**KAUNO TECHNOLOGIJOS UNIVERSITETAS**
**PANEVĖŽIO TECHNOLOGIJŲ IR VERSLO FAKULTETAS**

**Jonas Mainelis**

# AUTONOMINIO ROBOTO SU LAZERINIU SKENERIU NAVIGACIJOS TYRIMAS

Baigiamasis magistro projektas

**Vadovas**
Prof. dr. Vytenis Sinkevičius

**PANEVĖŽYS, 2017**

**KAUNO TECHNOLOGIJOS UNIVERSITETAS**
**PANEVĖŽIO TECHNOLOGIJŲ IR VERSLO FAKULTETAS**

# AUTONOMINIO ROBOTO SU LAZERINIU SKENERIU NAVIGACIJOS TYRIMAS

Baigiamasis magistro projektas
**Valdymo technologijos (621H66001)**

**Vadovas**
Prof. dr. Vytenis Sinkevičius

**Recenzentas**

**Projektą atliko**
Jonas Mainelis

**PANEVĖŽYS, 2017**

**KAUNO TECHNOLOGIJOS UNIVERSITETAS**

Panevėžio technologijų ir verslo fakultetas
(Fakultetas)

Jonas Mainelis
(Studento vardas, pavardė)

Valdymo technologijos, 621H66001
(Studijų programos pavadinimas, kodas)

„Autonominio roboto su lazeriniu skeneriu navigacijos tyrimas“

# AKADEMINIO SĄŽININGUMO DEKLARACIJA

2017 m. Gegužės 23 d.
Panevėžys

Patvirtinu, kad mano, **Jono Mainelio**, baigiamasis projektas tema „Autonominio roboto su lazeriniu skeneriu navigacijos tyrimas“ yra parašytas visiškai savarankiškai ir 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)*

**TVIRTINU**:
KTU Panevėžio technologijų ir verslo fakulteto
Technologijų katedros vedėjas

_____

20….. ..... .....

# BAIGIAMOJO PROJEKTO UŽDUOTIS

**Išduota studentui:** *Jonui Maineliui* Grupė *PME-5*

**1. Darbo tema:**

Lietuvių kalba: *Autonominio roboto su lazeriniu skeneriu navigacijos tyrimas*

Anglų kalba: *Research on Navigation of Autonomous robot with Laser Scanner*

Patvirtinta 2017 m. kovo mėn. 30 d. dekano potvarkiu Nr. *V25-13-8*

**2. Darbo tikslas:** *Ištirti autonominio roboto navigacijos sistemos tikslumą naudojant lazerinį skanerį.*

**3. Reikalavimai ir sąlygos:** *Ištirti mobilų robotą Pioneer P3-AT. Iištirti SICK LMS100 lazerinio skenerio veikimą navigacijos sistemoje.*

**4. Projekto struktūra.** *Turinys konkretizuojamas kartu su vadovu, atsižvelgiant į BP pobūdį.*

*1. Literatūros apžvalga*
*1.1.Autonominių robotų apžvalga*
*1.2. Roboto orientavimosi galimybes*
*1.3. Naudojamos programinė įrangos apžvalga*
*1.4. Monte Karlo algoritmo apžvalga*
*2.Metodologinė dalis*
*2.1. Roboto navigacijos komponentų sujungimas*
*2.2. Patalpos žemėlapio sudarymas*
*3. Tiriamoji dalis*
*3.1. Navigacijos tikslumo galutiniame taške tyrimas*
*3.2. Navigacijos veikimas dinaminėje aplinkoje*

**5. Ši užduotis yra neatskiriama baigiamojo projekto dalis.**

**6. Projekto pateikimo gynimui kvalifikacinėje komisijoje terminas** *2017-06-01* *(data)*

Užduotį gavau: _____ *2017-02-01*
*(studento vardas, pavardė, parašas)* *(data)*

Vadovas: _____ *2017-02-01*
*(pareigos, vardas, pavardė, parašas)* *(data)*

Mainelis, Jonas. Autonominio roboto su lazeriniu skeneriu navigacijos tyrimas. Magistro baigiamasis projektas / vadovas prof. Vytenis Sinkevičius; Kauno technologijos universitetas, Panevėžio technologijų ir verslo fakultetas fakultetas.

Mokslo kryptis ir sritis: Valdymo technologijos

Reikšminiai žodžiai: *mobilus robotas, lazerinis skeneris, 2D navigacija*

Panevėžys, 2017. 50 p., 44 pav.

## SANTRAUKA

Mobiliųjų robotų populiarumas vis didėja. Darbe analizuojami roboto pozicijos nustatymo būdai. Ištirtas Pioneer P3AT roboto su lazeriniu skeneriu SICK LMS100 navigacijos sistema. Navigacijos tyrimui naudojama ROS (angl. *Robot operating system*) programa. Išanalizuoti ROS programos paketai, kurie reikalingi navigacijai. Suderinami bei sukalibruojami davikliai, kad gautųsi kuo tikslesnės reikšmės. Kad mobilus robotas patikimai judėtų patalpoje, jis turi žinoti kur jis yra. Didžiausia mobiliųjų robotų problema - tai pozicijos patikimumas bei tikslumas. Šiame darbe naudojamas Monte Karlo lokalizacijos metodas. Šis metodas buvo išbandytas eksperimentiškai bei ištirtas jo tikslumas. Tiriamas dalelių kiekio poveikis galutiniam taškui. Pritaikius adaptyvųjį Monte Karlo algoritmą buvo matuojamas kintamas dalelių skaičius. Lazeriniu skeneriu nuolat nuskaitoma aplinka, todėl net jei roboto kelyje atsiranda nežinomų kliūčių, jis gali perplanuoti maršrutą. Keičiant navigacijos koeficientus, tikrinamas poveikis roboto judesiui. Žemėlapio ir maršruto planavimas atliekamas ROS aplinkoje, kur nešiojamojo kompiuterio ekrane buvo matomi vykstantys procesai.

Šiame darbe gauta autonominė transporto priemonė kuri pagal lazerinio skenerio duomenis 2D žemėlapyje pagal trumpiausią atstumą pasiekia nurodytas koordinates. Pirmiausiai toje vietoje, kur norima, kad robotas važinėtų po aplinką, sudaromas 2D žemėlapis naudojantis tuo pačiu lazeriniu skaneriu. Toliau nurodant robotui koordinates jis pagal gautą žemėlapį važiuoja trumpiausiu atstumu apvažiuodamas atsiradusias kliūtis.

Mainelis, Jonas. Research on Navigation of Autonomous robot with Laser Scanner: Master‘s degree / supervisor prof. Vytenis Sinkevičius. Panevezys Faculty of Technologies and Business, Kaunas University of Technology.

Research area and field: Control Technologies

Key words: mobile robot, laser scanner, 2D navigation

Panevėžys, 2017. 50 p., 44 pic.

## SUMMARY

Popularity of the mobile robots is growing all the time. In this project, the ways to determine position of the robot are analyzed. The navigation system of the Pioneer P3AT robot with the laser scanner is examined. ROS (Robot operating system) program are used for the investigation. The packages of the ROS program which are required for navigation were examined. The sensors are compiled and calibrated because of the more accurate values. Mobile robot needs to know its location for the reliable movement in the room. The reliability and accuracy of the position are the biggest problems of the mobile robots. Monte Carlo localization method is used in this project. This method was experimentally tested and the accuracy was examined. The influence of the amount of the elements for the final destination is examined. The variable number of the elements was measured using adaptive Monte Carlo algorithm. The environment is scanned using the laser scanner all the time, consequently even if there are some unknown obstacles in the way of the robot, it is able to plan the different rout. The effect for the movement of the robot is checked by changing navigation coefficients. The planning of the map and the rout is done in the ROS environment, where the active processes were visible in the screen of the laptop.

The autonomous transport vehicle which reaches the indicated coordinates in the shortest distance in the 2D map using the data of the laser scanner was received in this project. Firstly, the 2D map of the wanted environment for the movement of the robot is created by the same laser scanner. Further, the robot goes to the indicated coordinates in the shortest distance by avoiding the obstacles that appears using the 2D map that it has received.

# TURINYS

**ĮVADAS** ... **8**
**1. LITERATŪROS APŽVALGA** ... **9**
1.1. Autonominių robotų panaudojimas ... 9
1.2. Robotų orientavimosi galimybės erdvėje ... 10
1.2.1. Greičio ir nuvažiuoto atstumo matavimas ... 11
1.2.2. Orientavimas naudojant inercines sistemas ... 11
1.2.3. Įvairūs orientyrai ... 12
1.2.4. GPS ... 14
1.2.5. Signaliniai šviesos šaltiniai ... 15
1.1.5. Roboto orientavimasis pasitelkiant žemėlapius ... 16
1.2. Monte Karlo lokalizacijos algoritmas ... 17
1.3. Robotų operacinė sistema (ROS) ... 19
1.3.1. ROS sudėtis ... 19
1.3.2. ROS programų paketai ... 20
1.4. Skyriaus išvados ... 22
**2. METODOLOGINĖ DALIS** ... **23**
2.1. Naudojama aparatūra ir programinė įranga ... 23
2.1.1. Mobilusis robotas Pioneer P3-AT ... 23
2.1.2. ROS programinė įranga ... 24
2.1.3. SICK LMS-100 lazerinis skeneris ... 26
2.1.4. Odometrija ... 27
2.2. Komponentų koordinatės ... 28
2.4. Žemėlapio sudarymas SLAM algoritmui ... 30
2.5. Roboto navigacijos tikslumas naudojantis lazeriniu skeneriu ... 31
2.6. Skyriaus išvados ... 32
**3. TIRIAMOJI DALIS** ... **33**
3.1. Dalelių kiekio tyrimas ... 33
3.2. Roboto baterijos stebėjimas ... 38
3.3. Roboto maršruto planavimas ... 38
3.4. Skyriaus išvados ... 40
**IŠVADOS** ... **41**
**LITERATŪROS SĄRAŠAS** ... **42**
**PRIEDAI** ... **44**

# ĮVADAS

Technologijų vystymosi sparta pakeičia žmogaus darbo jėgą automatizuotomis sistemomis. Šių sistemų tobulėjimas padidina gamybos produktyvumą. Vis dažniau atsiranda sistemų, kurių veikimui nereikalingas žmogaus protas. Autonominis robotas šiais laikais yra itin sparčiai besivystanti technologijų sritis. Autonominio roboto pritaikymas gana platus. Yra daug naudojamų autonominių transporto priemonių tiek gamyklose, tiek gelbėjimo ar paieškos operacijose. Robotai tampa vis labiau integruoti į mūsų kasdieninį gyvenimą, jie pakeičia žmogų pavojingose darbo vietose. Šių robotų projektavimas pagrįstas saugumu, bei tiksliu orientavimusi dinamiškoje aplinkoje. Roboto sąveika su žmogumi reikalauja ypatingų saugumo garantijų. Robotas ne tik turi važinėti savo maršrutu, bet ir išvengti bei apvažiuoti įvairias kliūtis kelyje.

Autonominės transporto priemonės lokalizavimas ir žemėlapių sudarymas vienu metu paremtas SLAM (angl. *Simultaneous Localization and Mapping*) algoritmais. Pagrindinė šių algoritmų esmė palikus robotą nežinomoje vietoje, jis laisvai juda ir sukuria nuoseklų žemėlapį. Nors tai atrodo paprasta, iš tikro atsiranda rimta problema dėl to, kad realybėje jutikliai ir vykdikliai neveikia idealiai, atsiranda paklaidų. SLAM algoritmai plačiai naudojami paieškos ir gelbėjimo operacijose, povandeniniams žemėlapiams sudaryti, dujų tiekimo žemėlapiams ir kt. Pagrindinis autonominės transporto priemonės uždavinys – tai tikslus roboto judesys, kuriuo išvengiama įvairių kliūčių, atidžiai stebint aplinką. Yra įvairių autonominių robotų kelio planavimo (SLAM) algoritmų. Šių algoritmų esmė ne tik apskaičiuoti geriausią maršrutą, bet ir išvengti vienos ar kitos kliūties. Tam reikalinga tiksli aparatūra, kurią būtina susikalibruoti pagal atliekamos užduoties sudėtingumą. Daugeliu atveju reikalinga labai tiksli navigacijos sistema, ypač kai ta navigacija skirta sandėliavimo darbams.

**Tyrimo objektas** – Pioneer P3-AT mobilusis robotas.

**Tiriamojo projekto tikslas** – ištirti autonominio roboto navigacijos sistemos tikslumą naudojant lazerinį skanerį.

**Tiriamojo projekto uždaviniai**

1. Išsiaiškinti autonominio roboto valdymo galimybes.
2. Sukurti patalpos žemėlapį naudojant lazerinį skanerį.
3. Išbandyti roboto orientavimosi galimybes sudarytame žemėlapyje.
4. Patikrinti roboto veikimą dinaminėje aplinkoje.

**Tyrimo metodai –** mokslinės literatūros analizė, bandymų rezultatai.

# 1. LITERATŪROS APŽVALGA

## 1.1. Autonominių robotų panaudojimas

Mobilūs robotai turi didelį pritaikomumą, kur naudojamos bet kokios transporto priemonės arba mechaninės automatinės sistemos:

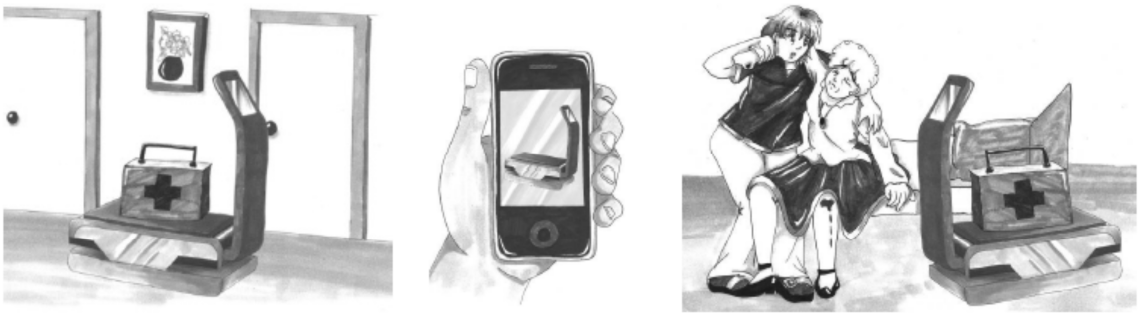
– Transportavimo sektoriuje yra daug vietų, kurias gali pakeisti autonominis robotas: robotas gali gabenti, tiek maisto vežimėlius, tiek vaistus, tiek šiukšles, paštą bei rūbus iš skalbyklų.

– Kitas sektorius – vandens tiekimas gyventojams. Robotas gali vežioti gėrimus įvairiose vietose (1.1 pav.).

1.1 pav. Autonominis robotas žmonėms vežioja vandenį [1]

– Robotai gali automatiškai valyti dideles zonas: prekybos centrus, oro uostus, įvairias gamybines patalpas, valyti stiklus, siurbti dulkes.

– Ligoninėse naktinėse pacientus gali aptarnauti mobilusis robotas. Autonominis robotas greičiau reaguotų nei žmogus (1.2 pav.).



1.2 pav. Mobilūs robotai medicinos skyriuje [1]

– Muziejų, ekskursijų, parodų ir įvairių vietų gidai.

– Agrokultūroje: vaisių ir daržovių rinkime, sodinime, trešime.

– Miškuose: valyti miškus, vykdyti gaisrų prevenciją, dalyvauti medžių kirtimuose.

– Pavojingos sritys: pavojingos aplinkos stebėjimui bei tvarkymui, vulkanuose, branduolinėse elektrinėse, naftos gamyklose.

– Pramogose: įvairūs interaktyvūs žaislai.
– Dujų ar naftos vamzdžių patikrinimui.
– Alyvos valymui.
– Statybose ir pastatų griovimuose.
– Kosmoso tyrinėjimuose.
– Nuotoliniam kosminių stočių tikrinimui.
– Kariuomenėje.
– Lėktuvų, traukinių patikrinimui.
– Seniems ir neįgaliems žmonėms. Mobilūs robotai gali atlikti ligonių reabilitaciją (1.3 pav.). Robotas gali turėti dideles dainų, poemų duomenų bazes.

1.3 pav. Mobilūs robotai skirti reabilitacijai [1]

### 1.2. Robotų orientavimosi galimybės erdvėje

Vienas iš svarbiausių uždavinių visoms autonominėms sistemoms yra įgyti žinių apie savo aplinką. Matavimams gauti naudojami įvairūs jutikliai. Kai kurie jutikliai skirti matuoti paprastas reikšmes, tokias kaip roboto elektronikos vidaus temperatūrą, variklių sukimosi greičius. Kiti, sudėtingesni jutikliai, gali būti naudojami gauti informaciją apie roboto aplinką ar net tiesiogiai išmatuoti robotų poziciją aplinkoje.

Pirmiausia, jutikliai, naudojami išgauti informaciją apie roboto aplinką. Kadangi mobilusis robotas juda aplink, jis dažnai susiduria su nenumatytomis aplinkos detalėmis, todėl reagavimas į objektus yra ypač svarbus.

Jutikliai klasifikuojami dvejomis funkcinėmis ašimis: vidaus / išorės ir pasyvūs / aktyvūs. Vidiniai jutikliai skirti gauti vidines sistemos (roboto) vertes, pavyzdžiui: variklio greitį, ratų apkrovas, roboto rankų sąnarių kampus, akumuliatoriaus įtampą. Išoriniai jutikliai gauna informaciją apie roboto aplinką, pavyzdžiui: atstumo matavimai, šviesos intensyvumas, garso amplitudės. Taigi, išoriniai jutiklių matavimai reikalingi siekiant gauti prasmingas aplinkos informacijos reikšmes. Pasyvieji jutikliai skirti matuoti aplinkos energiją.

