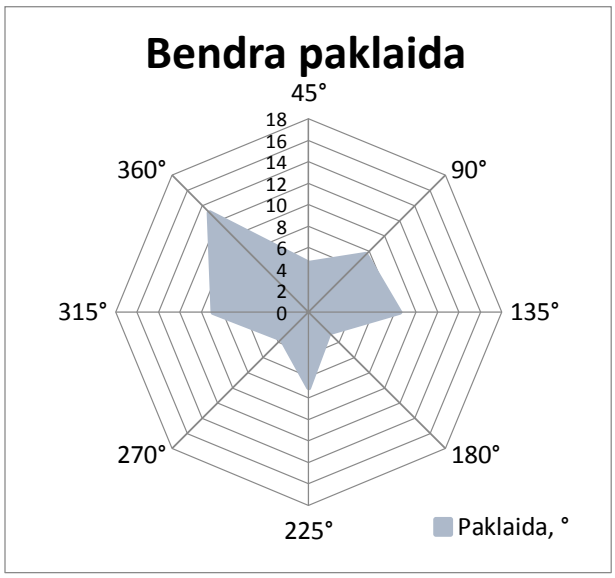
Gauti duomenys procentais buvo perskaičiuoti į laipsnius, kadangi pateikus duomenis procentais jie neatspindi realios paklaidos, nes didesnis procentinis nuokrypis nebūtinai reiškia, kad robotas nukrypo didesniu kampu, negu su mažesniu procentiniu nuokrypiu, 2.6 lentelė. Taip pat jie buvo atvaizduoti ir grafiškai. 2.7 – 2.8 paveikslai. Iš šių duomenų matyti, kad roboto nukrypimui nuo užduoto pasisukimo neturi įtakos kokiu kampu jam buvo liepta pasisukti. Tai priklauso tik nuo roboto aparatinės įrangos ir jutiklių tikslumo, kadangi pasisukimo kampo paklaida nėra proporcinga užduotam pasisukti kampui.

2.6 lentelė. „X“ ir „Y“ ašių paklaida išreikšta laipsniais

Ašis / Kampas	Paklaida nuo vidurkio, °		
	x	y	Bendra
0°	0.0	0.0	0.0
45°	4.0	5.1	4.5
90°	0.4	14.7	7.5
135°	8.2	8.8	8.5
180°	4.3	1.1	2.7
225°	10.2	3.9	7.1
270°	6.8	0.3	3.6
315°	10.3	7.5	8.9
360°	15.0	11.1	13.1



2.7 pav. „X“ ir „Y“ ašių paklaida laipsniais atvaizduota grafiškai



2.8 pav. Bendra „X“ ir „Y“ ašių paklaida laipsniais atvaizduota grafiškai

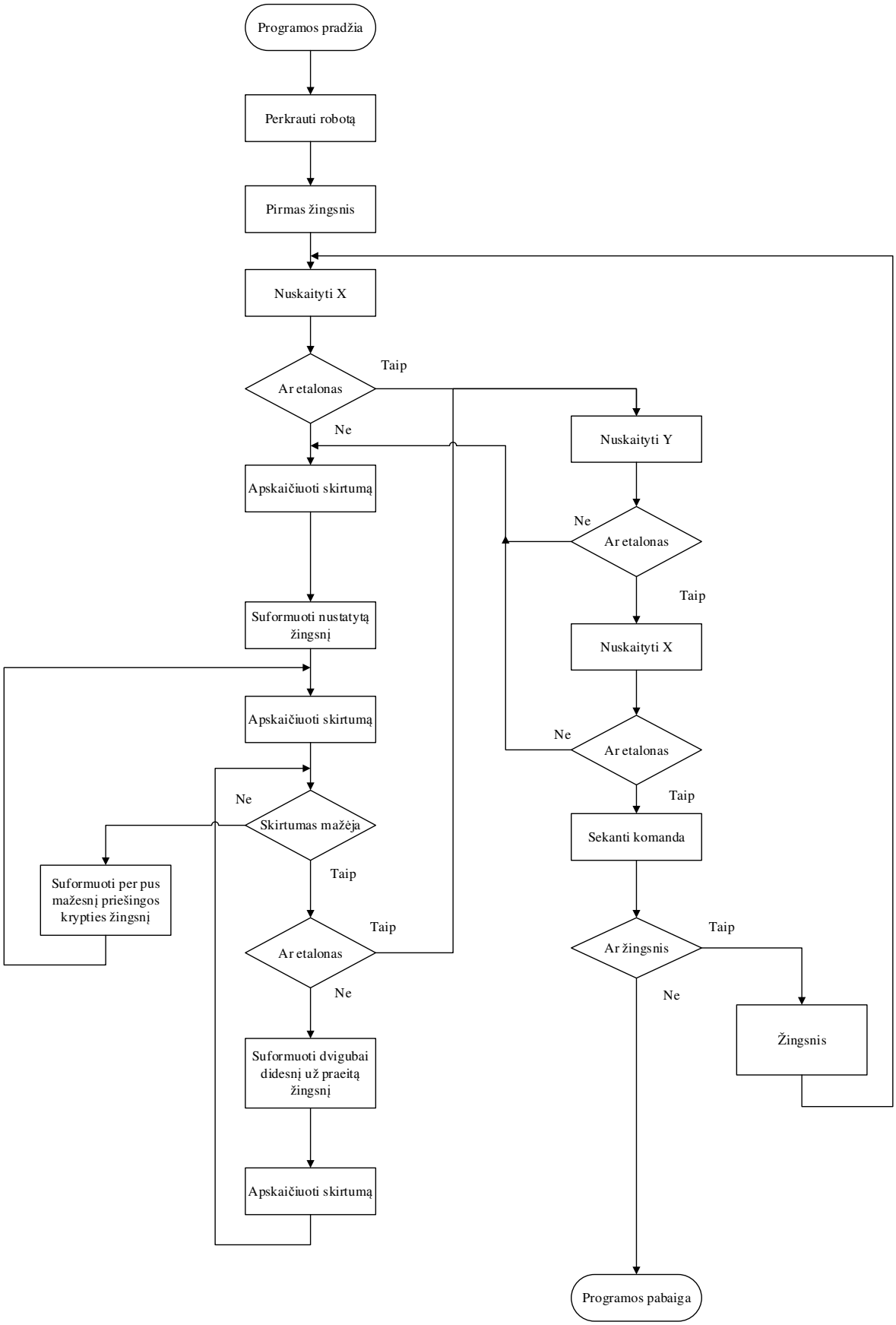
Remiantis gautais eksperimentiniais rezultatais, buvo nuspręsta koordinatę laikyti atitinkančia etaloną kai skirtumas yra mažesnis kaip 3%, kadangi net robotui nejudant akcelerometro išėjime yra tam tikros deviacijos ir iš eksperimentinių duomenų, kuomet robotas buvo pastatytas ranka tam tikru kampu į tikslią pozicija, matoma, kad koordinatės reikšmė kinta maždaug per 20 punktų. Taip pat nustatytas minimalus korekcijos žingsnis 20, kurį įvykdžius robotas pasisuka apie 2 laipsnius.

2.2.2 Nuokrypų nuo užduotos trajektorijos tikslinimo algoritmo sudarymas

Apžvelgęs skirtingus algoritmus ir metodus mobilaus roboto pozicijai nustatyti bei atlikęs eksperimentus su robotu ir išanalizavęs gautus duomenis nusprendžiau remtis vienu iš greičiausio nusileidimo metodų: „Rosenbroko“.

Išanalizavęs „Rosenbroko“ metodą sudariau roboto maršruto tikslinimo algoritmą, kuris kiekvieno žingsnio pabaigoje tikrins ir, jei reikia, koreguos roboto pasisukimo kampą, kad atitiktų eksperimentiškai nustatytas etalonines akcelerometro ašių vertes. 2.9 paveikslas.

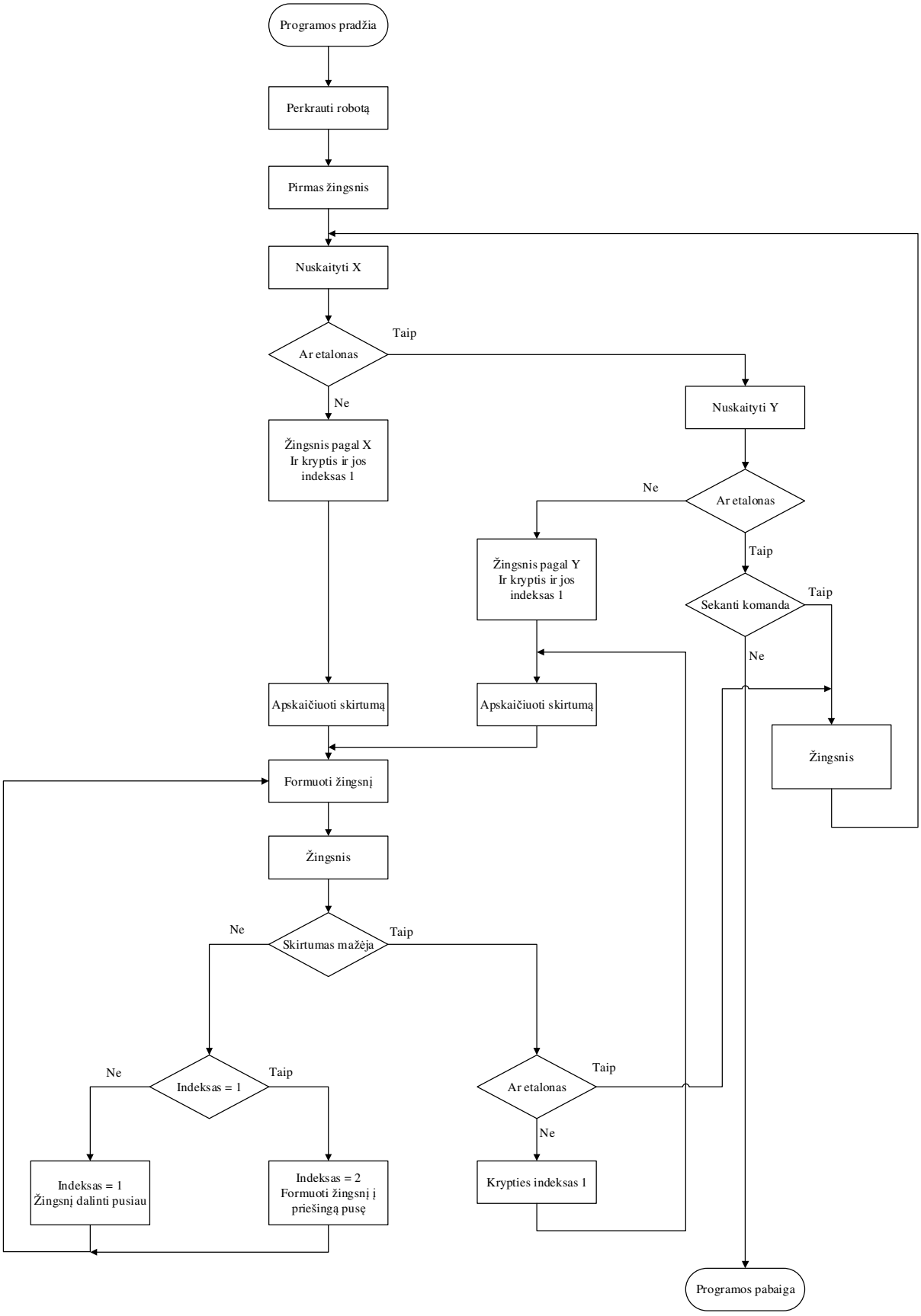
Šiuo metodu kiekviena koordinatė yra derinama paeiliui, pirmiausia derinama viena koordinatė kol tenkina nustatytos paklaidos vertes. Tuomet derinama kita koordinatė, kol atitinka vertes. Kai gauti duomenys atsižvelgiant į nustatytą paklaidą tenkina tikslą, skaitome, kad pasisukimo kampo tikslinimas atliktas sėkmingai. Šis derinimas vyksta formuojant ir vykdant skirtingo dydžio ir skirtingos krypties pasisukimo komandas atsižvelgiant į gautus rezultatus po kiekvienos korekcijos.



2.9 pav. Mobilaus roboto maršruto tikslinimo algoritmas

Startavus programai perkraunamas robotas „Khepera II“. Tuomet iš komandų sąrašo inicijuojamas pirmas žingsnis - pirma komanda saraše. Atlikus užduotą žingsnį nuskaitomas roboto analoginis kanalas, kuriuo nustatoma „X“ koordinatės padėtis 0-1020 ribose, kas atitinka 0-4,09V įtampą. Gavus „X“ koordinatę ji lyginama su etalonine koordinate, jei skirtumas mažesnis arba lygus 10, tuomet užskaitoma kaip atitinkanti etaloną. Jei neatitinka, tuomet skaičiuojamas skirtumas ir suformuojama „C,10,10“ komanda, kuri atitinka 1.8 laipsnio pasisukimą pagal laikrodžio rodyklę. Vėl skaičiuojamas skirtumas ir lyginamas su paskutiniu apskaičiuotu. Jei skirtumas didėja tuomet formuojamas per pus mažesnis žingsnis praeitam, tačiau jau priešinga sukimosi kryptimi. Ir vėl skaičiuojamas skirtumas, jei skirtumas galiausiai pradeda mažėti, tada lyginamas su etaloniniais duomenimis, jeigu neatitinka, tuomet formuojamas dvigubas žingsnis paskutiniam ir vėl skaičiuojamas skirtumas ir lyginamas su buvusiu. Kai atitinka „X“ koordinatė etaloninius duomenis, nuskaitoma „Y“ koordinatė ir atliekami tie patys žingsniai kaip ir su „X“ koordinate, kol galiausiai tenkina nustatytas skirtumo ribas. Tuomet dar kartą patikrinama „X“ koordinatė, jeigu neatitinka - vėl vykdomas koordinatės tikslinimas. Jeigu koordinatė atitinka - nuskaitoma sekanti komanda. Jeigu tai judėjimo komanda, tuomet vykdomas kitas žingsnis, jeigu ne, tuomet programos algoritmas užsibaigia.

Atlikus eksperimentinius algoritmo tyrimus buvo pastebėta, kad atlikus korekciją ir kelis kartus esant teigiamam pokyčiui, formuojamas labai didelis žingsnis, tarkime, pasukus minimaliu nustatytu žingsniu ir gavus mažesnį skirtumą, sukama dvigubu žingsniu, gavus dar geresnį rezultatą sukama vėl dvigubu paskutiniajam, todėl jau 3 korekcijos žingsnyje užduotas pasisukimas tampa 4 kartus didesnis už pirminį korekcijos žingsnį. Tokiu atveju dažniausiai jau trečiame korekcijos žingsnį yra gerokai peršokama norima padėtis, tuomet keičiama korekcijos kryptis ir vėl einama link atitikimo. Todėl roboto pozicijos korekciją galima būtų palyginti su silpnai slopstančiais švytavimais ir tam prireiktų nemažai žingsnių. Todėl buvo tobulinamas ir koreguojamas algoritmas, sukurtas patobulintas algoritmas pavaizduotas 2.10 paveiksle. Šiame algoritme įvedamas indeksas, kuris nusako roboto kryptį ir žingsnio dydį.