Pasyvūs jutikliai yra temperatūros zondai, mikrofonai ir CCD ar CMOS kameros. Aktyvūs jutikliai skleidžia tam tikrą energijos rūšį į aplinką ir matuoja grįžtamos energijos pokytį. Tai ultragarsiniai jutikliai, lazeriniai skeneriai ir t.t. Kadangi aktyvieji davikliai labiau sąveikauja su aplinka, jie dažnai pasiekia puikias eksploatacines savybes. Tačiau aktyvūs jutikliai taip pat turi ir blogų savybių: grįžtamoji energija gali stipriai paveikti jutiklio matuojamas charakteristikas. Aktyvūs jutikliai labai jautrūs trukdžiams, kuriuos dažniausiai gauna iš aplinkos. Pavyzdžiui, skleidžiamus signalus aptinka kiti netoliese esantys robotai ar panašūs davikliai, kurie įtakoja gautus matavimus.

### 1.2.1. Greičio ir nuvažiuoto atstumo matavimas

Greičio ir nuvažiuoto atstumo matavimas arba odometrija yra daugiausiai naudojamas autonominiuose robotuose. Roboto greitis ir nuvažiuotas atstumas apskaičiuojamas enkoderiu. Transporto priemonės padėtis ir nuokrypis apskaičiuojamas iš ratų pasisukimo. Tačiau naudojantis odometru po kelių metrų rezultatai dreifuoja, atsiranda paklaidos [2]. Gaunamas trumpalaikis tikslumas, bet yra nebrangus ir gretai nuskaito padėtį. Greičio ir nuvažiuoto maršruto metodas yra paremtas judesio integravimu laike, todėl atsiranda vis didėjanti paklaida. Paklaida didėja, didėjant nuvažiuotam atstumui. Padidinus matavimų tikslumą, ši problema sumažėtų, tačiau vis tiek paklaida būtų didėjanti. Tyrėjai paklaidų sumažinimui naudoja Kalmano filtravimo algoritmus ir PID reguliavimą. Algoritmas pagal x ir y koordinates apskaičiuoja tam tikrą vektorių, suderina robotui kelią, maršrutą. Tačiau važiuojant kreive, robotas truputį nukrypsta nuo maršruto [3]. Orientuotis vien pagal odometrą negalima, todėl jis imamas kaip papildomas sistemos elementas. Paklaidos gali būti sisteminės arba nesisteminės. Sisteminės paklaidos atsiranda dėl roboto konstrukcinių netikslumų, tokių kaip skirtingų ratų, nesimetrinių dalių. Nesistemines paklaidas lemia kintantys parametrai, tokie kaip ratų praslydimai, netolygios apkrovos [4].

### 1.2.2. Orientavimas naudojant inercines sistemas

Inerciją matuoja pagreičio jutikliai. Dažniausiai tiesiaeigiam pagreičiui nustatyti naudojami akcelerometrai, o kampiniui pagreičiui – giroskopai. Kad akcelerometrai nustatytų padėtį, jų matavimus reikia integruoti du kartus. Ši sistema gera tuo, kad neveikia papildomi išoriniai poveikiai, nereikia keisti parametrų keičiant konstrukciją. Tačiau yra ir minusų – per ilgesnį laiko tarpą duomenys nebeatitinka tikrųjų. Inercinės sistemos netinkamos orientavimuisi ilgesniam laikui. Naudojant inercines sistemas su greičio ir nuvažiuoto atstumo sistemomis, paklaidos yra ženkliai sumažinamos.

Akcelerometro esmė – matuoti objekto pagreitį. Akcelerometras nustato objekto kryptį vektorine forma, kur galima apskaičiuoti vektoriaus dydį. Pagal akcelerometro duomenis galima nustatyti roboto pagreitėjimą, roboto ašies orientaciją, aptikti vibracijas bei smūgį.

Matuojant kampinį pagreitį galima sumažinti greičio ir atstumo metodu gaunamas paklaidas. Kampinis greitis matuojamas giroskopu. Pagal giroskopo duomenis galima panaikinti atsiradusius momentinius netikslumus, todėl jie plačiai naudojami autonominiuose robotuose.

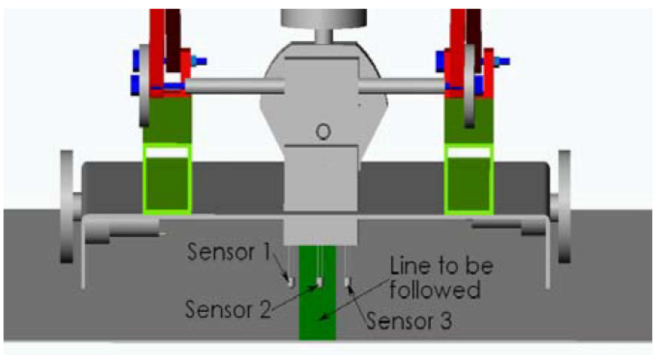
Sujungus giroskopo ir akcelerometro duomenis gaunama sistema, kurios pagalba yra nustatoma roboto padėtis bei greitis.

Magnetiniai kompasai skirti matuoti žemės magnetinį lauką. Tam, kad butų žinomas roboto pasisukimas nuo pradinio taško, naudojami magnetometrai. Jų pagalba galima nustatyti magnetinio lauko stiprį ir kryptį. Autonominių robotų navigacijos sistemose magnetinio komposo duomenys būna apskaičiuoti laipsniais nuo magnetinio šiaurės poliaus (azimutais).

### 1.2.3. Įvairūs orientyrai

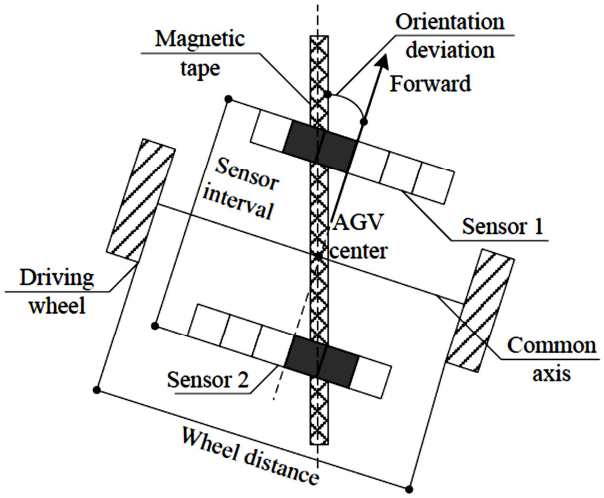
Įvairūs orientyrai – tai simboliai, kodai, ženklai, kurie priklijuojami ar kaip nors pritvirtinami prie sienų, grindų ar kitų objektų. Orientyrams atpažinti naudojamos kameros, skeneriai, ultragarsiniai jutikliai ir kt. Kiekvienas orientyras gali pasižymėti savita forma, spalva ar simboliu. Šio metodo esmė, kad robotas orientyrą išskirtų iš aplinkos. Šis metodas reikalauja į roboto atmintį įrašyti informaciją apie kiekvieną orientyrą. Ženklų nuskaitymas naudingas ne tik dėl to, kad robotas nustato savo padėtį aplinkoje, bet jis gali gauti daug daugiau informacijos, tokios kaip sumažinti ar padidinti greitį, pasisukti, įjungti papildomą funkciją ir t.t.

Autonominiai robotai gali orientuotis pagal juostą, priklijuota ant grindų. Juostos būna dviejų stilių: magnetinė arba kokios nors spalvos. Prie autonominio roboto pritvirtinamas jutiklis kuris seka juostą. Vienas iš pliusų yra tai, jog juostą galima perklijuoti ir taip pakeisti roboto maršrutą. Spalvotai juostai sekti naudojamas paprastas analoginis optinis jutiklis, kuris reaguoja į šviesos intensyvumą nuo juostos paviršiaus. Dažniausiai naudojami didelio intensyvumo raudonos spalvos LED fototranzistoriai. Jie turi išgaubtą lęšį, kurio paskirtis sumažinti matomumo ribą iki 5 laipsnių. Jutikliai turi būti atsparūs kambario šviesai. Norint išgauti didesnį tikslumą, reikia naudoti trijų jutiklių masyvą (1.4 pav.). Sukalibravus jutiklius, gaunamas autonominio roboto tikslus linijos sekimas su labai maža paklaida [5]. Šis roboto orientavimosi metodas gana paprastas ir pigus.




1.4 pav. Roboto sekimas spalvota juosta [5]

Spalvota juosta pigesnė, tačiau ji greičiau susidėvi ir tenka ją keisti, bei dėl įvairaus purvo ar kitų išorinių poveikių juosta uždengiama ar pažeidžiama. Lanksti magnetinė juosta pranašesnė, ji ne taip jautri užteršimui bei gali būti įmontuojama į grindis. Siekiant robotui išlaikyti skersinę ir išilginę poziciją, galima naudoti magnetinę juostą su keliais RFID skaitytuvais (1.5 pav.). Tokiu būdu robotas pastebės net menkiausią nukrypimą nuo ašies. Be to, pridėjus magnetines juostas šonuose, galima gauti puikias pozicionavimo bei patikimumo savybes [6].



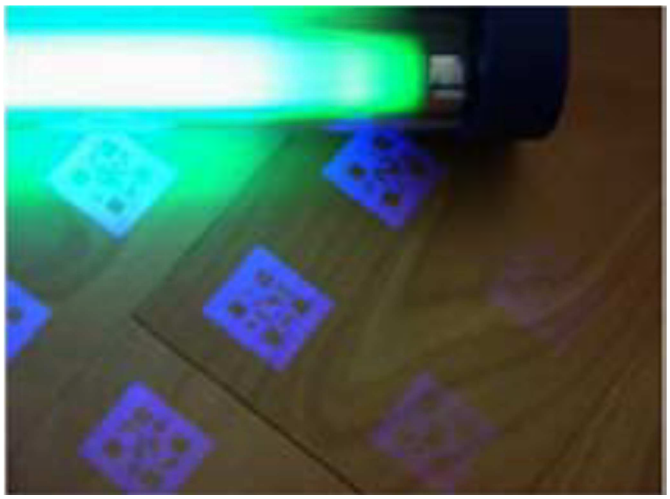

1.5 pav. Roboto orientavimasis naudojant RFID skaitytuvus [6]

Paprasčiausias būdas robotui nepasimesti iš maršruto yra sekti laidus, kurie paslėpti grindyse. Laidais perduodamas radijo signalas, o robotas jutiklio pagalba jį mato ir važiuoja pagal laidą.

Robotui orientuotis gali būti naudojami natūralūs orientyrai, kurie nėra specialiai įrengti navigacijai. Natūralūs objektai neturi keisti savo vietos, todėl reikalingi ir pagalbiniai orientyrai, kurie suteikia informacijos išskirtine savo forma ar simboliais. Šablonų atpažinimo sistemos tikslumas priklauso nuo atstumo. Robotas negali iš toli nustatyti savo tikslios pozicijos. Tikslumui palaikyti reikia, kad robotas orientyrą matytų aiškiai, nepakrypusį, nes tada atsiranda orientyro nuskaitymo klaidų. Tam tikslui robotui reikia gero apšvietimo bei tikslaus privažiavimo prie orientyro.


1.6 pav. Roboto orientavimasis naudojant brūkšninius kodus [7]

Dažnai orientyrams naudojami brūkšniniai kodai (1.6 pav.). Brūkšniniai kodai būna įvairių tipų. Jiems nuskaityti reikalingas brūkšninių kodų skaitytuvas. Pagal brūkšninius kodus ant sienų, grindų ar kitų objektų robotas ne tik gali orientuotis, bet gali gauti kitos naudingos informacijos, tokios kaip roboto greičio apribojimus, informaciją apie produktus, sandėliavimo zonas. Be to galimi ir nematomi brūkšniniai kodai. Jie pasirodo tik apšvietus UV lempa (1.7 pav.). Kiekvienas brūkšninis kodas gali turėti masyvą informacijos, tikslias x ir y koordinates [8]. Kai robotas nuskaito brūkšninį kodą, jis žino savo santykinį kampą su konkrečiu brūkšniniu kodu ir santykinį atstumą iki kodo. Brūkšniniai kodai yra pigi orientavimosi sistema, nes kainuoja tik rašalas, o jei jie dar ir nematomi tai nepakenkia estetiniam patalpos vaizdui.

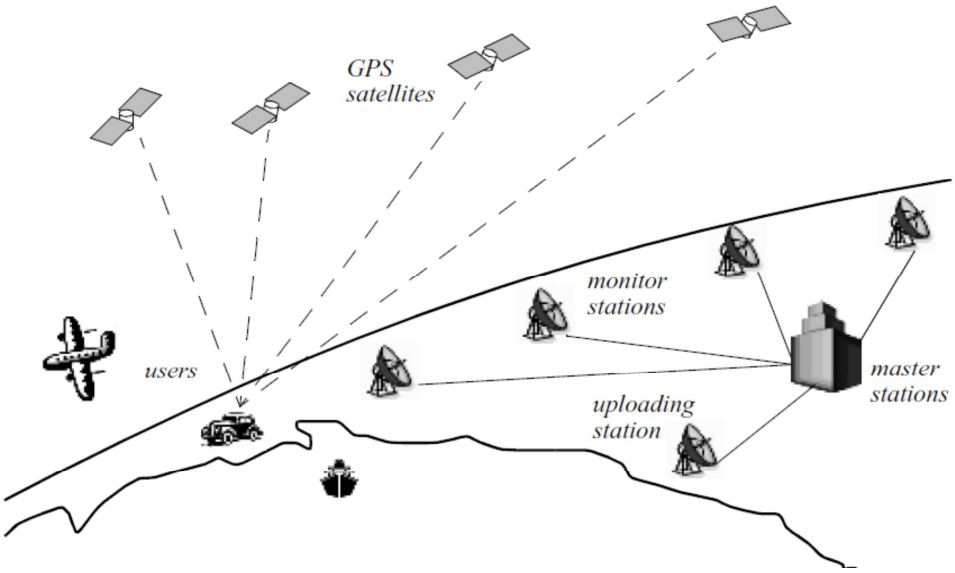


1.7 pav. Nematomų brūkšninių kodų žymėjimas [8]

Kai naudojamas robotų orientavimasis pagal orientyrus, robotuose įmontuoti mikrovaldikliai yra per silpni apdoroti daug informacijos. Naudojantis roboto realaus judėjimo navigacija pagal orientyrus, reikalingas galingas kompiuteris. Orientyrai turi gerai matytis, būti orientuoti į roboto jutiklio pusę. Be to, reikalingas geras apšvietimas.

### 1.2.4. GPS

GPS (angl. Global Positioning System) – Visuotinė padėties nustatymo sistema, arba Globali Pozicionavimo Sistema, išduoda objekto koordinates bet kurioje vietoje, kur yra ryšys su palydovais (1. 8 pav.). Globali padėties nustatymo sistema (GPS) iš pradžių buvo sukurta kariniams tikslams, tačiau dabar laisvai prieinami visiems. GPS palydovai sinchronizuoti taip, kad jų signalai būtų siunčiami tuo pačiu metu. Visi GPS palydovai radio signalais į Žemę perduoda savo koordinates, laiką ir identifikavimo kodą. GPS imtuvas skaitydamas dviejų ar daugiau palydovų informacijos gavimo laikus nustato santykinį atstumą iki palydovo.



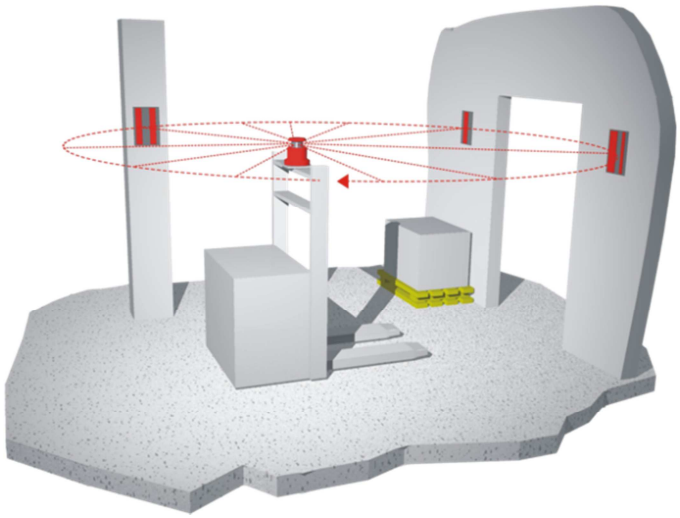

1.8 pav. Visuotinė vietos nustatymo sistema [9]

Teoriškai tokiai navigacijai užtektų tik trijų duomenų taškų, tačiau laiko tikslumui reikalingi mažiausiai keturi, nes laiko intervalai skaičiuojami nanosekundėmis. Taigi, nors trys palydovai tiksliai nustato poziciją trijose ašyse, GPS imtuvui reikia keturių palydovų. Tai reiškia, kad sėkmingai būtų nustatoma vieta, reikia tiesioginio ryšio su palydovais. Taigi, uždarose patalpose, namų kvartaluose su aukštais pastatais arba miške sunku patikimai gauti informaciją iš keturių palydovų. Be to, jų paklaida ganėtinai didelė. Dėl šių priežasčių GPS nėra populiarus jutiklis autonominiams robotams.