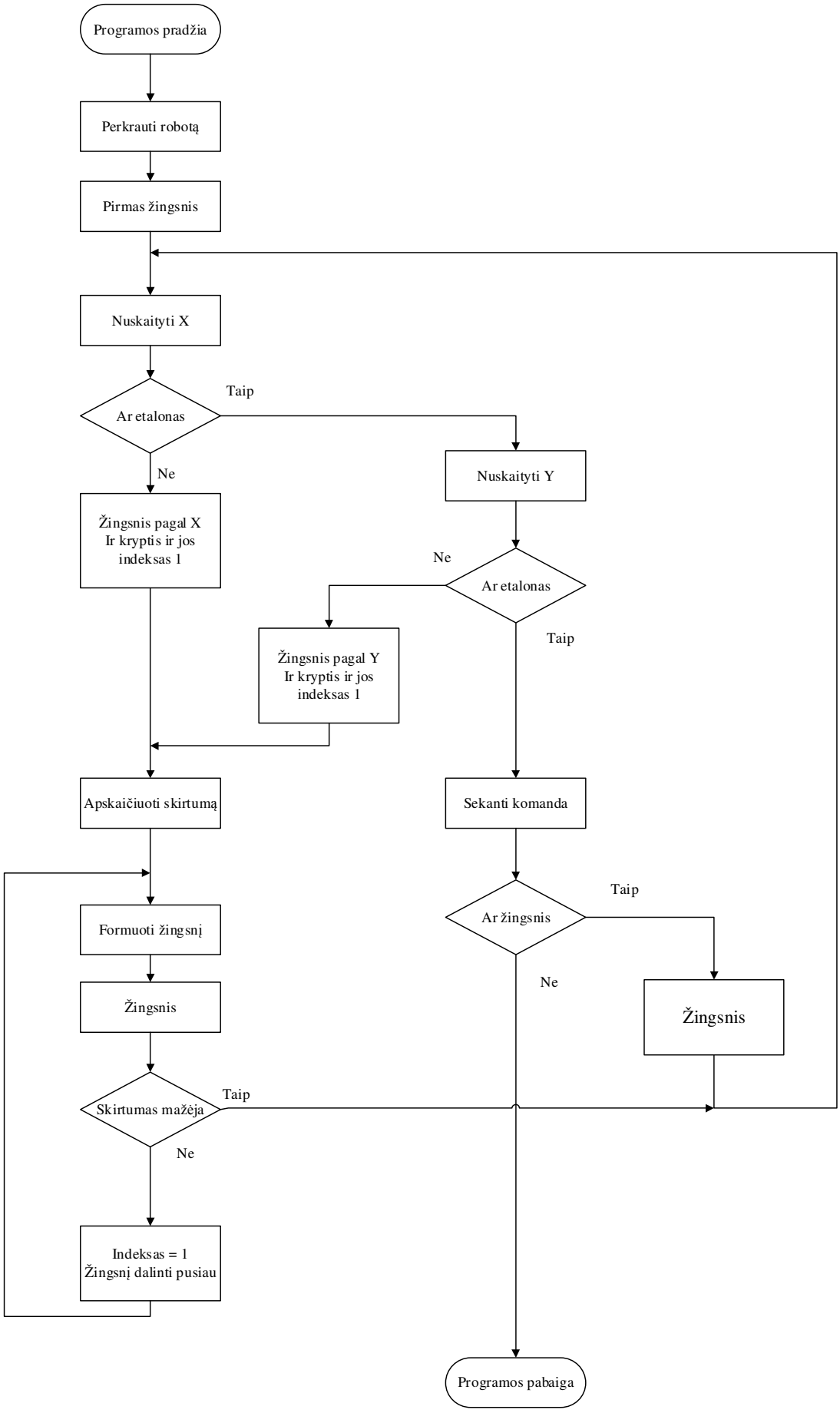


2.10 pav. Koreguotas mobilaus roboto maršruto tikslinimo algoritmas

Startavus programai robotas perkraunamas ir įvykdomas pirmas žingsnis. Tuomet nuskaitomas analoginis mobilaus roboto įėjimas į kurį siunčiami duomenys iš akcelerometro „X“ ašies. Gauta „X“ koordinatė lyginama su etalonine reikšme, jeigu reikšmė tenkina nustatytą +/- 3% paklaidą nuskaitoma „Y“ koordinatė ir taip pat lyginama su etalonu ir jeigu atitinka tuomet vykdomas kitas žingsnis arba programa užsibaigia jeigu robotas įvykdė visus užduotus žingsnius.

Kai „X“ koordinatė netenkina etaloninių sąlygų, į kintamųjų sąrašą įvedamas žingsnio ir krypties indeksas 1, kuris reiškia, kad bus atliekamas pirmas pasisukimas į dešinę. Taip pat apskaičiuojamas skirtumas tarp etaloninių duomenų ir gautų. Tuomet robotas pasukamas minimaliu žingsniu ir vėl skaičiuojamas skirtumas tarp gautų dydžių. Jeigu skirtumas nemažėja ir indeksas buvo nelygus 1 tuomet žingsnis dalinamas pusiau ir formuojamas dar vienas žingsnis ir t.t. Jeigu skirtumas nemažėja, o indeksas nustatytas 1, tuomet formuojamas priešingas žingsnis ir jis įvykdomas. Vėl matuojami skirtumai ir jei skirtumas nemažėja viskas kartojama per naują. Jeigu skirtumas mažėja, tuomet tikrinama ar koordinatė atitinka etaloną. Jeigu atitinka - vykdomas sekantis žingsnis arba užbaigiama programa. Tačiau jeigu skirtumas mažėja, bet tinkamos reikšmės dar nepasiekė, tuomet vėl skaičiuojamas skirtumas, formuojamas žingsnis kuris yra įvykdomas ir vėl stebima, ar skirtumas sumažėjo, ar ne ir t.t., kol tiek „X“, tiek „Y“ tenkina sąlygas.

Šiame algoritme komandos formavimui naudojama „Centaurus CPN“ funkcija *„ForPort(a:int):string ="C,"^IntDecimal(a)^","^IntDecimal(~a);“* Kuri formuoja roboto pasisukimo komandą pagal apskaičiuotą skirtumą nuo etaloninių duomenų. Tačiau ištyrus eksperimentinius duomenis buvo pastebėta, kad skirtumas tarp etaloninių ir realių duomenų gali būti artimas 100. Tokiu atveju formuojama komanda, kuri pasuktų robotą net 20 laipsnių. Atmetus korekcijos komandos formavimą nuo skirtumo dydžio, buvo nuspresta korekciją daryti naudojant tik minimalų ir per pus mažesnį priešingą žingsnį. Šis algoritmas pavaizduotas 2.11 paveiksle.



2.11 pav. Galutinis mobilaus roboto maršruto tikslinimo algoritmas

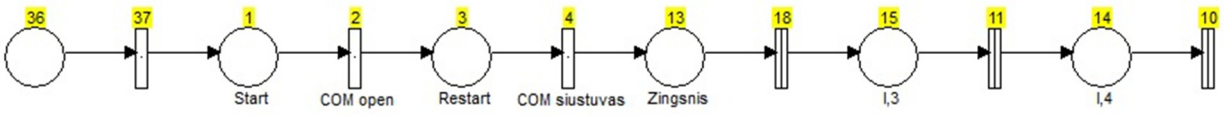
Startavus programai robotas perkraunamas, įvykdomas pradinis žingsnis iš sąrašo. Įvykdžius žingsnį nuskaitoma „X“ koordinatė, kuri yra lyginama su etaloniniais duomenimis, jeigu koordinatė sutampa, nuskaitoma „Y“ koordinatė ir taip pat lyginama su etalonu, jeigu atitinka vykdomas sekantis žingsnis arba užbaigiama programa.

Esant „X“ koordinatės neatitikimui nustatomas žingsnio ir krypties indeksas lygus 1, kuris atitinka roboto sukimasi pagal laikrodžio rodyklę ir roboto pasisukimo žingsnį. Tuomet skaičiuojamas skirtumas tarp gautų duomenų ir etaloninių. Apskaičiavus skirtumą suformuojamas ir įvykdomas korekcijos žingsnis. Įvykdžius žingsnį skaičiuojama, ar skirtumas mažėja ar didėja. Skirtumui didėjant formuojamas per pus mažesnis žingsnis į priešingą pusę. Skirtumui mažėjant kartojamas suformuotas žingsnis kol nuskaityta koordinatė tenkins etaloninius duomenis. Jeigu įvykdžius žingsnį skirtumas tampadidesnis už prieš tai buvusį, tuomet formuojamas per pus mažesnis priešingos krypties žingsnis.

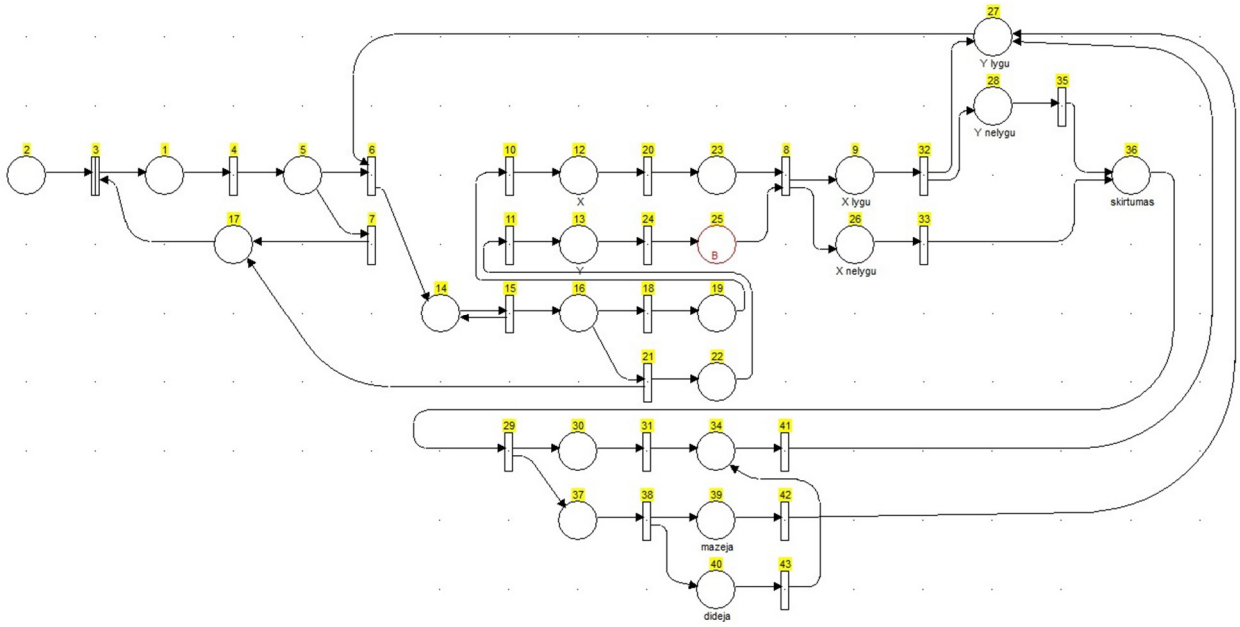
Šiame algoritme atmetus žingsnio kitimą, jis tampa ženkliai paprastesnis įgyvendinti programiškai, taip pat roboto pasisukimo kampo patikslinamui naudojama mažiau žingsnių, dėl to korekcijos laikas ženkliai sutrumpėja.

2.2.3 Maršruto tikslinimo programos realizavimas

Mobilaus roboto maršruto uždavimui naudojamas programos fragmentas pateiktas 2.12 paveiksle. Čia iš sąrašo komandų imamas žingsnis, kuris yra įvykdomas ir kiekvieno žingsnio pabaigoje nuskaitomi analoginiai kanalai, kuriuos nuskaičius formuojama eilutė kuri apdorojama 2.13 paveiksle.

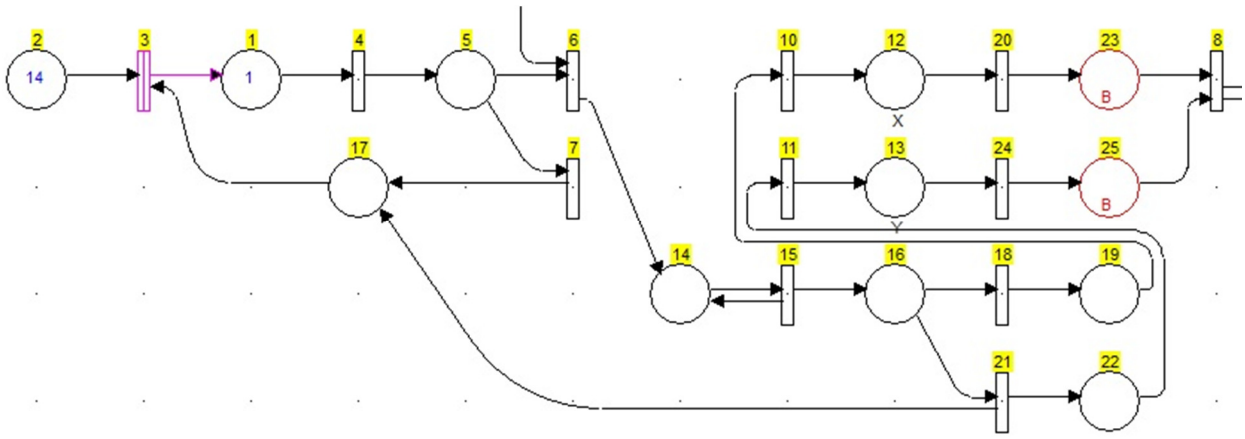


2.12 pav. Roboto judesio komandos uždavimas ir analoginių kanalų nuskaitymas



2.13 pav. Gautų duomenų apdorojimas ir korekcijos vykdymas

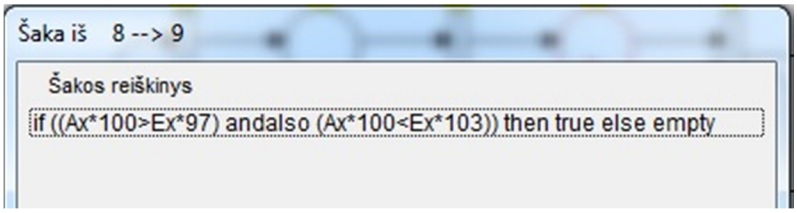
Nuskaičius roboto analoginius kanalus, gaunama eilutė, kurios apdorojimui naudojamas pertri tinklo fragmentas, pavaizduotas 2.14 paveiksle. Pirmajame žingsnyje suaktyvavus perdavą Nr. 2 yra užduodama kiek iš viso žetonų bus perduodama ir kiek pozicijų bus stebima. Perdavoje Nr. 3 užduodamas vėlinimas. Į perdavą Nr. 1 siunčiama tekstinė eilutė, perdavoje Nr. 4 tikrinama, kiek simbolių sudaro šią eilutę. Į perdavą Nr. 5 atsiunčiamas atsakymas, perdavoje Nr. 6 tikrinama eilutę sudarančių simbolių skaičius didesnis už 8, o perdavoje Nr. 7 tikrinama ar simbolių skaičius yra mažesnis už 8. Kuomet simbolių skaičius neviršija 8, į perdavą Nr. 17 atsiunčiamas reiškinys *„TRUE“* ir žetonas yra grąžinamas į perdavą Nr. 3. Tačiau jei perdavoje Nr. 6 simbolių skaičius viršija 8, tai į perdavą Nr. 14 atsiunčiamas simbolių skaičius. Perdavose Nr. 15, 16, 18, 19 pagal užduotas sąlygas nustatoma kurioje eilutės „I,746,820“ vietoje žiūrint iš dešnės į kairę yra pirmasis kablelis. Rezultate kabutės ir kablelis yra pašalinami, paliekant likusią eilutės dalį. Iš perdavos Nr. 10 į Nr. 12 *„string“* kintamasis keičiamas į *„integer“*. Perdavos Nr. 12 rezultate turime kintamąjį *„integer“* 2.15 pav., kuris ir yra „X“ koordinatė naudojama tolimesniuose matavimuose. Analogiškai gaunama ir „Y“ koordinatė.