### 1.2.5. Signaliniai šviesos šaltiniai

Signalinių šviesos šaltinių sistemos yra sparčiai naudojamos lėktuvų ir laivų navigacijoje, taip pat ir autonominiams robotams (1.9 pav.). Švyturiai yra tiksli, patikima ir gana pigi orientavimosi priemonė. Naudojant šią sistema galima išgauti didelius duomenų nuskaitymo dažnius ir patikimumą, tačiau išauga jos priežiūros kaina. Šviesos šaltinių išdėstymas yra labai svarbus, nes jie neturi būti uždengti kitų objektų.

Signalinių švyturių trišalės sistemos veikimas pagrįstas roboto nuvažiuotu atstumu nuo matomo signalinio švyturio. Kad tinkamai veiktų trišalė sistema, ant roboto turi būti įrengtas imtuvas. Jis turi matyti tris ar daugiau signalinių švyturių. Tam tikslui gali būti naudojami 360 laipsnių skaitytuvai (9 pav.). Orientyrai gali pažymėti įvairius atstumus bei, kur reikia tikslumo, bazuoti robotą reikiamoje pozicijoje.

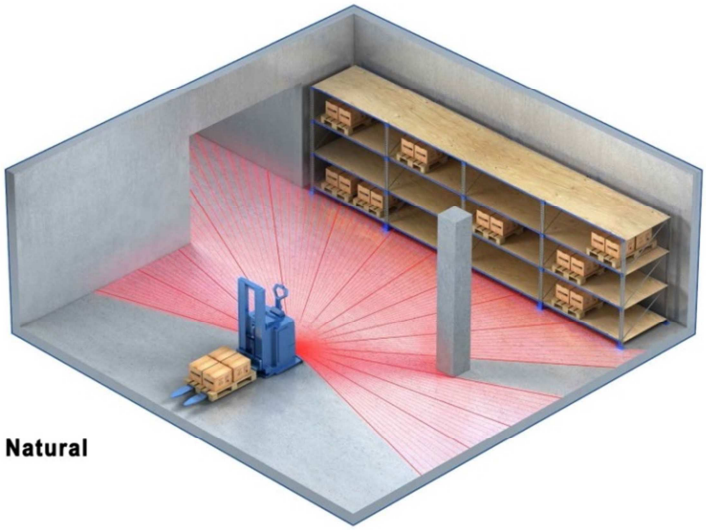


1.9 pav. Roboto orientavimasis naudojant 360 laipsnių skaitytuvą [10]

Gali būti ant sienų naudojami atšvaitai, o ant roboto siųstuvas. Pagal gautą informaciją, algoritmu apskaičiuojama roboto buvimo vietą bei padėtį. Šio metodo trukumas, yra tai kad švyturių skleidžiama šviesa turi sklisti kūgio formos signalais, tam kad būtų matoma iš visų pusių.

### 1.1.5. Roboto orientavimasis pasitelkiant žemėlapius

Roboto orientavimasis naudojant žemėlapius, tai navigacijos sistema, kur roboto atmintyje yra įkelti vietinės aplinkos žemėlapiai. Turimas žemėlapis lyginamas su esamu. Jei turimas žemėlapis sutampa su esamu, robotas pagal matomus objektus apskaičiuoja savo vietą žemėlapyje ir nustato judėjimo kryptį. Žemėlapis būna nubraižytas tam tikromis programomis, CAD brėžiniais arba jutiklių duomenų bazėmis. Pritaikant tam tikras programas, robotas per pirmą važiavimą pats gali nusibraižyti sau žemėlapį. Šio metodo trūkumas tai, kad aplinka turi kuo mažiau keistis, tam kad sutaptų jutiklių duomenys. Šis metodas dažniausiai naudojamas tik nedidelėje, uždaroje aplinkoje [11]. Pozicijai nustatyti naudojant žemėlapius yra du skirtingi pradžios taškai: vienas – tai įkeltas sukurtas žemėlapis, kitas – robotas pats turi nusibraižyti žemėlapį tyrinėdamas. Robotas pradeda tyrinėti aplinką neturėdamas jokių pradinių žinių. Robotas juda taip, kad per mažiausią laiko tarpą apvažinėtų didžiausią plotą. Visą jo tyrinėjimą lemia jutiklių pasirinkimas. Naudojant lazerinį skenerį robotas nuo vieno taško iki kito taško braižo 2D žemėlapį (1.10 pav.). Bandant robotą uždaroje aplinkoje kur daug kliūčių, gaunasi ribotas roboto greitis, didesnėje erdvėje robotas sugeba judėti truputi greičiau [12]. Norint, kad robotas spėtų reaguoti į įvairias kliūtis ar objektus, reikalinga apriboti roboto greitį. Naudojantis lazeriniu skeneriu, duomenų sparta, dreifas, įvairios optinės savybės, objektų bei lazerio nuolydžio kampai yra svarbiausi parametrai roboto orientavimo kokybei.




1.10 pav. Roboto orientavimasis naudojant skenerį [9]

Autonominiai robotai naudoja įvairius algoritmus, kurių pagalba yra apdorojama gauta informacija iš lazerinių skenerių, tam kad gautų tikslų ir patikimą aplinkos žemėlapį (7 pav.). Svarbiausias žemėlapių metodo naudojimo tikslas atpažinti braižomą žemėlapį išsaugotame žemėlapyje. Tik taip robotas gali orientuotis aplinkoje. Šis metodas gana plačiai naudojamas sandėliuose, nes žemėlapiai nėra dideli ir robotas gali saugiai veikti dinaminėje aplinkoje.

## 1.2. Monte Karlo lokalizacijos algoritmas

Monte Karlo lokalizacijos algoritmas paremtas dalelių filtro pagrindu. Tai navigacijos sistema, kuri orientuojasi pasitelkiant žemėlapius. Žemėlapį sudaro didelis kiekis dalelių (taškų), kur kiekviena dalelė yra galima roboto pozicija žemėlapyje. Dalelių filtras inicializuoja žemėlapyje daug skirtingų taškų, kuriuose robotas gali atsistoti.

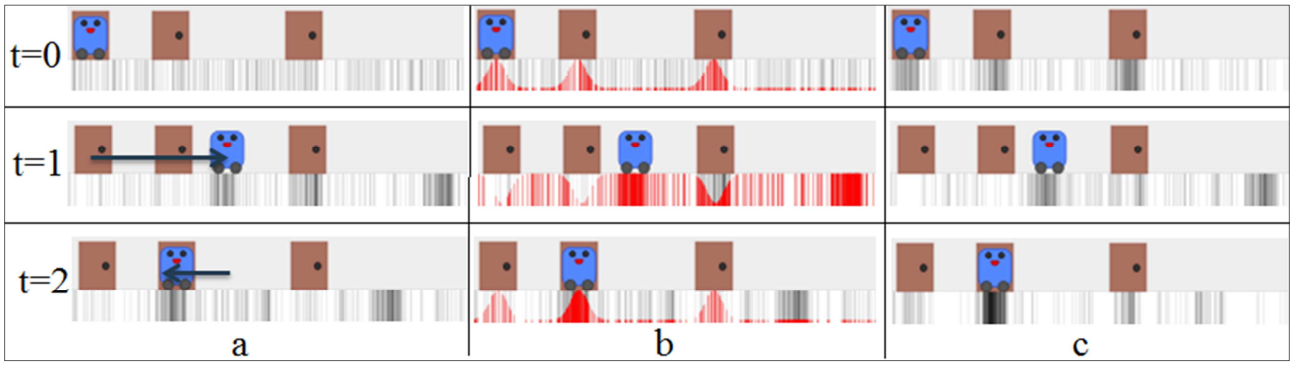
Dalelių filtras paremtas dviejų daviklių skaičiavimais: tai nuvažiuoto atstumo ir greičio matavimo jutiklio – enkoderio bei lazerinio skenerio. Roboto pozicija (vieta ir kryptis) aplinkoje yra aprašoma atsitiktiniais kintamaisiais $X_t$. Kiekvienas kintamasis $X_t$ susijęs su roboto judėjimo matavimu $U_t$. Pagal odometro reikšmes, gaunamas judėjimas, kur robotas juda nuo $X_t$ iki $X_{t+\Delta t}$ . Lazerinio skenerio duomenimis gaunami aplinkos matavimo koeficientai $Z_t$. Taip dalelių filtras vertindamas prieš tai buvusiais užimtas dalelių reikšmes sudaro kintamuosius *bel*($X_t$), kur pagal Bajeso lygtis (Bajeso filtrą *angl. Bayesian filter*) remiantis minėtais duomenimis gaunamas dalelių filtro užimtumas.

$$X_t := \left\{X_t^{[1]}, X_t^{[2]}, \cdots, X_t^{[j]}\right\}, \tag{1.1}$$

Kiekviena dalelė $X_t^{[j]}$, kur j = (1,…,m) yra konkreti momentinė reikšmė per laiką t. Tai yra hipotezė, kur gali būti užimta pozicija per laiką t, kur m žymi dalelių skaičių nustatytoje reikšmėje

$X_t$. Taip pat, kiekviena dalelė turi savo svorį (angl. *weight*), kur atitinka tikimybę pagal tankio funkciją.

Monte Karlo lokalizacijos algoritmo pavyzdys pateiktas 1.11 pav. Robotas juda koridoriumi viena ašimi. Koridoriuje yra trejos durys, bet robotas nežino, kurioje vietoje jis yra. Algoritme yra nustatytas tam tikras dalelių kiekis. Roboto galimai užimtas daleles atvaizduoja po paveikslėliais esantys brūkšniai. Algoritmą sudaro trys etapai: a – roboto judesys, b – skenerio duomenų atnaujinimas, c – vietos tikslinimas. Pradžioje (t=0, a) robotas nejuda, jis nežino kur yra. Robotas gali būti bet kuriame taške, nors fiziškai jis yra prie pirmų durų. Antrame etape (t=0, b) pavaizduotas skenerio duomenų gavimas. Pagal gautus duomenis robotas aptinka duris. Kadangi žemėlapyje yra trejos durys, tai jis gali būtų prie vienų iš jų. Robotas priskiria kiekvienai dalelei tam tikrą koeficientą (*angl. weight*), kuris nurodo dalelių užimtumą. Trečiajame etape (t=0, c) vietos informacija atnaujinama. Pagal dalelių užimtumo koeficientus robotas sugeneruoja naujas dalelių grupes, kur robotas gali stovėti. Pagal dabartinius duomenis, robotas žino, kad stovi prie vienų iš durų. Robotui pajudėjus tam tikrą atstumą į dešinę (t=1, a), pasislenka ir dalelės, nes dalelės jo galimai užimama vieta. Robotas fiziškai dabar yra tarp antrų ir trečių durų. Po skenerio duomenų gavimo (t=1, b) robotas supranta, kad stovi ne prie durų. Robotas priskiria dalelėms svertinius koeficientus. Kadangi robotas dabar stovi ne prie durų, jis panaikina daleles, kurios nesutinka su gautu aplinkos matavimu. Trečiame etape (t=1, c) robotas sugeneruoja naujas dalelių grupes, kurios parodo, kad galimos dvi roboto buvimo vietos. Robotas pajuda į kairę ir sustoja ties durimis (t=2, a). Fiziškai jis yra ties antromis durimis. Nusiskenavęs aplinką robotas aptinka duris (t=2, b). Vėl įvertinami dalelių svertiniai koeficientai. Tada robotas generuoja naują dalelių grupę pagal gautas ankstesnes daleles (t=2, c). Po šio veiksmo robotas sėkmingai nustato savo padėtį žemėlapyje, nes liko tik viena dalelių grupė.




1.11 pav. Monte Karlo lokalizacijos algoritmas

## 1.3. Robotų operacinė sistema (ROS)

ROS yra atviro kodo robotų operacinė sistema, kurią naudojant galima sukurti programas įvairiems robotams. Ši robotų operacinė sistema standartizuota API protokolais ir turi platų komunikacijų ryšį įvairiems komponentams[14].

Šiuo metu ROS programinė įranga turi žemėlapių, navigacijos, objektų atpažinimo, įvairių įrankių judesių planavimo, modeliavimo bei įrenginių mokymosi pavyzdžius. ROS operacinė sistema lanksti, nes suteikia galimybę programuoti trimis programavimo kalbomis: C++, Python ir Lisp. ROS operacinės sistemos įrankiai palengvina darbą su įvairiomis robotų sistemomis. Operacinės sistemos įrankiai leidžia imituoti bei valdyti įvairias robotų komponentų dalis ir vizualizuoti visus judesius kompiuterio ekrane.

### 1.3.1. ROS sudėtis

Yra trys skirtingos ROS lygių sistemos. Pirmasis failų sistemos lygis (angl. *Filesystem Level*) yra skirtingų failų organizavimas kompiuteryje. Antrasis kompiuterinės grafikos lygis (angl. *Computation Graph Level*), kuriame keitimasis resursais vyksta tiesiogiai tarp duomenų. Visa tai atsispindi grafikuose. Trečiasis Bendruomenės lygis (angl. *Community Level*) naudojamas dalintis skirtingais programinės įrangos failais.

Failų sistemos lygis iš esmės yra padalintas į du svarbius vienetus: paketus (angl. *packages*) ir paketų blokus (angl. *stacks*):

Paketai yra mažiausi ir pagrindiniai vienetai ROS failų sistemoje. Tai įvairūs kodai, bibliotekos, vykdomosios programos, mazgai (angl. *nodes*), nustatymų failai ir paleidimo failai (angl. *launch files*). Paketų blokai yra grupė tarpusavyje susietų paketų, kurie pasižymi aukšto lygio funkcionalumu.

Kitas svarbus failas ROS sistemoje yra „Manifest“. Jis gali priklausyti tiek pakuotei, tiek pakuočių blokui, ir teikia bendrą informaciją apie konkretų paketą arba paketų bloką. Pavyzdžiui, trumpas aprašymas, ką jis atlieka, jo licencijos tipai ir priklausomybė su kitais paketais. Paketų priklausomybė yra svarbus aspektas, nes daugelis paketų yra priklausomi vieni nuo kitų, todėl jie privalo būti įdiegti.

Kompiuterinės grafikos lygis paaiškina infostruktūrą apie ROS komponentų komunikavimą. Šis lygis remiasi P2P tinklo procesais – mazgais ir yra sudarytas iš įvairių vienetų:

Mazgai – tai ROS sistemoje vykdomosios programos, kurios atlieka skaičiavimo funkciją. Jie gali būti sujungti su kitais mazgais naudojant pranešimus (angl. *messages*) arba paslaugas (angl. *services*).

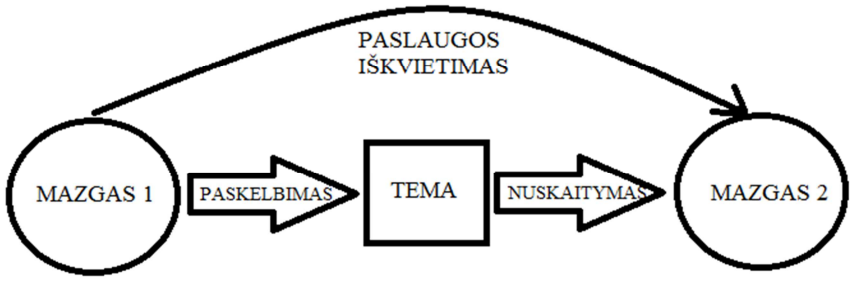
Pranešimai sudaro primityvų duomenų tipą, pavyzdžiui, sveikieji skaičiai, loginės reikšmės ar matricos, kurių duomenys perduodami tarp mazgų.

Temos (angl. *topics*) – tai mazgai, kurie perduoda ir gauna pranešimus. Šie mazgai yra skelbiami su specialiomis vardinėmis temomis. Taigi, mazgai gali skelbti arba laukti specialios temos apie norimą pranešimą. Daugelis mazgų gali užsisakyti tą pačią temą bei gauti pranešimus.

Naudojantis paslaugomis bet kurie du taškai gali komunikuoti naudojant prašymo – atsakymo (angl. *request – responce*) metodą. Kai mazgas nori iškviesti paslaugą , jis turi kreiptis į ją ir laukti atsakymo. Kai paslaugos mazgas gauna kreipimąsi, jis pateikia atsakymą su reikalaujamais duomenimis. Skirtingai nei komunikavime tarp temų, čia galimas komunikavimas tik tarp dviejų taškų.

Valdantysis (angl. *master*) lygis paruošia ROS tinklą darbui ir yra pirmasis procesas, kuris turi būti paleistas, kai naudojamasi ROS. Jo funkcija registruoti visus mazgus (angl. *nodes*), temas (angl. *topics*) ir paslaugas (angl. *services*).

1.12 paveiksle parodytas ROS kompiuterinės grafikos lygis, kur yra paleisti du mazgai. Pirmas mazgas siunčia pranešimą į temą, o antras mazgas laukia pranešimo iš temos. Kitas komunikavimo būdas yra išsikviesti paslaugą į antrą mazgą. Mazgas 1 iškviečia paslaugą, o antras mazgas pateikia norimų duomenų atsakymą.