2.14 pav. „X“ ir „Y“ koordinačių gavimas

2.15 pav. Gauta „X“ koordinatė

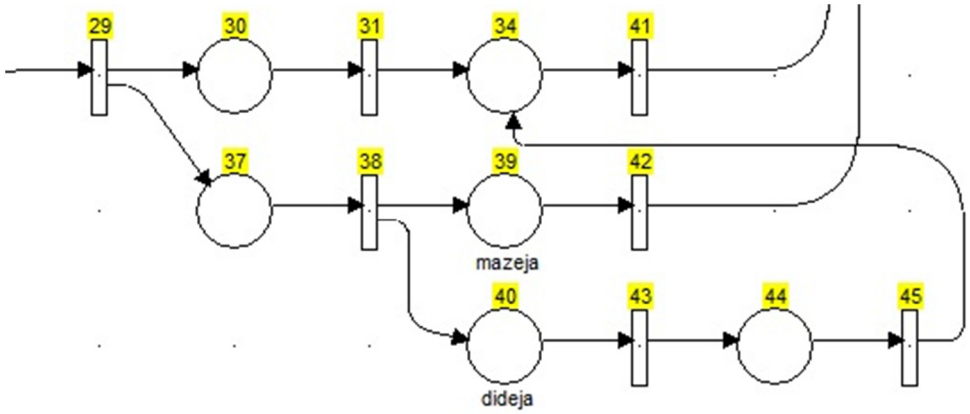
Toliau gautos koordinatės tikrinamos ar atitinka etaloninius duomenis. „X“ koordinatė tikrinama šakoje iš 8 perdavos į 9 ir 26 pozicijas. Šioje šakoje skaičiuojama ar nuskaityta „X“ koordinatė, kuri priskiriama „Ax“ globaliam kintamajam ir atitinka nustatytas 3% paklaidos vertes lyginant su etaloniniu dydžiu, kuris įrašomas į „Ex“ kintamąjį. Jeigu koordinatė tenkina nustatytas sąlygas, šakoje iš 8 perdavos į 9 žetoną, aktyvuojamas 9 žetonas ir laikoma, kad koordinatė atitinka etaloninius duomenimis 2.16 paveikslas. Jeigu sąlygų netenkina, tuomet aktyvuojamas 26 žetonas ir toliau skaičiuojamas skirtumas tarp etalono ir gautos koordinatės atimant iš etaloninio dydžio „Ex“ gautą koordinatę „Ax“ 2.17 paveikslas.



2.16 pav. Gauta „X“ koordinatė lyginama su etaloniniais duomenimis

2.17 pav. Skaičiuojamas skirtumas

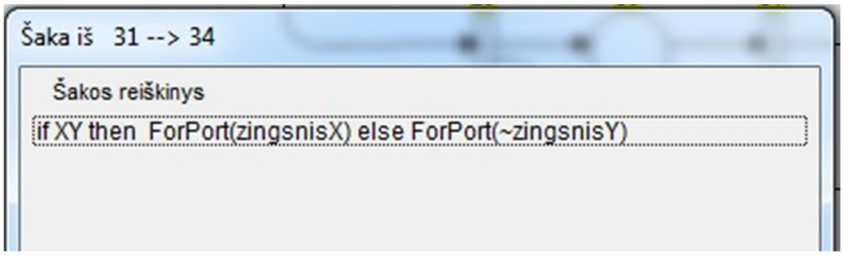
Šis apskaičiuotas skirtumas įrašomas į kintamąjį „n“. Toliau formuojamas nustatytas pradinis pasisukimo žingsnis, po kurio vėl nuskaitoma koordinatė, paskaičiuojamas skirtumas ir įrašomas į „n1“ kintamajė ir skaičiuojama, ar skirtumas sumažėjo ar padidėjo lyginant naujai gautą skirtumą „n1“ su buvusiu skirtumu „n“. Korekcijos žingsnio formavimas ir skirtumo lyginimas pavaizduotas 2.18 paveiksle.



2.18 pav. Korekcijos žingsnio formavimas

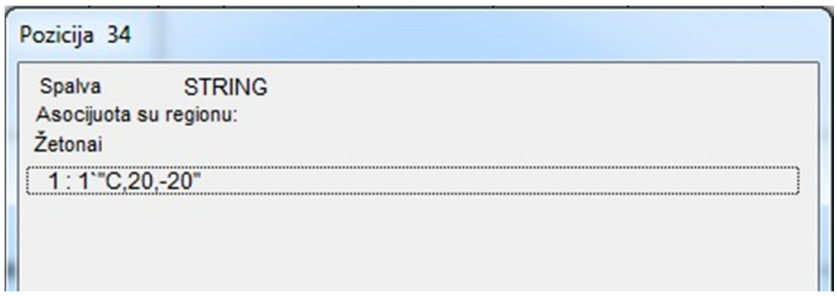
Pradiniam ir vėlesniem žingsniam jeigu skirtumas mažėja, naudojama funkcija pavaizduota 2.19 paveiksle. Čia naudojami kintamieji „zingsnisX“ ir „~zingsnisY“, į kuriuos įrašomas žingsnio dydis. „~“ ženklas reiškia, kad reikšmė neigiama.

Išbandžius korekcijos programą eksperimentiškai buvo nustatyta, kad pirminis sumanymas žingsnio dydį formuoti atsižvelgiant į skirtumo dydį yra netikslingas. Kadangi iš eksperimentinių duomenų gauta, kad skirtumas gali viršyti 80, todėl suformavus tokį pat arba per pus mažesnį žingsnį būtų netikslinga, nes nėra tiesioginės priklausomybės tarp gauto skirtumo ir nuklydimo laipsniais, tokiu atveju galima net pabloginti esamą situaciją, tuomet programa gali užsigeneruoti ir sukinėdama robotą į abi puses dideliais žingsniais bei taip ir nesurasti tinkamos pozicijos. Buvo nuspręsta formuoti minimalų pasisukimo žingsnį, kurį galėtų atlikti robotas, t.y. „10,-10“, kuris atitinka apie 2 laipsnių pasisukimą palei laikrodžio rodyklę. Tačiau kokybiškam algoritmo veikimui reikalingas ir per pus mažesnis žingsnis, kuris bus naudojamas pradedant didėti paklaidai. Tokiu atveju buvo galutinai nuspręsta naudoti „20,-20“ žingsnį, kaip pradinį ir sekančius žingsnius esant teigiamai skirtumo mažėjimo sąlygai.



2.19 pav. Pradinio korekcijos žingsnio formavimas

Suformuotas korekcijos žingsnis perduodamas į 34 žetoną įvykdžius sąlygas užduotas šakoje iš 31 perdavos. Šis žingsnis pavaizduotas 2.20 paveiksle.



2.20 pav. Suformuota žingsnio komanda

Jeigu atlikus korekcijos žingsnį skirtumas pradeda didėti, tuomet formuojamas per pus mažesnis priešingos krypties žingsnis. Jo formavimui pasitelkiama funkcija „*div*“, kuri dalina žingsnio vertę pusiau. Ši funkcija vykdoma 43 perdavoje, čia 2.21 paveikslas, o jos rezultatas perduodamas šakoje iš 45 į 34 elementą, 2.22 paveisklas.

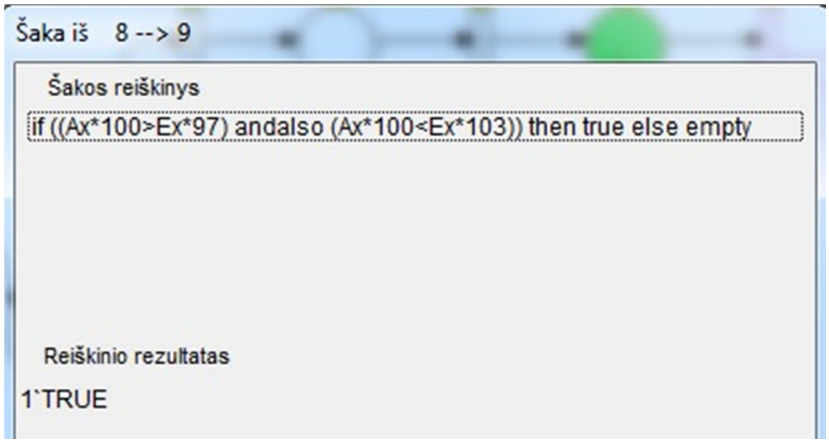
2.21 pav. Per pus mažesnio žingsnio formavimas

2.22 pav. Suformuotas per pus mažesnis žingsnis

Iš 2.22 paveikslo matome 43 perdavoje inicijuoto reiškinio rezultatai įvesti į „zingsnisX“ ir „~zingsnisY“ kintamuosius ir suformuota judesio komanda „C,-10,10“ kuri yra per pus mažesnė ir priešingos krypties negu buvo formuojama pradiniu korekcijos žingsniu.

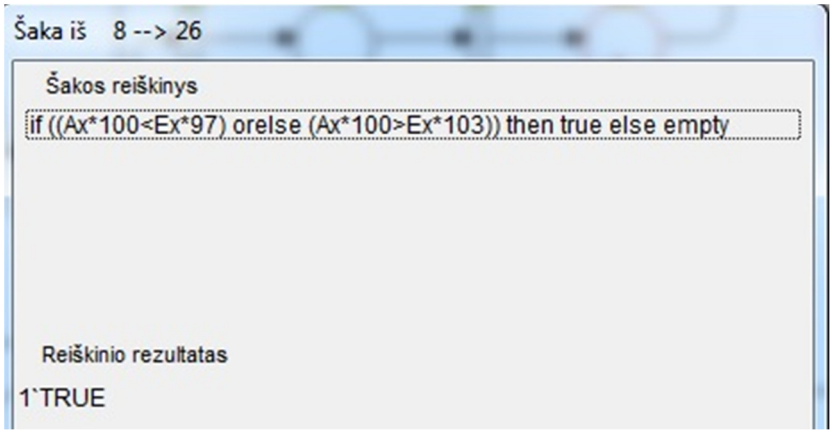
2.2.4 Eksperimentiniai tyrimai

Atliekamas eksperimentas su robotu kuomet užduodama komanda pavažiuoti apie 10 cm, tada sustojus pasitikrinti orientaciją ar nepakito roboto pasisukimo kampas, jei pakito reikia sureguliuoti iki nustatytų ribų, jeigu atitinka duomenis, tuomet pasisuka 90 laipsnių pagal laikrodžio rodyklę ir vėl tikrina padėtį ir reikalui esant koreguoja pasisukimo kampą. Įvykdžius komandą važiuoti pirmyn nuskaitomi duomenys ir lyginami su etaloniniais duomenimis, gaunama, kad inicijuotas reiškinys šakoje iš 8 perdavos į 9 poziciją yra teigiamas, 2.23 paveikslas, tai reiškia, gad nuskaityta „X“ koordinatė tenkina užduotą sąlygą ir atitinka etaloninius duomenis. Taip pat ir „Y“ koordinatė atitiko duomenis, todėl buvo vykdoma 90 laipsnių pasisukimo komanda.



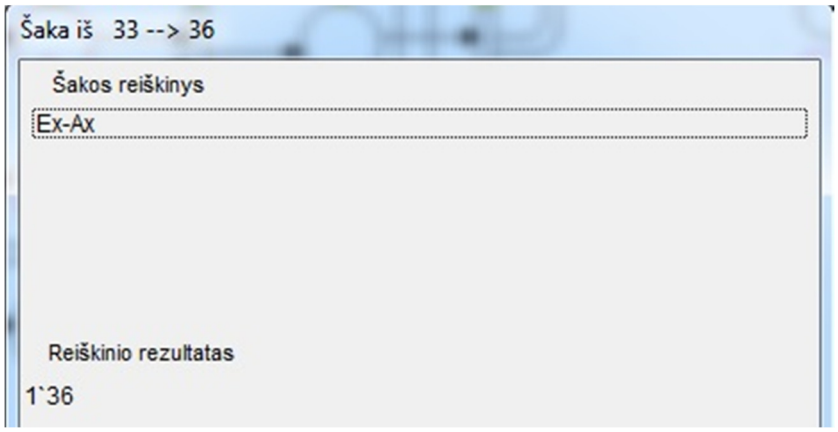
2.23 pav. Šakos iš 8 perdavos į 9 poziciją reiškinys ir gautas rezultatas

Pasisukus robotui 90 laipsnių kampu algoritmo programa vykdoma toliau ir vėl nuskaitomos koordinatės. Nuskaičius koordinates programa užfiksavo neatitikimą iškeltom sąlygom, todėl buvo aktyvuota 26 pozicija, kuri reiškia, kad „X“ koordinatės paklaida yra didesnė kaip 3 procentai lyginant su etalonine, 2.24 paveikslas.



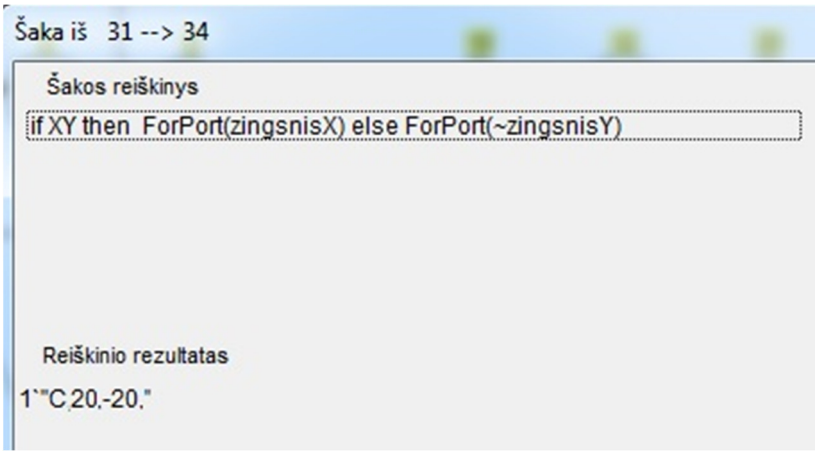
2.24 pav. Aktyvuota šaka iš 8 perdavos į 26 poziciją

Aktyvavus šaką iš 33 perdavos į 36 poziciją iš įvykdžius šakos reiškinį gautas skirtumas „36“, kuris perduodamas į 36 poziciją, 2.25 paveikslas.



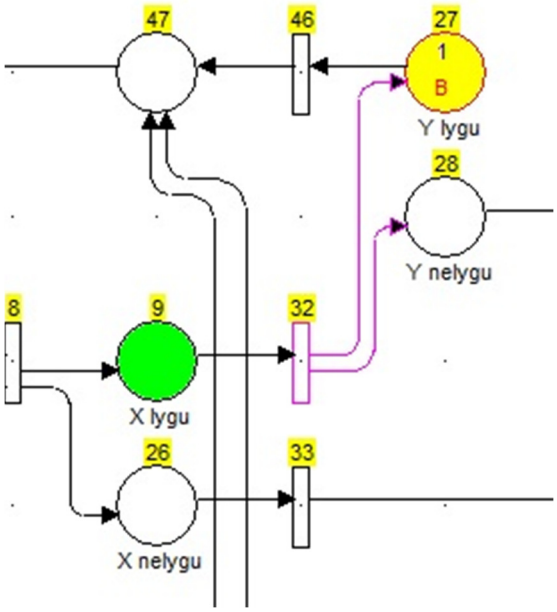
2.25 pav. Šakos reiškinys ir užbaigimo rezultatas

Formuojamas pasisukimas į dešinę ir vėl skaičiuojamas ar atitinka užduotas reikšmes, suformuotas pasisukimo kampas atvaizduotas 2.26 paveiksle.



2.26 pav. Suformuotas korekcijos žingsnis

Įvykdžius šį žingsnį ir patikrinus ar atitinka etaloninius duomenis buvo gautas teigiamas atsakymas į 9 ir 27 pozicijas, 2.27 paveikslas, todėl toliau programa nebuvo vykdoma, nes nebuvo užduota daugiau roboto judesių.



2.27 pav. Gauti teigiami „X“ ir „Y“ koordinačių duomenys

Taip pat realizavus sukurtą algoritmą „Centaurus CPN“ programiniame pakete buvo atlikta po 5 eksperimentus pasukant robotą kas 45° nuo 0° iki 360° pasisukimo ribose. Šie duomenys buvo užfiksuoti ir suvesti į 2.7 lentelę.