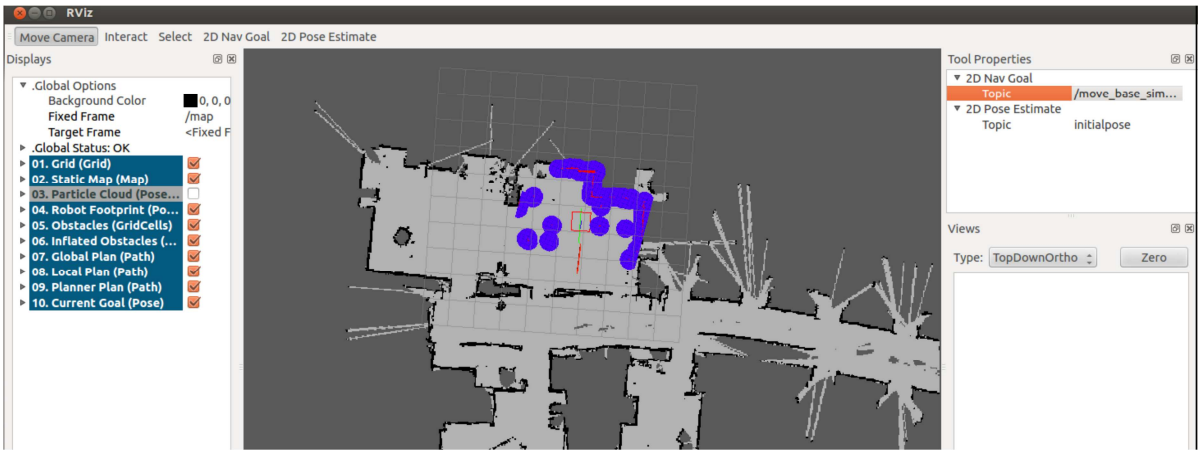



1.12 pav. ROS komunikavimo infostruktūra

Svarbiausias ROS bendruomenės bruožas yra dalinimasis programine įranga bei kodais didelėse ROS robotų bendruomenėse. Yra įvairių būdų kaip gauti ROS failus. Yra daug saugyklų (*svn* arba *github*), kur galima rasti robotų programinės įrangos kodus. Pagrindinis informacijos šaltinis yra ROS viki žodynas (ROS Wiki), kur galima rasti įvairios informacijos ir pamokas apie turimus paketus. Taip pat yra ROS vartotojų interneto puslapis (answers.ros.org), kur vartotojai dalinasi įvairiais klausimais, atsakymais bei komentarais.

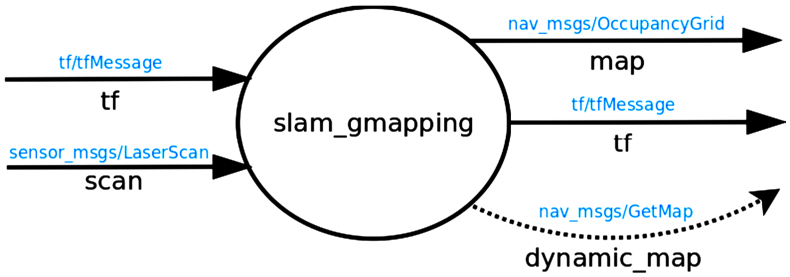
### 1.3.2. ROS programų paketai

ROS programos RVIZ paketas yra galingas vizualizacijos įrankis, kuris skirtas atvaizduoti žemėlapius, robotus, objektus, lazerinio skenerio duomenis, kameros vaizdus, kitus jutiklius ir vykdiklius. RVIZ programos pavyzdys pateiktas 1.13 pav. Kairėje pusėje matomi visi pasirinkti elementai, kurie bus rodomi pagrindiniame lange. Pagrindiniame lange matosi sukurtas žemėlapis ir roboto vieta jame. Taip pat violetine spalva matoma lazerinio skenerio matomi objektai.



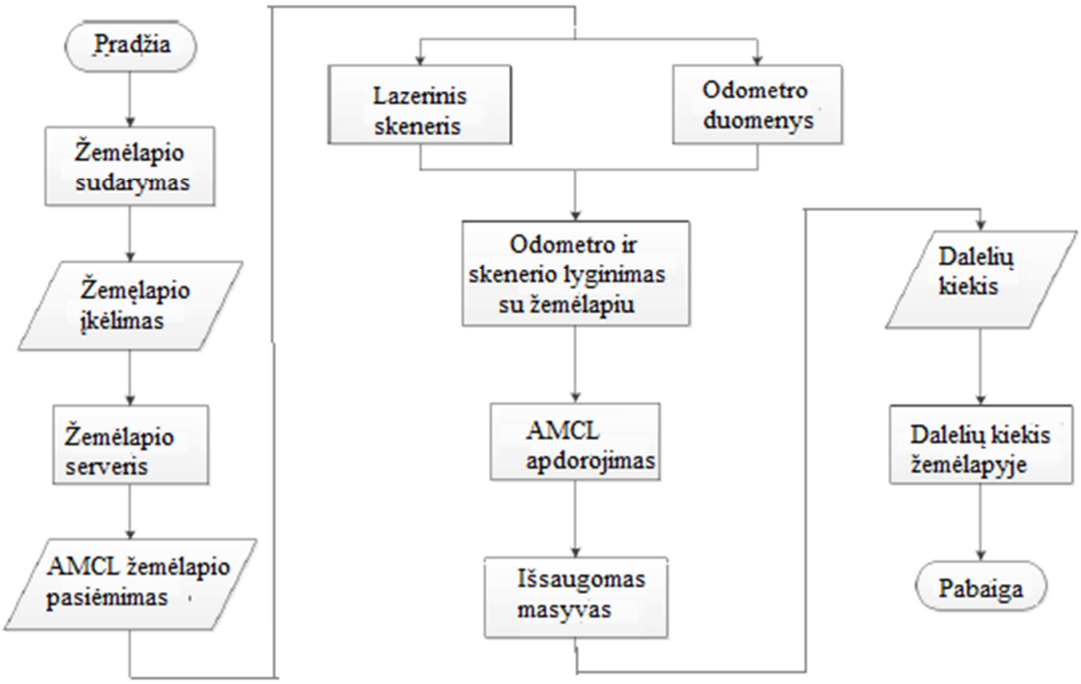

1.13 RVIZ vizualizacijos programa

ROS operacinė sistema turi *GMapping* paketą. Šis paketas leidžia naudoti SLAM algoritmą, naudojantis lazerio, nuvažiuoto atstumo ir greičio duomenimis. *GMapping* paketo programa sukuria roboto 2D užimtumo tinklelį žemėlapyje ir apskaičiuoja savo poziciją. Nustatymuose turi būti pažymėta, kad tai žemėlapis arba dinaminis žemėlapis (*map*), o pagrindiniai odometro ir lazerinio skenerio (*LaserScan*) duomenys turi būti transformuojami į reikiamas koordinates (tf). Odometro reikšmės nėra tiesiogiai gaunamos, todėl jų nėra pavaizduota 1.14 paveiksle.




1.14 pav. SLAM algoritmo ir jutiklių komunikavimo grafikas

ROS operacinėje sistemoje yra Monte Karlo algoritmas (1.15 pav.). Pirmiausiai robotas turi susidaryti patalpos žemėlapį, nes lokalizacijai jau reikia turėti žemėlapį, pagal kurį lyginami duomenys. Sudarytas žemėlapis įkeliamas į žemėlapio serverį. Tada Adaptyvusis Monte Karlo lokalizacijos algoritmas pasiima sukurtą žemėlapį. Robotas matuoja lazerinio skenerio ir odometro reikšmes, pagal jas skaičiuojama vieta žemėlapyje. Tada gautas reikšmes palygina su turimu žemėlapiu. Jei rezultatas sutampa, AMCL algoritmas apdoroja informaciją ir išduoda užimamų dalelių masyvą. Tas dalelių masyvas išsaugomas atmintyje ir rezultatą galima pamatyti programoje.




1.15 pav. Monte Karlo algoritmas ROS programoje

### 1.4. Skyriaus išvados

1. Ratų enkoderiai pameta reikšmes dėl ratų prasisukimo, todėl atsiranda didelė matavimų paklaida.
2. Lazeriniai skeneriai reikalauja galingo kompiuterio, kuris galėtų apdoroti didelį informacijos srautą bei skenavimas riboja roboto judėjimo greitį.
3. Nei vienas jutiklis negali surinkti tiek informacijos, kad užtektų autonominiam robotui orientuotis aplinkoje, todėl dažniausiai naudojamos kelios orientavimosi sistemos viename robote.
4. Naudojantis odometro ir lazerinio skenerio duomenimis, galima sukurti navigacijos sistemą pagal Monte Karlo lokalizacijos algoritmą.
5. Robotų operacinė sistema „ROS“ – galinga programa robotikoje. Su „ROS“ programa galima sukomutuoti reikalingus komponentus navigacijos sistemai. ROS programa galima sukurti Patalpos žemėlapius naudojantis lazeriniu skeneriu.

# 2. METODOLOGINĖ DALIS

## 2.1. Naudojama aparatūra ir programinė įranga

### 2.1.1. Mobilusis robotas Pioneer P3-AT

Mobilūs robotai yra tie robotai, kurie gali laisvai judėti iš vienos vietos į kitą. Mobilumas robotui suteikia galimybę daug laksčiau atlikti įvairias užduotis. Dauguma robotų yra naudojami pavojingose vietose, kur gali būti pakenkta gyvam organizmui. Autonominis robotas geba aptikti objektus jutiklių pagalba be nuotolinio valdymo įtaisų. Kadangi mobilūs robotai dažniausiai važinėja vienoje plokštumoje, ant žemės, jie žemėlapyje užima tris kintamuosius. Tai yra, X,Y koordinatės, kurios nurodo tašką žemėlapyje ir roboto orientaciją jame, kampas θ.

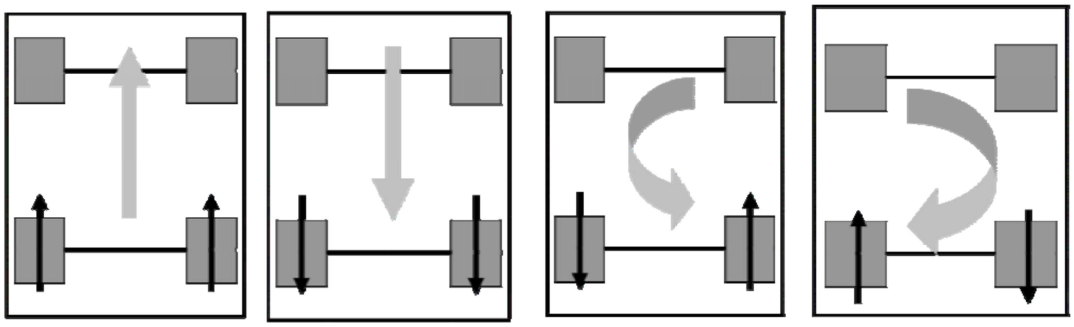
Tyrimui panaudotas Pioneer P3-AT mobilusis robotas (2.1 pav.). Tai universalus, keturiais ratais varomas mobilus robotas. Jis yra nedidelis, keturių variklių, kurio pasisukimui reikia, kad ratai suktųsi į priešingas puses. Šis robotas yra pakankamai galingas visureigis laboratorinėje aplinkoje. Paprasčiausiame rinkinyje jis turi bateriją, avarinio sustabdymo mygtuką, ratų enkoderius ir mikrovaldiklį su Acros operacine sistema. Pagrindinę robotų įrangos informaciją teikia ARIA programa. ARIA yra C++ biblioteka, kuri suteikia teisę kontroliuoti ir gauti duomenis iš visų MobileRobots platformų, taip pat ir jų priedų. Ši programa yra atviro kodo programa [15]. Vartotojo sąsaja paprasta, ji naudoja komandos eilutes robotui valdyti. Pioneer P3-AT robotą galima naudoti su ROS (angl. *Robot Operating System*) operacine sistema. ROS programa turi specialią biblioteką ROSARIA, kuri skirta roboto ir kompiuterio komunikacijai.


2.1 pav. Pioneer P3-AT

Pioneer P3-AT specifikacija:

- matmenys 508 mm ilgio, 497 mm pločio ir 277 mm aukščio;
- maksimaliai robotas gali būti pakrautas 12 kg, jeigu važinėjama lygiomis grindimis;
- jeigu važinėjama ant žolės ar purvo, galima jam užkrauti 10 kg krovinį;
- maksimalus greitis 0,7 m/s , maksimalus sukimosi greitis 140 laips/s;
- turi tris baterijas, su jomis robotas gali dirbti 2–3 valandas.

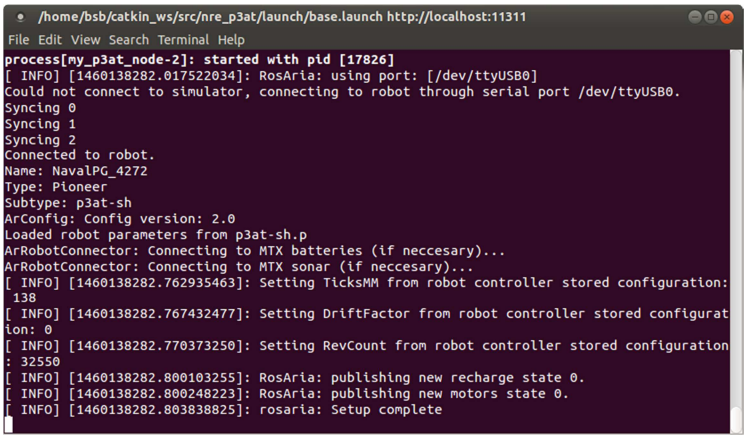
Robotui pajudėti į priekį reikia sukti visus keturis ratus į priekį, Norint važiuoti atgal, reikia sukti visus keturis ratus atgal. Kad robotas pasisuktų, vienos pusės ratai turi suktis į vieną puse, kitos pusės ratai į kitą pusę (2.2 pav.).



2.2 pav. Pioneer P3-AT judėjimo galimybės.

### 2.1.2. ROS programinė įranga

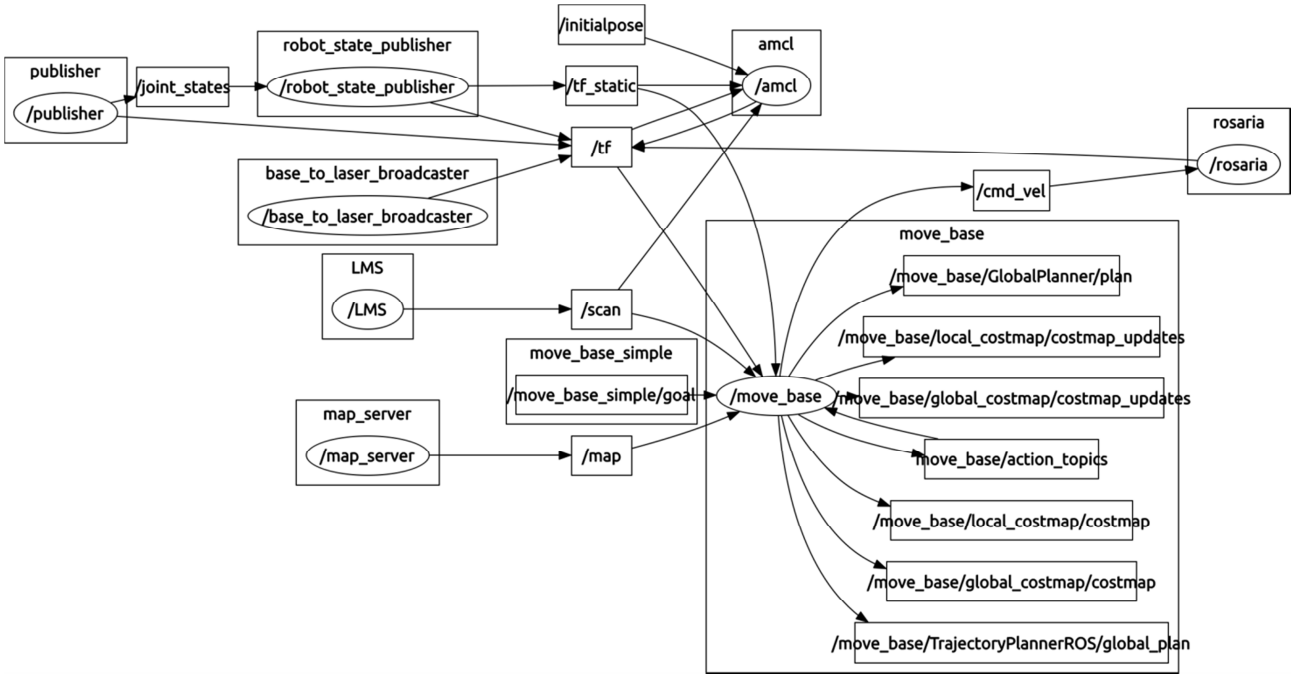
Tyrimui panaudota ROS operacinė sistema. ROS yra atviro kodo robotų operacinė sistema, kurią naudojant galima sukurti programas įvairiems robotams. „ROS Indigo“ programa veikia su „UBUNTU 14.04“ operacine programa. ROS programa paleidžiama per komandos eilutę – „roscore“. Komandos eilute – „rosrun rosaria RosAria _port:=/dev/ttyUSB0“ paleidžiame Pioneer P3-AT programos paketą. Kadangi robotas su kompiuteriu sujungtas per RS232 konverterį (2.1 pav.), todėl reikia nurodyti kompiuterio USB prievado numerį. Jeigu pasirinktas teisingas nuoseklusis prievadas (angl. *port*), tai matysis terminale 2.3 pav.




2.3 pav. Pioneer roboto susijungimas su kompiuteriu.

Tačiau „UBUNTU“ operacinei sistemai reikia suteikti vartotojui leidimus naudotis prievadu, naudojantis „sudo adduser USER dialout“ komandos eilute (kur USER yra jūsų kompiuterio vardas) suteikiamos vartotojui naudotis prievadu.