2.7 lentelė. Gauti duomenys

Ašis / Kampas	Sukosi pats										vidurkis	
	x	y	x	y	x	y	x	y	x	y	x	y
0°	852	772	860	776	852	780	864	780	860	784	858	778
45°	809	780	812	776	794	782	784	774	788	786	797	780
90°	764	814	768	816	756	802	758	812	774	808	764	810
135°	702	792	712	794	700	790	714	796	700	788	706	792
180°	730	786	732	790	722	784	724	774	718	796	725	786
225°	702	708	698	704	694	712	700	688	688	696	696	702
270°	758	684	760	688	764	694	756	670	756	678	759	683
315°	846	728	834	736	850	734	838	720	846	732	843	730
360°	836	768	842	764	844	756	830	760	838	762	838	762

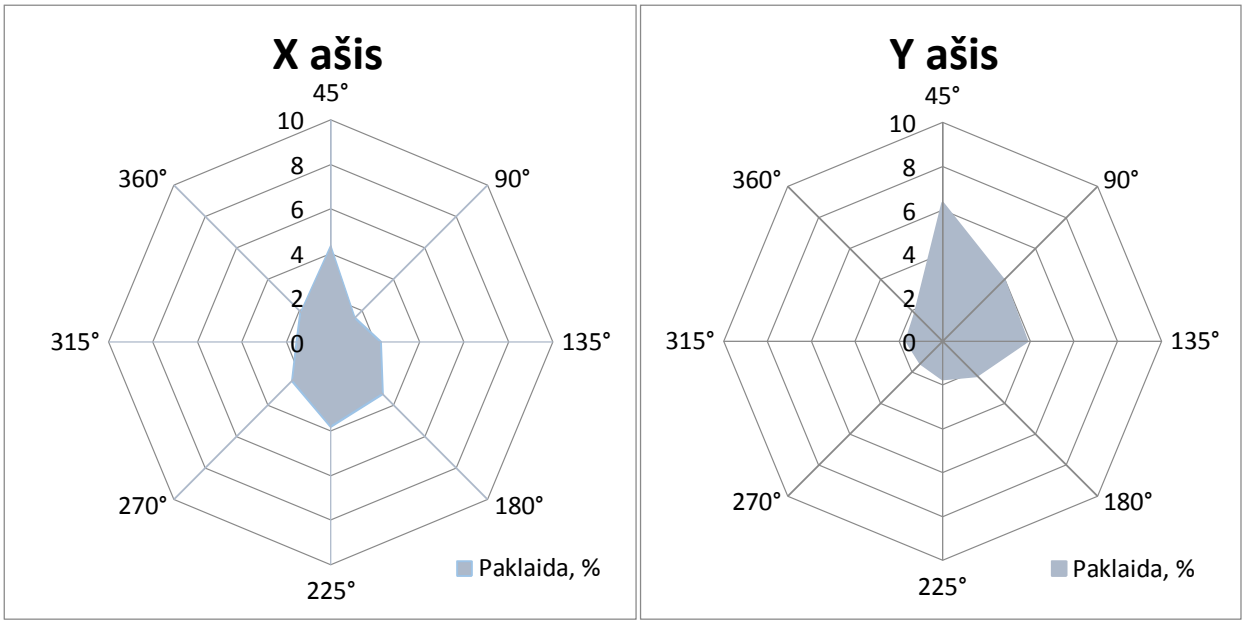
Iš gautų reikšmių buvo paskaičiuotos procentinės paklaidos. Palyginus su duomenimis, gautais 2.8 lentelėje, matome, kad nebėra paklaidų artimų ar viršijančių 10%. Tačiau kai kurie

rezultatai ir suprastėjo, tarkim, ties 180° paklaida išaugo beveik dvigubai. Tačiau tai nutiko todėl, kad užduota ganėtinai didelė paklaida dėl akcelerometro išėjimo vertės dreifo, kadangi net nejudant robotui duomenys šiek tiek kinta, todėl negalima gauti visiškai tikslios padėties ir reguliavimo kokybės.

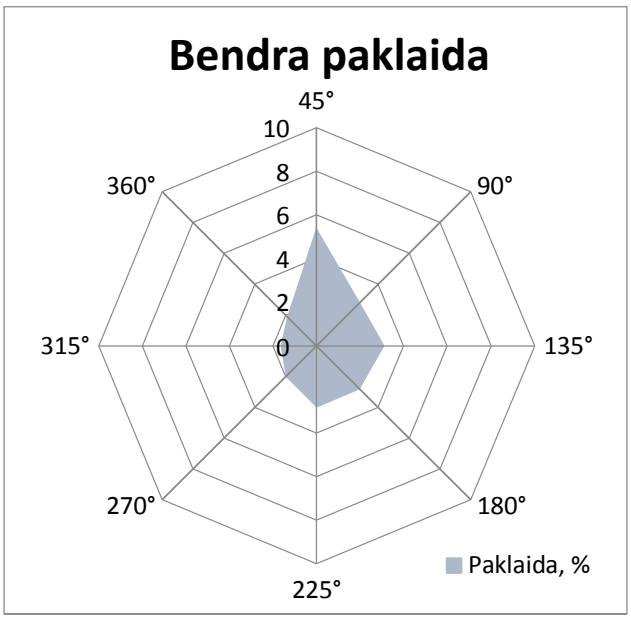
2.8 lentelė. Gauti duomenys

Ašis / Kampas	Paklaida nuo vidurkio, %		
	x	y	Bendra
0°	0.0	0.0	0.0
45°	4.3	6.3	5.3
90°	1.5	4.0	2.8
135°	2.3	3.9	3.1
180°	3.3	2.2	2.8
225°	3.8	1.7	2.8
270°	2.5	1.4	2.0
315°	1.5	1.7	1.6
360°	1.9	1.8	1.9

Taip pat šie duomenys atvaizduojami grafiškai 2.28 - 2.29 paveiksluose, iš kurių geriau matosi, kokia paklaida buvo prie tam tikrų kampų. Taip pat matosi, kad maksimali paklaidos vertė sumažėjo net per pusę.