ROS programa tinkama sujungti visus navigacijai reikalingus komponentus bei apdoroti jų informaciją. Pagal 2.4 paveiksle matome visus veikiančius programos paketus. „ Publisher“ praneša apie roboto modelio vietą ir jo jungčių vietą. Panaudojus „tf“ paketus, sutvarkomos kiekvieno elemento koordinačių ašys. „Amcl“ – tai adaptyvusis Monte Karlo lokalizacijos algoritmas. Jis gauna duomenis iš „LMS“ lazerinio skenerio. „Map_server“ saugo jau turimą patalpos žemėlapį. „Movie_base“ kontroliuoja visus roboto judesius bei atlieka skaičiavimus roboto maršrutų sudarymui. „Rosaria“ – tai mobilaus roboto Pioneer P3-AT paketas. Viską sujungus, gaunama navigacijos sistema.




2.4 pav. ROS programos navigacijos komponentai

Robotui orientuojantis aplinkoje, reikia žinoti savo gabaritus. Programoje reikia suvesti roboto matmenis, kad būtų apskaičiuojama erdvė, kurioje robotas gali apsisukti. Šie duomenys suvedami masyvu: $[x_0,y_0]$, $[x_1,y_1]$ , $[x_2,y_2]$, ... Nulinės koordinatės skaičiuojasi nuo roboto vidurio. Pioneer P3-AT matmenys : *[0.254, -0.230], [-0.254, -0,230], [-0.254, 0.230], [0,254, 0,230]* .

Robotui užstrigus ar pametus vietą, navigacija turi atkūrimo paketą. ROS programoje yra dviejų tipų atkūrimo taškai. Vienas būdas: panaikinti gautą poziciją ir sulyginti vietinio ir globalaus žemėlapio reikšmes. Jei reikšmės atitinka, tai užskaitomas kaip sėkmingas pozicijos nustatymas. Jei šis būdas nesuveikia, robotas apsisuka aplink savo ašį 360° ir nuskenuodamas žemėlapį ieško sutapimo globaliame žemėlapyje, kurį turi savo atmintyje.

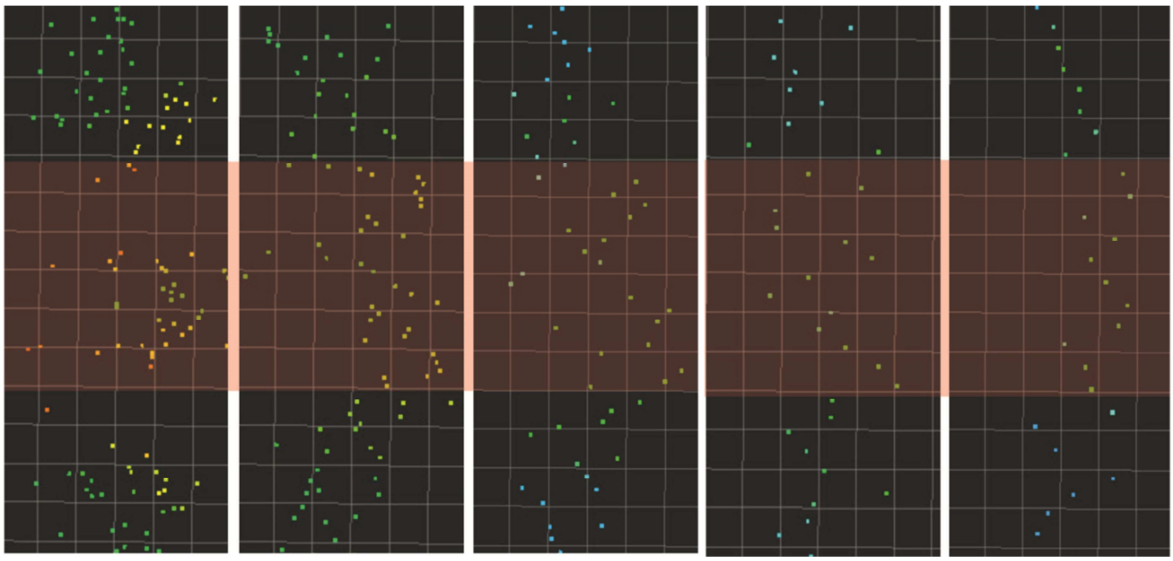
2.1.3. SICK LMS-100 lazerinis skeneris

Tyrimui atlikti naudojamas 2D SICK LMS-100 lazerinis skeneris. Šis skeneris skirtas tam tikroms zonoms, patalpoms stebėti, objektams matuoti, aptikti ir pozicijai nustatyti. Skeneris veikia lazerio šviesos impulsais: yra impulsinis lazeris, matuojantis laiką, per kurį šviesa grįžta iki jo nuo objekto [16]. Atstumas apskaičiuojamas panaudojant šviesos sklidimo laiką nuo lazerio impulso pradžios iki atspindžio impulso atsiradimo fotoimtuve. Impulsinio lazerio šviesą atspindi vidinis veidrodis, kuris sukiojasi 270° kampu su 0,25° rezoliucija. Taip gaunamas 2D kliūčių, atspindinčių lazerio spindulį, vaizdas. Šis jutiklis turi 1 081 matavimo reikšmę. Skenavimo dažnis apie 25–50 Hz. Maksimalus atstumas iki kliūties 20 m, bet dėl mažo atspindžio nuo kartono ar juodos spalvos šis atstumas sumažėja iki 18 m. Matavimo tikslumas yra +/- 30 mm. Jutiklis jungiamas internetiniu kabeliu. Šio lazerinio skenerio biblioteką taip pat turi ROS operacinė sistema. Skenerio programos paketas „lms1xx“ skirtas sujungti skenerį su ROS sistema. Pasitelkus lazerinį skenerį bus gaunamas 2D aplinkos vaizdas tam tikrame aukštyje.

Šis lazerinis skeneris geriausiai veikia patalpų viduje, nes nuo saulės spindulių jis rodo klaidingus duomenis. Jis turi trikdžių su juodais objektais dėl šviesos absorbcijos bei nemato stiklinių paviršių, pavyzdžiui, stiklinių durų, langų. Atliktas bandymas, kaip įvairios objektų sugeriamosios savybės paveikia matavimo rezultatus. Ant balto popieriaus lapo, per vidurį, užklijuota juoda juostelė (2.5 pav.).

2.5 pav. Lazerinio skenerio šviesos absorbcija

Atlikti keli matavimai su skirtingais atstumais nuo kliūties (10 cm, 20 cm, 30 cm, 40 cm, 50 cm). 2.6 paveiksle  pavaizduoti gauti lazerinio skenerio duomenys. Vienas programos kvadratas yra lygus 1 cm. Idealiu atveju visi skenavimo taškai turėtų sueiti į vieną tiesią liniją, tačiau dėl triukšmų atsiranda objekto paviršiaus nukrypimai (2.6 pav.). Esant jutikliui, nutolusiam 10–20 cm, skenavimo taškai išsidėstę apie 50 mm plote. Tolstant toliau nuo lazerio taškai išsidėstę apie 30 mm plote.



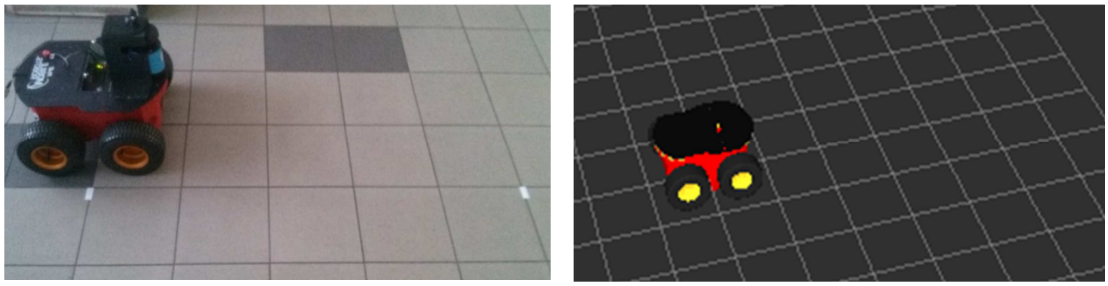
10cm 20cm 30cm 40cm 50cm

2.6 pav. Lazerinio skenerio duomenys

Mažo atstumo matavimai parodo, kad atsiranda ganėtinai rimta skenerio klaida tarp baltos ir juodos spalvos. Tai yra sisteminė klaida. Esant jutikliui, nutolusiam 20–30 cm , klaida siekia apie 20 mm.

### 2.1.4. Odometrija

2D lokalizacijai reikalingos tikslios odometro reikšmės. Tai yra nuvažiuoto atstumo ir greičio reikšmės. Odometro reikšmės labai jautrios klaidoms, nes roboto pozicija apskaičiuojama integruojant greičio matavimą per tam tikrą laiką. Todėl prieš dirbant, tam, kad būtų gaunami kuo tikslesni skaičiavimai, turi būti tiksliai sukalibruota įranga. Kalibravimas buvo atliekamas dviem etapais. Pirmuoju etapu tikrinta, ar robotas realybėje ir vizualizacijoje nuvažiuoja tą patį atstumą. Laboratorijoje pažymimas kelias tarp plytelių (2.7 pav., a).

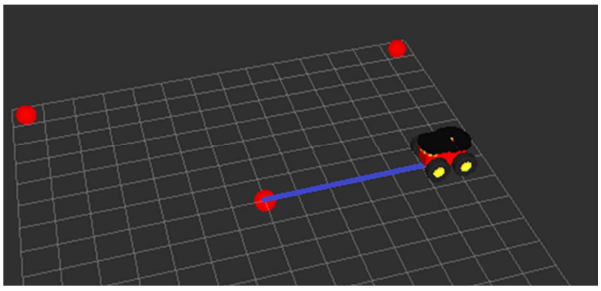


a b

2.7 pav. Roboto odometro kalibravimas: a – vaizdas laboratorijoje, b – vaizdas RVIZ programoje

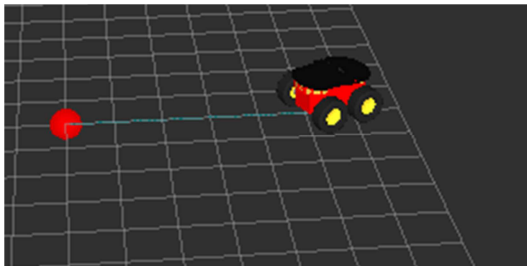
RVIZ programoje sumodeliuojamas robotas ir sudaromas tinklelis pagal realybėje esančias plyteles (2.7 pav., b). Tada kompiuterio klaviatūros pagalba valdome robotą. Robotas turi nuvažiuoti ir sustoti ties pažymėta plytele. Vaizdas realybėje ir vizualizacijoje turi sutapti. Jei

vizualizacijoje robotas neprivažiuoja iki reikiamos vietos (2.8 pav.), mažinamas koeficientas *TicksMM*, jei pravažiuoja, koeficientas didinamas. Bandymo keliu *TicksMM* koeficiento reikšmė gauta 166. Šis koeficientas – tai enkoderio apsisukimų skaičius, atitinkantis 1 mm nuvažiuoto kelio.



2.8 pav. Roboto nuvažiuotas atstumas RVIZ programoje

Kitas svarbus parametras – tai diferencialinis enkoderio apsisukimų skaičius per roboto vieną pilną apsisukimą (360°). Robotas pastatomas tiesiai, tada kompiuterio klaviatūros komandomis robotas sukamas 360° tol, kol grįžta į pradinę poziciją. Tas apsisukimas turi sutapti su atvaizdu RVIZ programos lange (2.9 pav.).



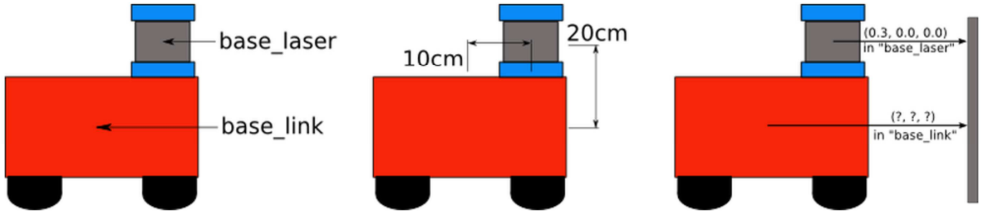
2.9 pav. Roboto pasukimas 360 laipsnių RVIZ programoje

Jei vaizdas realybėje nesutampa su atvaizdu vizualizacijoje, reikia keisti *RevCount* reikšmę. Bandymo keliu nustatyta 37350 *RevCount* reikšmė. Tada tiek programoje, tiek realybėje robotą sukant nors ir kelis kartus, vaizdai sutampa (programos kodas pridėtas prieduose).

### 2.2. Komponentų koordinatės

Koordinačių sistemos svarbios kiekvienoje roboto sistemoje. Robotas gali turėti įvairių komponentų, tokių kaip lazerinis skeneris, kamera, sonarai, roboto ranka, kurie gali būti pritvirtinti kurioje nors vietoje prie roboto rėmo. Kad sistema veiktų sklandžiai, daugelis ROS algoritmų reikalauja kiekvieno komponento koordinačių. Lazerinį skenerį pritvirtinus prie mobiliojo roboto, gaunama skenerio koordinačių ašis (2.10 pav.). Taigi, atsiranda dvi koordinačių sistemos, viena yra roboto, o kita – lazerinio skenerio. Roboto centro nulinės koordinatės aprašomos *base_link*, o prie roboto prisukto lazerinio skenerio nulinės koordinatės aprašomos *base_laser* funkcijoje, kurioje

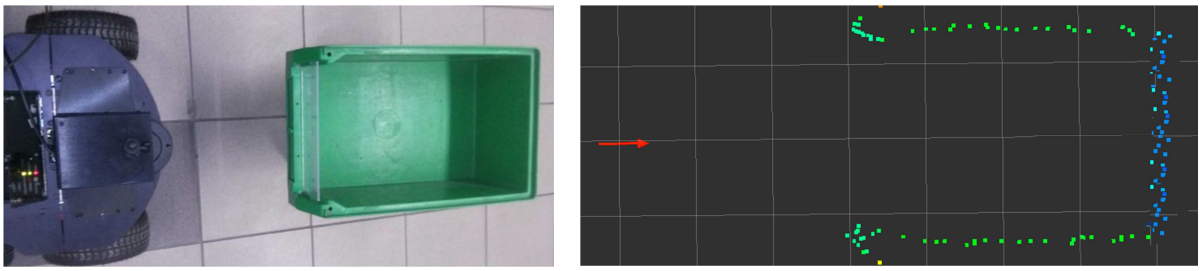
reikia pažymėti, kad skeneris yra 10 cm pastumtas į priekį (x ašis) ir 20 cm pakeltas į viršų nuo roboto centro (z ašis). Taigi, taip gaunami tikslūs duomenis apie roboto vietą nuo sienos.




2.10 pav. Lazerinio skenerio koordinačių nustatymas

Nustačius skenerio koordinates, atliktas bandymas, ar reikšmės atitinka realybę programoje RVIZ. Robotas buvo pastatomas lygiagrečiai su siena. Nustačius visus reikiamus parametrus RVIZ programoje įjungiamas lazeris. Vizualizacijos programoje pastebėta, kad siena ir robotas nerodomas lygiagrečiai, todėl buvo keičiama lazerinio skenerio qx ašis, kuri suka skenerį apie x ašį. Atliekant daugybę bandymų, nustatyta, kad qx ašį reikėjo pasukti -0.39 rad kampu (programos kodas pridėtas prieduose).

Kitas daviklių kalibravimas atliekamas prieš skenerį pastačius dėžutę, kurios vidiniai matmenys yra 0,4 x 0,3 m (2.11 pav., a). Pagal lazerinio skenerio duomenis RVIZ programoje sudarytas tinkliukas, kur vieno kvadratėlio dydis yra 1 cm (2.11 pav., b). Tinkamai suderinus lazerinį skenerį, pagal kvadratėlius galima nustatyti, kad dėžutės matmenys atitinka.



a b

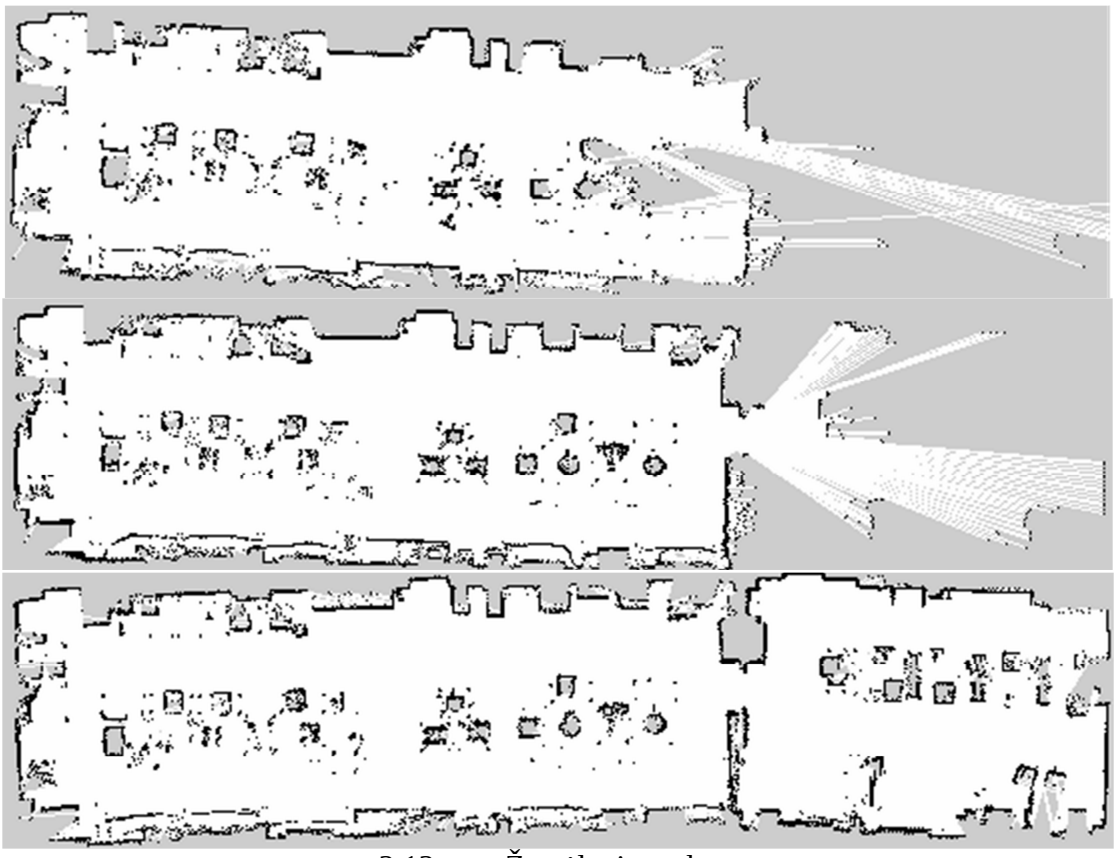
2.11 pav. Lazerio kalibravimas: a – vaizdas laboratorijoje, b – lazerinio skenerio pateikti matavimo duomenys

Taip pat patikrinta, ar tiksliai rodomas skenerio aukštis. Matuota, kokiame aukštyje pamatoma dėžutė, kuri buvo stumdoma rankiniu būdu. Nustatytas lazerinio skenerio aukštis nuo bazinių koordinačių 0,22 m.

Tada su robotu buvo važiuojama į sieną statmena kryptimi ir žiūrima, ar tiksliai rodomas sienos kontūras, privažiavus prie jos. Taip buvo nustatyta m lazerio x ašis, atitraukta nuo bazinių koordinačių 0,10.

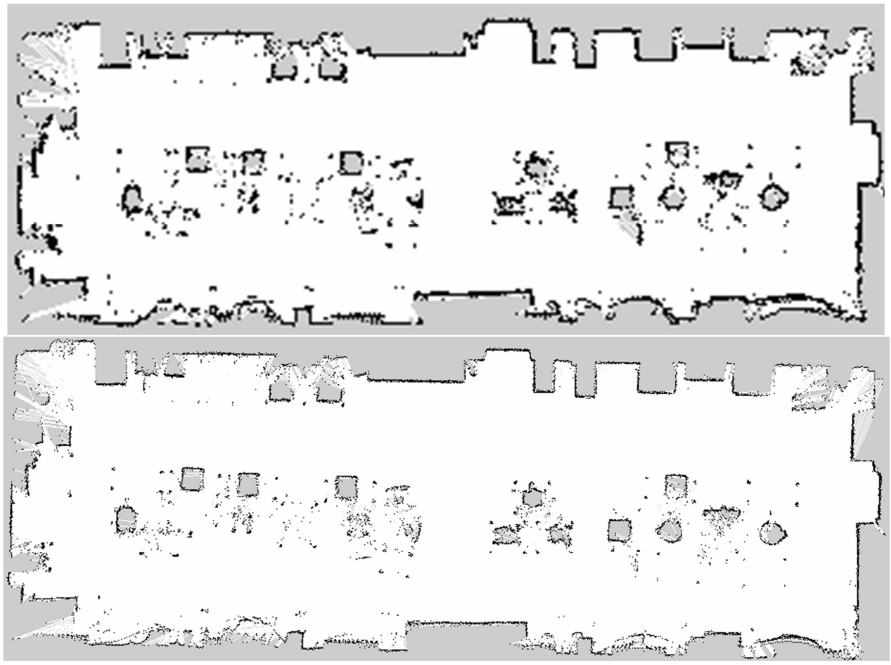
### 2.4. Žemėlapio sudarymas SLAM algoritmui

Tinkamai suderinus daviklius naudojantis ROS navigacijos paketu, pagal dalelių filtro reikšmes galima sukurti patalpos žemėlapį. Žemėlapis sudaromas odometro ir lazerinio skenerio pagalba. Valdant robotą kompiuterio klaviatūra, važinėjama po norimą aplinką ir braižomas žemėlapis naudojantis ROS programos paketu „GMapping“ (2.12 pav. ). Kompiuterio ekrane ROS „RVIZ“ programoje matomas žemėlapio susidarymas. Valdant robotą ir skenuojant aplinką, galima visą jutiklių informaciją įrašyti į „ROS bag“ failus, o po to pritaikius „GMapping“ algoritmą, sudaryti patalpos žemėlapį.



2.12 pav. Žemėlapio sudarymas

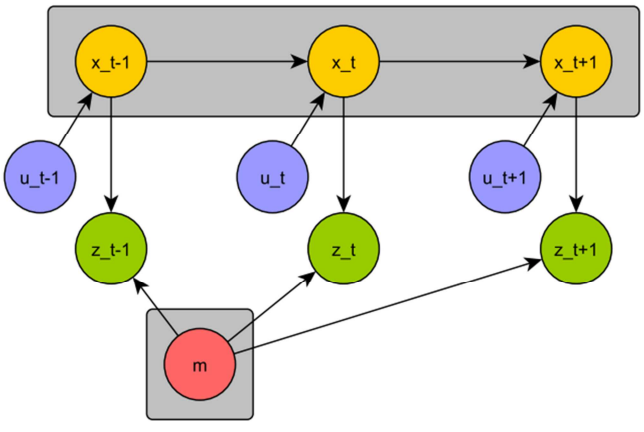
Galima gauti reikiamos rezoliucijos žemėlapius. Naudojant skirtingas skenerio rezoliucijas, sudaryti keli žemėlapiai (2.13 pav.): viršutinis 0,05m tikslumo, o apatinis 0,02 m tikslumo žemėlapis. Reikia nepamiršti, kad aplinka skenuojama su 2D skeneriu, kuris nuo žemės pakeltas 0,2 m, todėl objektai esantys žemiau ar aukščiau yra nematomi. Žemėlapiuose matomi kvadratai tai laboratorijoje esamos spintos, o taškai – tai stalų kojos. 0,01m tikslumo žemėlapis nesigauna, atsiranda tam tikrų taškų, kurių lazerinis skeneris nespėja nuskenuoti ir žemėlapyje atsiranda skylės.



2.13 pav. Skirtingų tikslumų žemėlapiai (viršutinis 0,05 m, apatinis 0,02 m)

## 2.5. Roboto navigacijos tikslumas naudojantis lazeriniu skeneriu

Nežinomoje aplinkoje robotas susiduria su dviem problemomis. Robotui reikia žinoti savo pajudėjimą iš vieno taško į kitą *x(t)* ir užimtą vietą žemėlapyje *m* (2.14 pav.). Šie du kintamieji gaunami iš roboto vidinio jutiklio u1:t (nuvažiuoto atstumo, odometrija) ir žemėlapio skenavimo prietaiso z1:t (lazerinis skeneris). Šis navigacijos būdas dažniausiai vadinasi SLAM (angl. *Simultaneous Localization and Mapping*), tai vienu metu roboto atliekamas lokalizavimas ir žemėlapio sudarymas.




2.14 pav. SLAM algoritmas [17].

Navigacijos paketo pagrindinis tikslas yra nurodyti robotui saugų kelią iki reikiamos vietos pagal nuvažiuoto atstumo, greičio ir aplinkos skenavimo duomenis. Kad programa veiktų sklandžiai, yra daug koreguojamų parametrų, kurie priklauso nuo roboto, jo kinematikos ir jutiklių.

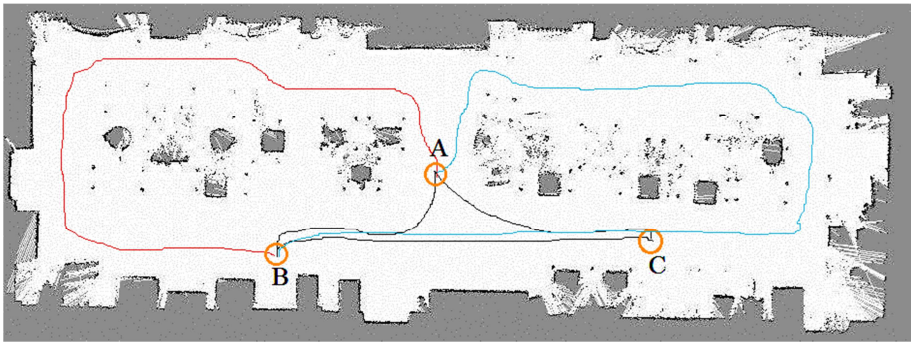
## 2.6. Skyriaus išvados

1. Odometro tikslumas yra labai svarbus jutiklis SLAM algoritmuose, todėl jo veikimą būtina patikrinti ir jį tiksliai sukalibruoti. Jei jis turi didelę paklaidą, gaunami dideli netikslumai sudarant žemėlapį.
2. SICK LMS100 lazeriniu skeneriu sudarytas patalpos žemėlapis 0,02 m rezoliucija. Apribojus robotui greitį, jis gali surinkti duomenis apie nežinomą aplinką ir nubraižyti žemėlapį.
3. Kiekvienam jutikliui, kurio duomenys susiję su roboto orientavimusi, reikia nustatyti jų koordinačių ašį. Pavyzdžiui, nors lazerinis skeneris ant roboto prisisuka per vidurį, tačiau yra vis tiek pasisukęs per 0,39 rad., todėl reikia jį sukalibruoti.
4. Panaudojus ROS vizualizacijos programą RVIZ gautas virtualus robotas, kuris realybėje valdomas kompiuterio klaviatūra.
5. GMapping lokalizacijos algoritmas paremtas Monte Karlo lokalizacijos principu, kur yra didelis dalelių kiekis, $xt$ ,o kiekviena dalelė yra galima roboto pozicija tam tikru laiku. Keičiant dalelių kiekį, galima išgauti vis kitokį navigacijos tikslumą.

# 3. TIRIAMOJI DALIS

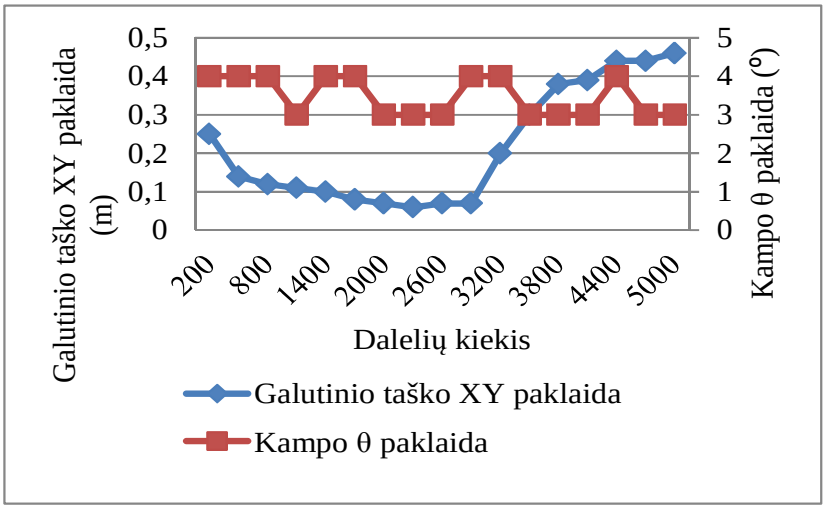
## 3.1. Dalelių kiekio tyrimas

Panaudojus gautą patalpos žemėlapį ir Monte Karlo lokalizacijos algoritmą, robotas pagal duotas koordinates nuvažiuoja į patalpoje nurodytus taškus. Jis įvertina ratų prasisukimą ar kitas paklaidas, palyginant turimą žemėlapį su nuskenuota aplinka, taip patikslindamas savo koordinates. Robotui buvo suvesti trys taškai, į kuriuos jis turi nuvažiuoti ir atsisukti į žemėlapio priekį. Robotui nuvažiavus į kiekvieną tašką, buvo matuojama jo paklaida XY koordinačių sistemoje ir pasisukimo kampo θ paklaida (3.1 pav.).




3.1 pav. Laboratorijos žemėlapis

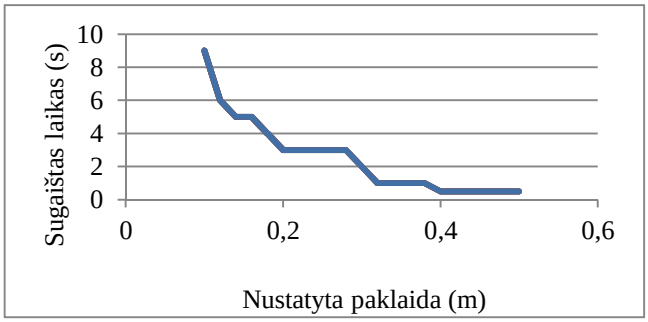
Dalelių kiekis – tai galimi roboto judėjimo bei sustojimo taškai. Keičiant dalelių kiekį, gaunamas skirtingas tikslumas galutiniam roboto sustojimo taškui. Didžiausias gautas tikslumas, kai dalelių kiekis pasiekia apie 2 000 reikšmę (3.2 pav.).




3.2 pav. Monte Karlo lokalizacijos dalelių kiekio reikšmė tikslumui

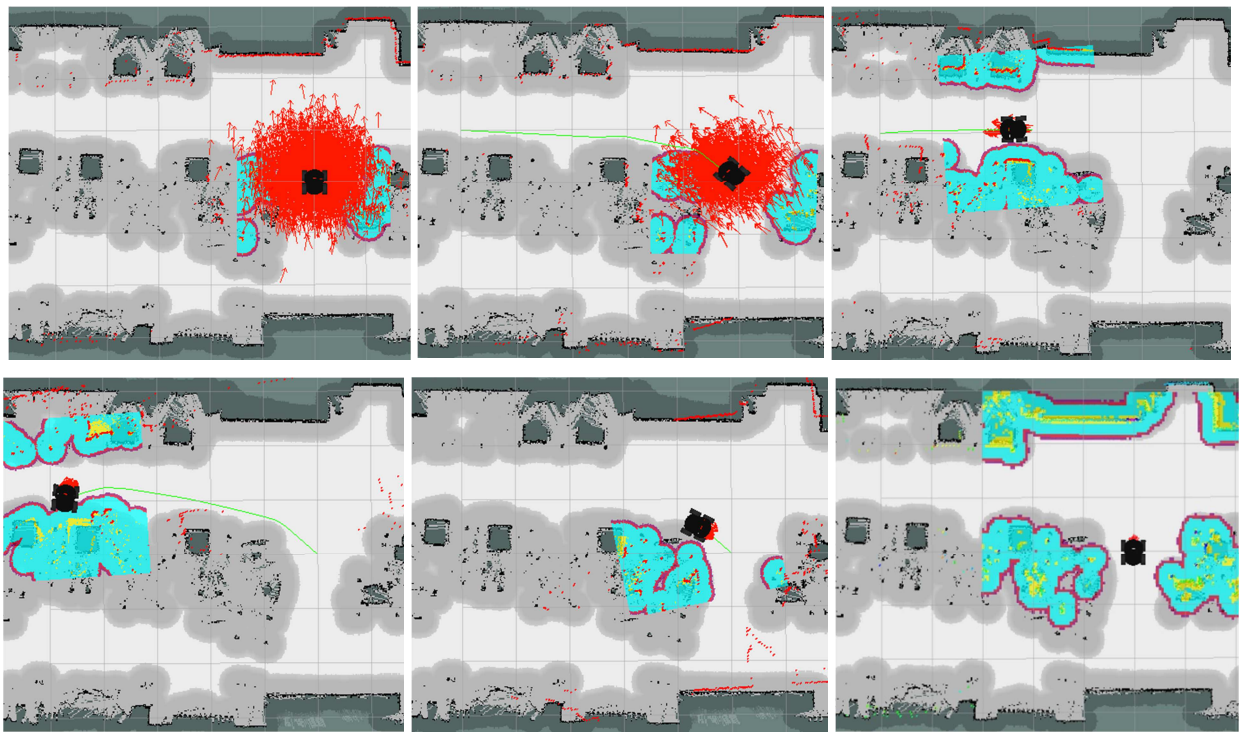
Dalelių kiekį padidinus iki 3800, atsiranda daugiau triukšmų bei apkraunamas kompiuterio procesorius, todėl tikslumas ne gerėja, o blogėja. Roboto pasisukimo kampo paklaida esant tiek mažam, tiek esant dideliam dalelių kiekiui, nekinta ir yra 3°.

Navigacijos programoje galima nusistatyti reikiamą roboto paklaidą galutiniam taškui. Tada robotas bando atsistoti tame taške nustatytam diapazone. Kadangi robotui sukantis jo ratai turi praslysti, todėl tiksliai atsistoti į reikiamas koordinates robotui iš pirmo karto nepavyksta, kartais robotas net apsisuka apie savo ašį. Jei bandoma gauti didelį pozicionavimo tikslumą, robotui sunku iš karto atsistoti į reikiamą tašką, todėl jam nuvažiavus prie galutinio taško reikia pasisukinėti bei apriboti judėjimo ir sukimosi greičius. Galima atsistoti su 10 cm paklaida, tada robotas vidutiniškai sugaišta 9 sekundes (3.3 pav.). Sukantis robotui galutiniame taške, robotas turi apriboti savo greitį iki 0,01 m/s.