2.28 pav. Procentinė „X“ ir „Y“ ašių paklaida atvaizduota grafiškai

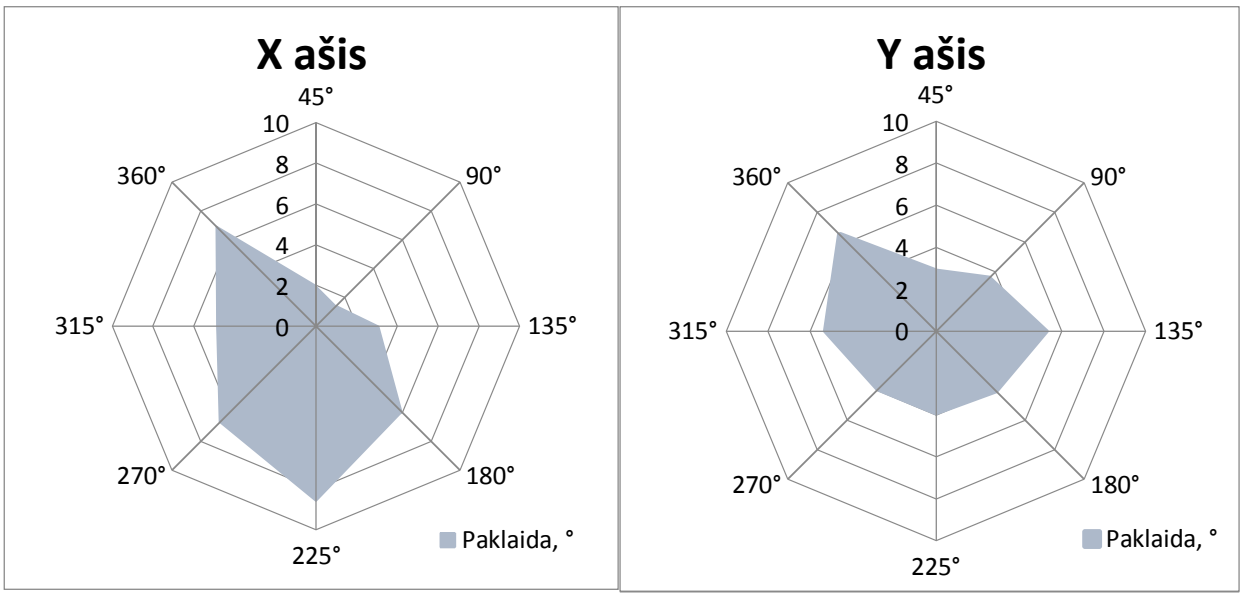


2.29 pav. Bendra procentinė „X“ ir „Y“ ašių paklaida atvaizduota grafiškai

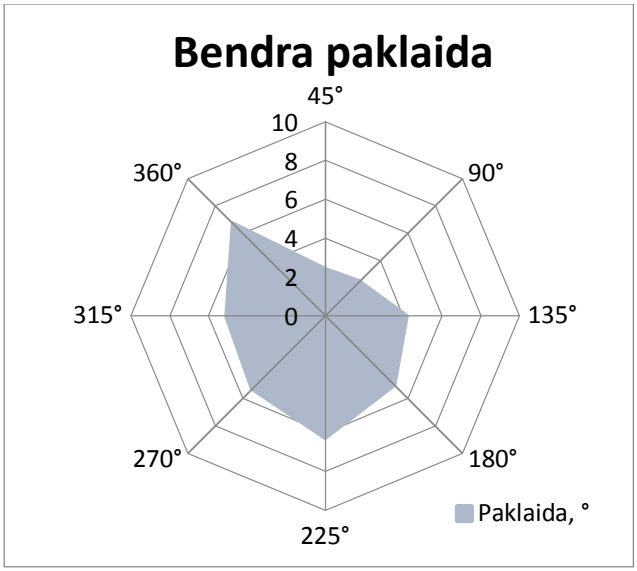
2.9 lentelėje suvesti gauti duomenys perskaičiuoti į laipsnius. Matome, kad maksimali bendra paklaida yra 6.7° lyginant su 13.1° gautų be reguliavimo ir maksimali ašinė paklaida gauta 8.6° lyginant su 15° gautų be reguliavimo. Šie duomenys atvaizduoti grafiškai 2.30 – 2.31 paveiksluose.

2.9 lentelė. Gauti duomenys

Ašis / Kampas	Paklaida nuo vidurkio, °		
	x	y	Bendra
0°	0.0	0.0	0.0
45°	2.0	2.9	2.4
90°	1.4	3.6	2.5
135°	3.1	5.2	4.2
180°	5.9	4.0	5.0
225°	8.6	3.9	6.2
270°	6.7	3.9	5.3
315°	4.9	5.3	5.1
360°	6.9	6.5	6.7



2.30 pav. „X“ ir „Y“ ašių paklaida laipsniais atvaizduota grafiškai



2.31 pav. Bendra „X“ ir „Y“ ašių paklaida laipsniais atvaizduota grafiškai

3 Išvados ir rezultatai

1. Išanalizuotos priemonės, naudojamos mobilaus roboto maršruto nustatymui, bei atlikta jų taikymo analizė. Iš šios apžvalgos pastebėta jog renkantis aparatūrą mobilaus roboto lokalizacijai būtina atsižvelgti ne tik į atstumus, kuriais keliaus robotas, bet taip pat ir paviršiaus tipą, pavirtimą, papildomos įrangos įrengimo aplinkoje galimybes.
2. Atlikti eksperimentai, kurių metu nustatyti tikslūs akcelerometro duomenys esant tam tikriems kampams, taip pat gauti duomenys iš realaus roboto bei juos palyginus apskaičiuotos gaunamos nuokrypos nuo užduoto pasisukimo kampo, kurios siekė iki 13 laipsnių.
3. Atlikus skirtingų algoritmų ir metodų analizę pasirinktas priimtiniausias metodas mobilaus roboto pasisukimo kampui nuklydus nuo užduotos kelio trajektorijos tikslinimui. Pagal pasirinktą metodą sudarytas mobilaus roboto maršruto tikslinimo algoritmas bei jis realizuotas „Centaurus CPN“ programiniame pakete.
4. Sukurta programa išbandyta su realiu mobiliu robotu, gauti duomenys išanalizuoti ir palyginti su pradiniais duomenimis. Iš gautų rezultatų matome, kad maksimali bendra paklaida yra 6.7° lyginant su 13.1° gautų be reguliavimo ir maksimali ašinė paklaida gauta 8.6° lyginant su 15° gautų be reguliavimo.
5. Iš gautų rezultatų matosi, kad yra pasiekiami geresni rezultatai roboto valdyme. Tačiau paklaidos išlieka dėl akcelerometro išėjimo duomenų dreifo. Todėl galima teigti, kad algoritmas atlieka savo užduotį, tačiau didelę įtaką pozicionavimo tikslumui turi aparatinė įranga.

4 Literatūros sąrašas

1. Sooyong Lee, Jae-Bok Song „Robust mobile robot localization using optical flow sensors and encoders“ [žiūrėta 2014-06-16]. Prieiga per internetą: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=1307287&queryText%3DRobust+Mobile+Robot+Localization+using+Optical+Flow+Sensors+and+Encoders
2. Josep M. Font-Llagunes, Joaquim A. Batlle „Consistent triangulation for mobile robot localization using discontinuous angular measurements“ [žiūrėta 2014-06-14]. Prieiga per internetą: http://www.sciencedirect.com/science/article/pii/S0921889009000803
3. Leonard, J.J., Durrant-Whyte, H.F. „Mobile robot localization by tracking geometric beacons“ [žiūrėta 2014-06-14]. Prieiga per internetą: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=88147&queryText%3DMobile+Robot+Localization+by+Tracking+Geometric+Beacons
4. Tae-jae Lee, Wook Bahn, Byung-moon Jang, Ho-Jeong Song, and Dong-il “Dan” Cho „A New Localization Method for Mobile Robot by Data Fusion of Vision Sensor Data and Motion Sensor Data“ [žiūrėta 2014-12-05]. Prieiga per internetą: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6491053&tag=1
5. Francisco Martín, Vicente Matellán, Pablo Barrera, José M. Cañas „Localization of legged robots combining a fuzzy-Markov method and a population of extended Kalman filters“ [žiūrėta 2014-06-10]. Prieiga per internetą: http://www.sciencedirect.com/science/article/pii/S0921889007001224
6. D. Herrero-Pérez, H. Martínez-Barberá, K. LeBlanc, A. Saffiotti „Fuzzy uncertainty modeling for grid based localization of mobile robots“ [žiūrėta 2014-06-10]. Prieiga per internetą: http://www.sciencedirect.com/science/article/pii/S0888613X10000782
7. Momotaz Begum, George K.I. Mann, Raymond G. Gosine „Integrated fuzzy logic and genetic algorithmic approach for simultaneous localization and mapping of mobile robots“ [žiūrėta 2014-06-16]. Prieiga per internetą: http://www.sciencedirect.com/science/article/pii/S1568494606001098

8. Juan Pomárico-Franquiz, Sanowar H. Khan, Yuriy S. Shmaliy „Combined extended FIR/Kalman filtering for indoor robot localization via triangulation“ [žiūrėta 2014-06-15]. Prieiga per internetą: http://www.sciencedirect.com/science/article/pii/S0263224114000062
9. Muhammad Latif Anjum, Jaehong Park, Wonsang Hwang, Hyun-il Kwon, Jong-hyeon Kim, Changhun Lee, Kwang-soo Kim, and Dong-i “Dan” Cho „Sensor Data Fusion using Unscented Kalman Filter for Accurate Localization of Mobile Robots“ [žiūrėta 2014-12-06]. Prieiga per internetą: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5669779
10. Lei Wang, Xiao-Ping Wang, Qi-di Wu „Ant System algorithm based Rosenbrock function optimization in multi-dimension space” [žiūrėta 2015-01-20]. Prieiga per internetą: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=1174440&queryText%3DRosenbrock
11. Stanislovas Bartkevičius, Kastytis Šarkauskas „Programų paketas CENTAURUS. Modeliavimas, identifikavimas, optimizavimas“ 2003 [žiūrėta 2015-02-05] Prieiga per internetą: https://www.ebooks.ktu.lt/eb/170/programu-paketas-centaurus-modeliavimas-identifikavimas-optimizavimas/
12. Justė Matusevičienė, magistrinis darbas „Mobilaus roboto grįžtamojo ryšio sistemos sukūrimas“, Kauno technologijos universitetas, 2012.
13. Mohamed Jallouli, Lobna Amouri, Nabil Derbel „AN EFFECTIVE LOCALIZATION METHOD FOR ROBOT NAVIGATION THROUGH COMBINED ENCODERS POSITIONING AND RETIMING VISUAL CONTROL“ [žiūrėta 2014-12-10]. Prieiga per internetą: http://yadda.icm.edu.pl/yadda/element/bwmeta1.element.baztech-article-BUJ6-0028-0003