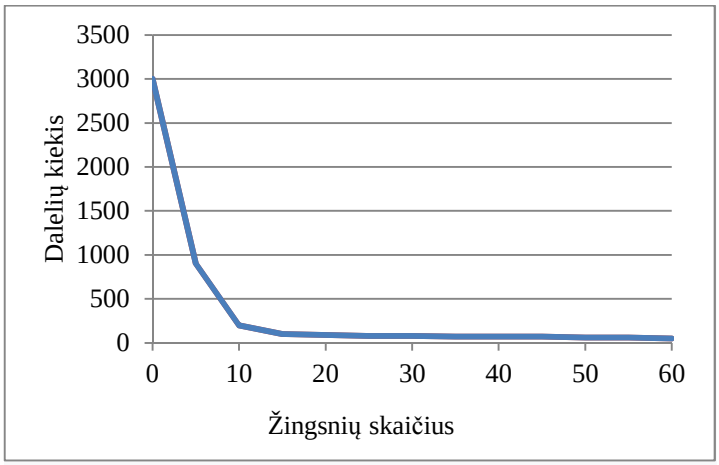

3.3 pav. Papildomai sugaištas roboto laikas pagal nustatytą paklaidą

Panaudojus adaptyvųjį Monte Karlo lokalizacijos algoritmą, galima gauti kintamas dalelių reikšmes, kurios priklauso nuo žingsnių skaičiaus. Pradžioje robotas gali užimti kelias pozicijas, todėl užimamos dalelės išsisklaidę po žemėlapį. 3.4 pav. viršuje matosi mažos rodyklės, kurių pradžia yra galima roboto pozicija, o rodyklės kryptis tai roboto posūkio kampas. Robotas daleles gali dėti vieną ant kitos ir pagal jų svertinius koeficientus tikslinti savo padėtį.



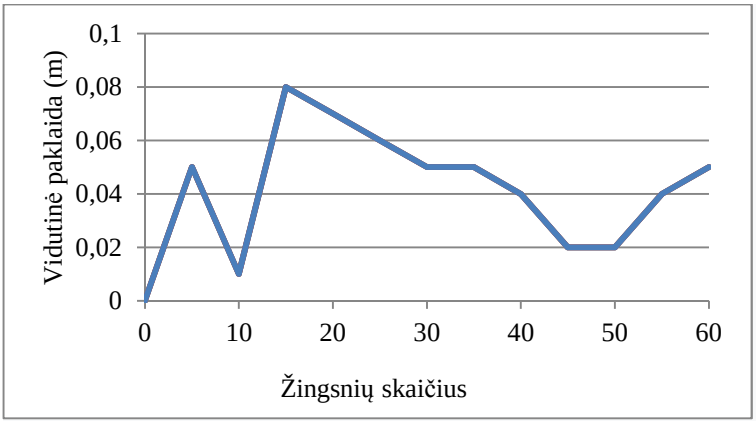
3.4 pav. Dalelių kiekio išsidėstymas kai skenavimo dažnis ne mažesnis nei 25 Hz

Robotui judan ir skenuojant aplinką jis daleles deda kuo arčiau viena kitos. Dažniausiai tos dalelės susideda į krūvą, todėl neverta apkrauti procesoriaus dideliu jų skaičiumi. Dalelių kiekio perskaičiavimo žingsnis yra 0,2 m. Robotui judant nuo taško A į tašką C (3.1 pav.) ir atgal susidaro 60 žingsnių. Jei pradžioje dalelių skaičius buvo 3000, tai po 10 žingsnių, po 2 metrų liko reikalinga tik 300 dalelių (3.5 pav.), nes dauguma dalelių susistūmė i vieną krūvą. Robotui toliau judant aplinkoje dalelių skaičius nukrito iki 100. Pastebėta, kad robotui staigiau sukantis dalelių kiekis ir išsidėstymas prasiplečia, tačiau judant toliau viskas atsistoja į savo vietas.



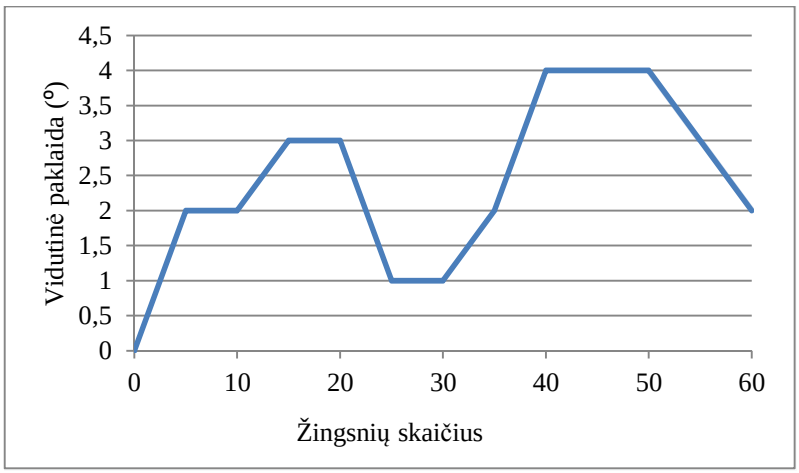

3.5 pav. Dalelių kiekio kitimas nuo žingsnių skaičiaus

Pamatavus roboto tikslumą kas 1 metrą gauta vidutinė 0,04 m paklaida (3.6 pav.), tačiau pastebėta, kad robotui po pasisukimo dalelių skaičius padidėja. Prarandamas tikslumas, nes atsiranda daugiau galimų roboto pozicijų, didėja taškų išsibarstymas, todėl roboto pasisukimo greitis ir pagreitis yra apribojamas (Programos kodas pridedamas į priedus).



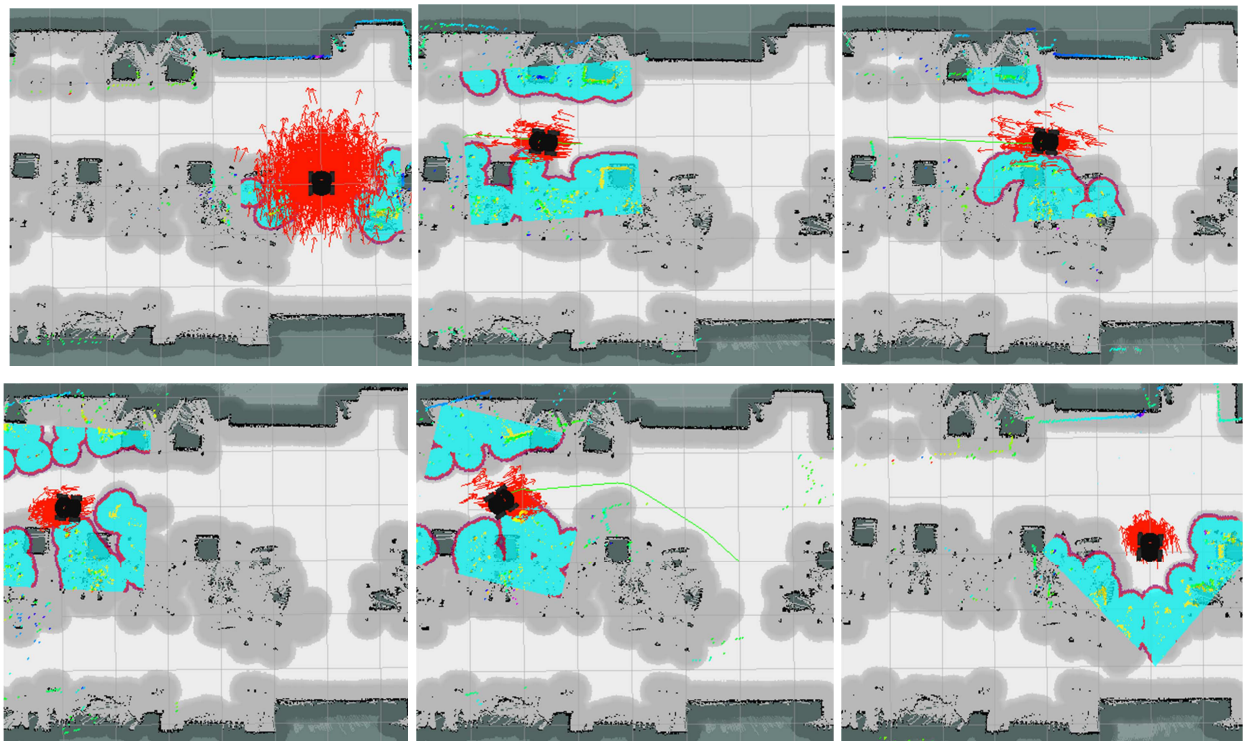

3.6 pav. Roboto pozicijos paklaida nuo žingsnių skaičiaus kai skenavimo dažnis > 25 Hz

Panaši situacija tampa pamatavus roboto pasisukimo kampą (3.7 pav.). Robotui nuvažiavus į tašką, paklaida sumažėja, tačiau važiuojant atgal paklaida vėl didėja, kol nepasiekia galutinio taško. Gauta vidutinė 2,4$^{o}$ paklaida.



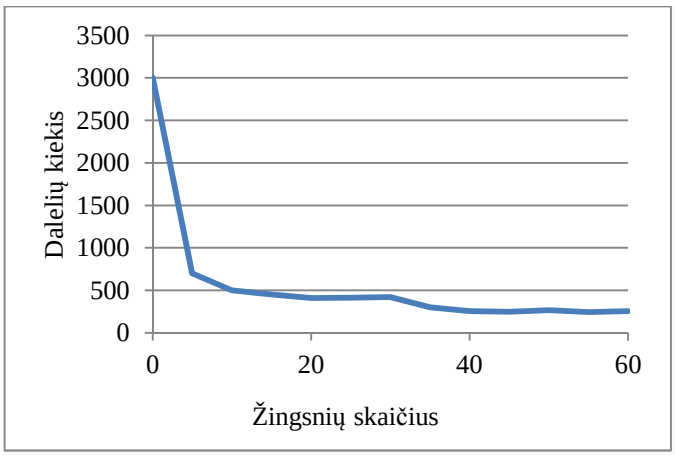

3.7 pav. Roboto orientacijos paklaida nuo žingsnių skaičiaus, kai skenavimo dažnis > 25 Hz

Žemėlapio atnaujinimo dažnis labai svarbus tikslios pozicijos gavimui. Jei skenavimo dažnis nukrenta iki 10 Hz ir mažiau, judant robotui, dalelės žemėlapyje išsimėto, ir taip prarandamas tikslumas. 3.8 paveiksle matome kaip robotui judant dalelės plečiasi. Jei robotas važiuoja tiesiai, dalelės plečiasi į šonus, tačiau po roboto pasisukimo jos atsiranda roboto priekyje ir gale. Dalelių skaičius nemažėja, todėl atsiradus didelei paklaidai robotas panaikina prieš tai buvusias daleles ir sudaro naujas.



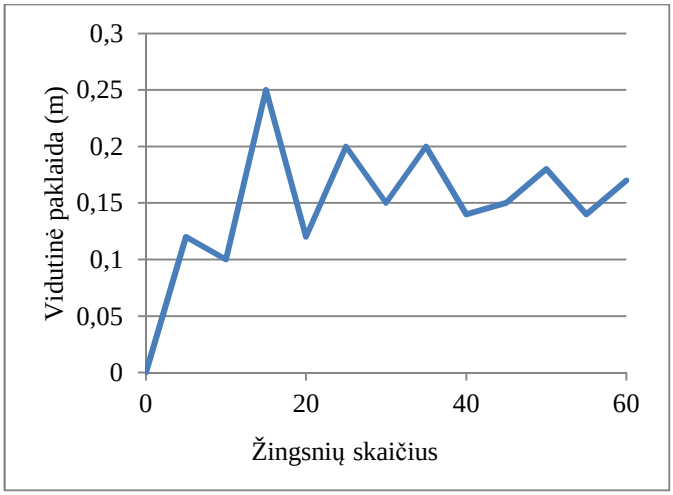
3.8 pav. Dalelių kiekio išsidėstymas, kai skenavimo dažnis mažesnis nei 10 Hz

Dalelių kiekio padidėjimas ir sumažėjimas pateiktas grafike (3.9 pav.). Pradžioje buvo 3000 dalelių, pajudėjus robotui dalelių skaičius nukrito iki 500, tačiau dalelės per daug išsiskyrę. Tai reiškia, kad pozicijos tikslumas sumažėjo.



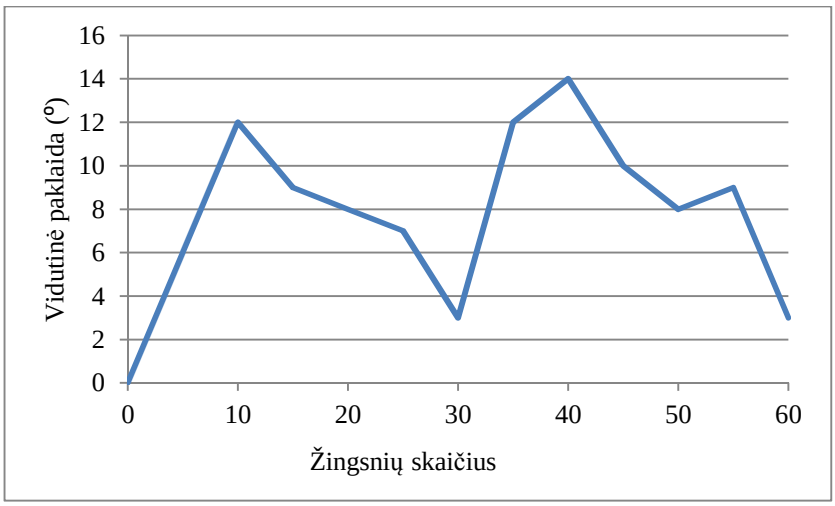

3.9 pav. Dalelių kiekio kitimas nuo žingsnių skaičiaus, kai skenavimo dažnis < 10 Hz

Matuojant taškų paklaidas, gauta ganėtinai didelė vidutinė paklaida 0,14 m (3.10 pav.). Pastebimas didelis netolygumas dėl išsibarsčiusių dalelių. Vieno taško paklaida siekė 25 cm, tai tikrai didelė paklaida tokioje mažoje erdvėje.




3.10 pav. Roboto pozicijos paklaida nuo žingsnių skaičiaus kai skenavimo dažnis < 10 Hz

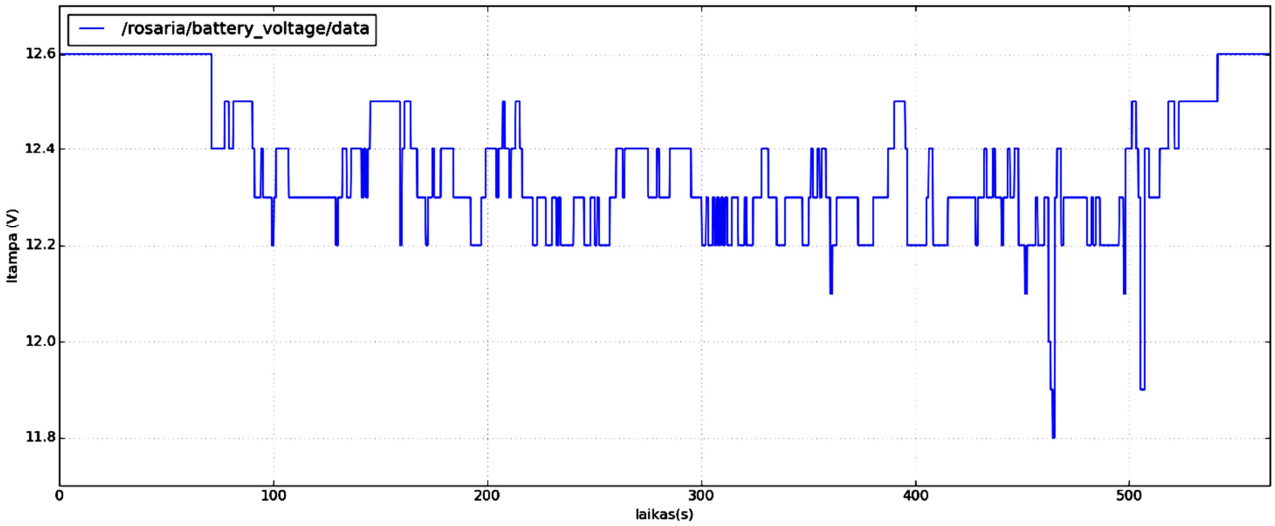
Robotas, turėdamas žemėlapyje daug išsisklaidžiusių dalelių, judėdamas savo maršrutu labai daug pasisuka į šalį. Taip atsitinka dėl to, kad jam sunku nuskaityti žemėlapį sukimosi metu (3.11 pav.). Gauta 8° vidutinė paklaida. Robotas, nuvažiavęs į galutinį tašką, apsisuka, todėl galutiniuose taškuose gauta 3° paklaida. Tačiau, kai robotas važiuoja tiesiai ir privažiavęs posūkį pasisuka, atsiranda papildomų dalelių, kurios padidina paklaidą iki 12°–14°.




3.11 pav. Roboto orientacijos paklaida nuo žingsnių skaičiaus kai skenavimo dažnis < 10 Hz

### 3.2. Roboto baterijos stebėjimas

Robotui važinėjant buvo matuota baterijos įtampa (3.12 pav.). Roboto baterija stovėjimo metu teikė 12,6 V įtampą. Robotui sukantis ar gaunant didesnę apkrovą baterijos įtampa nukrisdavo iki 12,2 V. Tačiau roboto baterijai nukritus žemiau negu 12 V, žemėlapyje atsiranda netikslumų. Kad robotas neprarastų savo tikslumo, reikia sekti baterijos įtampą, kad robotas galėtų nuvažiuoti pasikrauti bateriją tinkamu metu.

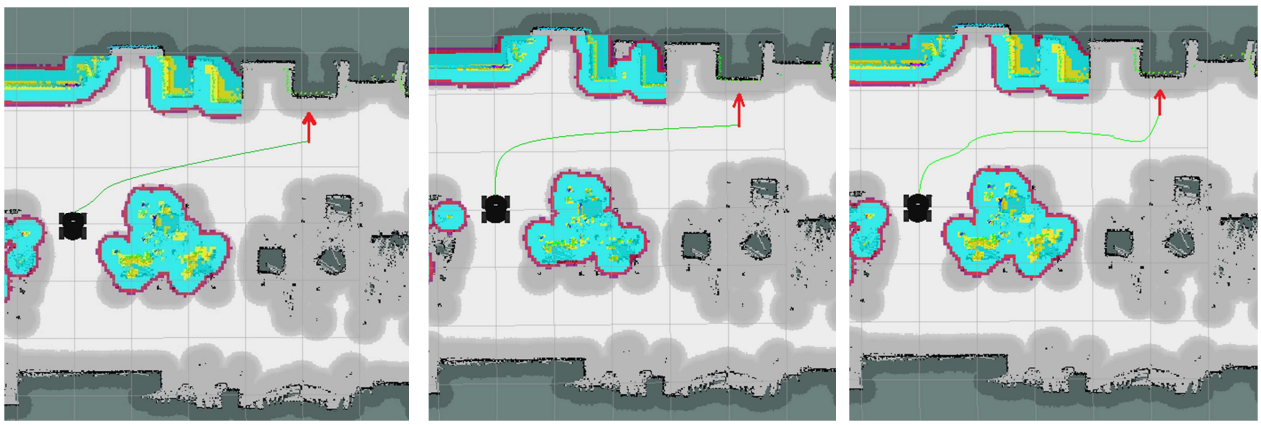



3.12 pav. Roboto įtampas stebėjimas

### 3.3. Roboto maršruto planavimas

Robotui judėti tam tikra trajektorija yra svarbūs *cost_ factor* ir *neutral_cost* parametrai. Keičiant šiuos parametrus, pasikeičia roboto maršrutas. Reikia pasirinkti tokį maršrutą, kuris greitai ir saugiai apvažiuotų kliūtis.

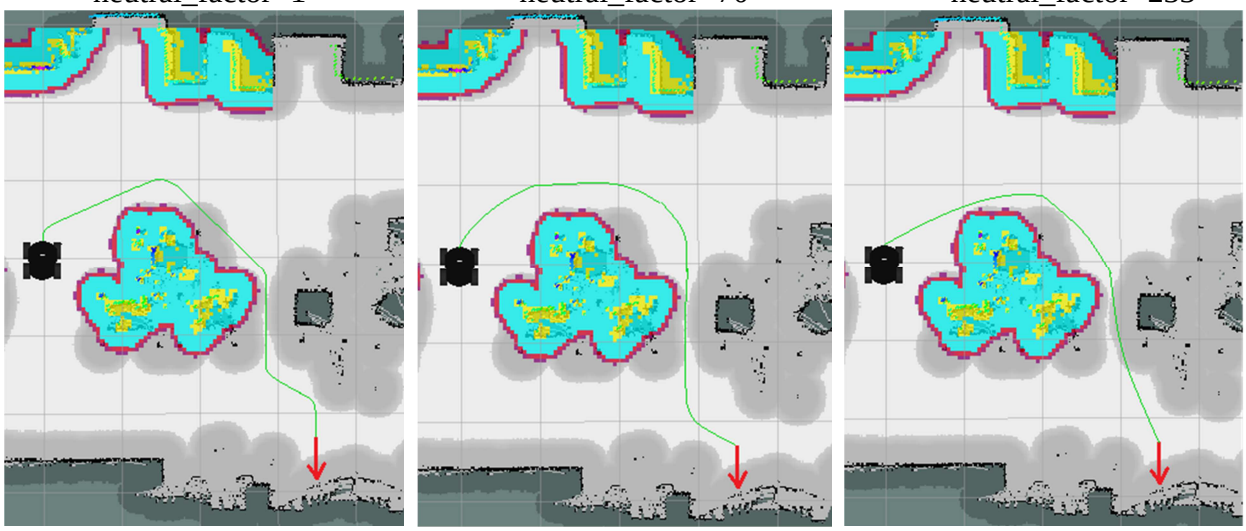
cost_factor=0.1 cost_factor=0.6 cost_factor=3.4



3.13 pav. SLAM algoritmo cost_factor parametro tyrimas

Pagal eksperimentą (3.13 pav.), per mažas ir per didelis *cost_factor* apsunkina roboto judėjimą tarp siaurų praėjimų. Jei *cost_factor* per mažas, robotas, tik privažiavęs prie kliūties, kurią reikia apvažiuoti, pradeda suktis. Jei *cost_factor* per didelis, robotas iš per toli pradeda suktis ir taip jis apsisunkina judėjimo trajektoriją. Aišku, tai tikriausiai priklauso nuo roboto kinematikos. Pagal gautus duomenis robotas tiksliausiai pradeda suktis, kai *cost_factor* yra 0.6.

neutral_factor=1 neutral_factor=70 neutral_factor=233

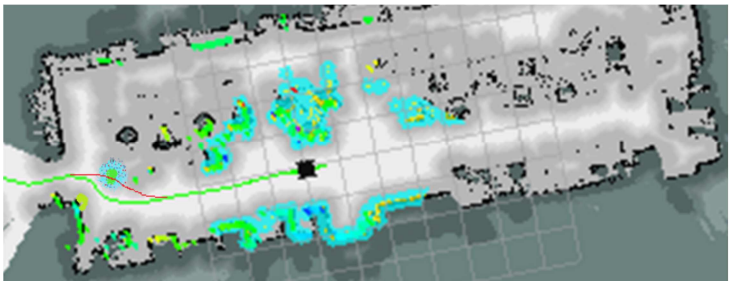


3.14 pav. SLAM algoritmo neutral_factor parametro tyrimas

Pagal 3.14 pav. matome, kad kitas parametras *neutral_factor* nustato tam tikrą spindulį, kuriuo robotas gali pasisukti. Pagal reikiamą trajektoriją nustatome, kad robotas geriausiai apvažiuoja kliūtis, kai *neutal_factor* reikšmė yra 70.

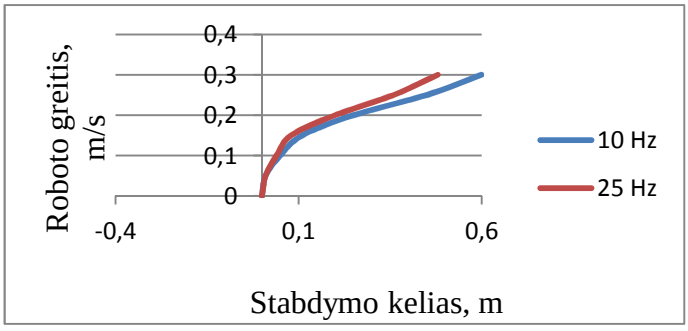
Robotas, važiuodamas nustatytais maršrutais, kelyje gali sutikti dinaminių kliūčių, pavyzdžiui, praeiti žmogus (3.15 pav.). Raudona linija – tai roboto senas maršrutas, tačiau toje vietoje atsiradus kliūčiai, robotas perplanuoja maršrutą (žalia linija.) Jei žmogus eitų toliau, robotas vėl perplanuotų maršrutą. Jei robotas jau prie pat žmogaus, robotas patenka į apsaugos zoną, kur jo greitis apsiriboja iki 0,05 m/s. Jeigu žmogus ar kitas objektas, pavyzdžiui, palikta dėžė, robotui

užstoja kelią, jis apsisuka aplink save 360°, ir, jei yra kitas koridorius, ar takas, robotas važiuoja link jo. Jei kito kelio nėra, robotas apie tai praneša garsiniu signalu.



3.15 pav. Roboto kelio perplanavimas išėjus ant kelio žmogui

Žmogus gali staiga išlysti iš už kampo. Tada robotas turi spėti sustoti arba nusukti į šalį. Pagal tai buvo įvertintas roboto greitis ir sustojimo kelias (3.16 pav.). Eksperimento metu, robotui nustatytas pastovus greitis ir judėjimas linija. Tada staiga uždengiamas skenerio jutiklis ir matuojamas sustojimo kelias. Pagal grafiką matome, kad sustojimo laikas dar priklauso nuo aplinkos skenavimo dažnio. Jei aplinkos skenavimo dažnis yra 10 Hz, o robotas važiuoja 0,15 m/s greičiu, jis sustoja už 0,14 m, tačiau jei roboto greitis siekia 0,3 m/s, tada robotas sustoja už 0,5 m. Tai parodo, kad didėjant greičiui sustojimo kelias didėja neproporcingai. Robotui skenuojant aplinką 25 Hz dažniu ir važiuojant 0,2 m/s greičiu, jis sustoja už 0,2 m. Jei vienoje patalpoje pastoviai vaikšto žmonės ir važinėja robotas, tai jo greitis gali būti 0,1 m/s. Toks greitis ganėtinai mažas, todėl avariniams sustojimams turėtų būti kita apsauginė sistema, pavyzdžiui, bamperis.




3.16 pav. Stabdymo kelio priklausomybė nuo roboto greičio.

### 3.4. Skyriaus išvados

1. Navigacijos sistema gauta roboto 0,06 m paklaida xy ašyje, ir $3^0$ paklaida roboto pasisukimui. Norint išgauti didesnį tikslumą, reikalingas ilgesnis laiko tarpas.
2. Naudojantis adaptyviuoju Monte Karlo lokalizacijos algoritmu gauta vidutinė 0,04 m paklaida galutiniam taškui.
3. Cost_factor koeficientą nustačius 0,6 ir neutral_factor nustačius 70, gaunamos paprastos, užapvalintos roboto judėjimo trajektorijos.

# IŠVADOS

1. Gauta autonominė transporto priemonė, kuri pagal lazerinio skenerio duomenis, 2D žemėlapyje trumpiausiu atstumu pasiekia nurodytas koordinates. Gautos laksčios judėjimo trajektorijos, kai cost_factor reikšme yra 0,6.
2. SICK LMS100 lazeriniu skeneriu sudarytas patalpos žemėlapis 0,02 m rezoliucija. Apribojus robotui greitį, robotas gali surinkti duomenis apie nežinomą aplinką ir nubraižyti žemėlapį. 0,01m rezoliucijos žemėlapio neišeina padaryti, nes lazerinis skeneris nespėja nuskaityti aplinkos ir žemėlapyje gaunasi skylės.
3. Naudojantis adaptyviuoju Monte Karlo lokalizacijos algoritmu, kai skenavimo dažnis didesnis negu 25 Hz , gauta vidutinė 0,04 m paklaida galutiniam taškui.
4. Mobilaus roboto maksimalus stabdymo kelias 0,5 m, kai roboto greitis 0,3 m/s ir skenavimo dažnis 25 Hz.

# LITERATŪROS SĄRAŠAS


1. Compagna, D.; Derpmann, S.; Mauz, K. (2009), The Operation of Autonomous Mobile Robot Assistants in the Environment of Care Facilities Adopting a User-Centered Development Design, Enterprise and Work Innovation Studies, 5, pp. 11-24 ISSN 1646-1223
2. Gianluca Ippoliti, Leopoldo Jetto, Sauro Longhi, Andrea Monterił „*Comparative Analysis of Mobile Robot Localization Methods Based On Proprioceptive and Exteroceptive Sensors*“ Via Brecce Bianche – 60131 Ancona Italy. 2007
3. Butdee, Suthep and Suebsomran, Anan and Vignot, Frederic and Yarlagadda, Prasad K.D.V. (2008) *Control and path prediction of an automate guided vehicle.* In: 9th Global Congress on Manufacturing and Management, 12-14, November, 2008, Gold Coast, Australia.
4. POVILAS CIŽAUSKAS. *Mobilaus roboto maršruto tikslinimo algoritmo sukūrimas ir tyrimas. Magistro Darbas.* Kaunas: Kauno technologijos universitetas. Prieiga per eLABa – nacionalinė Lietuvos akademinė elektroninė biblioteka, 2015
5. Shameek Ganguly, Akash Garg, Akshay Pasricha and S.K. Dwivedy „*Design and fabrication of a novel three wheel robot with a SLE based lifting mechanism and line tracking capability*“ 14*th* National Conference on Machines and Mechanisms (NaCoMM-09), NIT, Durgapur, India, December 17-18, 2009
6. Xing Wu*, Peihuang Lou, Ke Shen, Guangqing Peng, Dunbing Tang „ *PRECISE TRANSHIPPMENT CONTROL OF AN AUTOMATED MAGNETIC-GUIDED VEHICLE USING OPTICS POSITIONING*“ Yuedao Street, Nanjing, P. R. China 2014.
7. Proceedings - IEEE International Conference on Robotics and Automation „*Mobile robot navigation using self-similar landmarks* “ February 2000
8. Jinwook Huh, Woong Sik Chung, Sang Yep Nam, and Wan Kyun Chung „*Mobile Robot Exploration in Indoor Environment Using Topological Structure with Invisible Barcodes“* ETRI Journal, Volume 29, Number 2, April 2007
9. R. SIEGWART, R. NOURBAKHSH „*Autonomous Mobile Robots*“ A Bradford Book The MIT Press Cambridge, Massachusetts London, England 2004.
10. Interaktyvi svetainė. „*Automated Guided Vehicle Systems*“ žiūrėta 2017-05-17. http://www.goetting.de/komponenten/43600
11. DEIMANTAS ŽVIRBLIS. *Robotų tarpusavio orientavimo sistemos tyrimas. Magistro darbas* Vilnius: Vilniau Gedimino technikos universitetas. Prieiga per eLABa – nacionalinė Lietuvos akademinė elektroninė biblioteka, 2013

12. Yew Tuck Chin, Han Wang, Leng Phuan Tay, Hui Wang, William Y. C. Soh „*Vision Guided AGV Using Distance Transform*“ Proceedings of the 32nd ISR(International Symposium on Robotics), 19-21 April 2001
13. Interaktyvi svetainė „*Navigation Technology*“ žiurėta 2016-05-17. https://www.transbotics.com/learning-center/guidance-navigation
14. Fabio Brea. „*Potential field navigation for telepresence robots driven by BCI*“ Magistrali biennali. UNIVERSITÀ DEGLI STUDI DI PADOVA. 2016
15. ActivMedia Robotics. Pioneer 3 and Pioneer 2 *H8*-Series Operations Manual. version 3, August 2003.
16. Safdar Zaman, Wolfgang Slany, Gerald Steinbauer. ROS-based Mapping, Localization and Autonomous Navigation using a Pioneer 3-DX Robot and their Relevant Issues. ISBN 978-1-4577-0069-9.
17. Risto Kaijaluoto. *„Precise indoor localization for mobile laser scanner“*. Master's thesis. Aalto university. 2015

PUBLIKACIJOS DARBO TEMATIKA

18. Mainelis J., Sinkevičius V. Autonominio roboto su lazeriniu skeneriu navigacijos tyrimas. Studentų mokslinė konferencija „Technologijų ir verslo aktualijos - 2017“ pranešimų medžiaga. Panevėžys, 2017.

# PRIEDAI

# 1. VISŲ ROBOTO KOMPONENTŲ *LAUNCH* FAILAS

```
      <launch>

      <!-- Launching p2os RobotModel -->
        <include file="$(find p2os_urdf)/launch/pioneer3at_urdf.launch"/>

      <!-- Launching LMS1xx_node for laser Sick LMS100 via ethernet -->

      <node name="LMS" pkg="LMS1xx" type="LMS100"
args="_host:=169.254.233.11"/>

      <!-- Starting rosaria driver for motors and encoders -->
        <include file="$(find pioneer_utils)/sensors/rosaria.launch"/>

        <node pkg="tf" type="static_transform_publisher"
name="base_to_laser_broadcaster" args="0.09 0 0.22 -0.39
       0 0 base_link laser 1" />


     </launch>
```

# 1. PIONEER P3-AT SUJUNGIMO SU KOMPIUTERIŲ *LAUNCH* FAILAS

```
      <launch>

      <!—Pleidžiamos rosaria tvarkyklės varikliams ir enkoderiams -->
        <node       name="rosaria"       pkg="rosaria"       type="RosAria"
args="_port:=/dev/ttyUSB0">
        <rosparam>
            TicksMM: 166
            RevCount: 37350
            DriftFactor: 0
            trans_accel: 0
            trans_decel: 0
            rot_accel: 0
            rot_decel: 0
        </rosparam>
        <remap from="~cmd_vel" to="cmd_vel"/>
        </node>

      </launch>
```

# 1. ŽEMĖLAPIO SUDARYMO *LAUNCH* FAILAS

```
      <launch>
           <node pkg="move_base" type="move_base" respawn="false"
name="move_base" output="screen">
              <rosparam file="$(find
pioneer_utils)/navigation/common/costmap_common_params_p3at.yaml"
command="load" ns="global_costmap" />
              <rosparam file="$(find
pioneer_utils)/navigation/common/costmap_common_params_p3at.yaml"
command="load" ns="local_costmap" />
              <rosparam file="$(find
pioneer_utils)/navigation/common/local_costmap_params.yaml" command="load"
/>
              <rosparam file="$(find
pioneer_utils)/navigation/local_navigation/global_costmap_params.yaml"
command="load" />
              <rosparam file="$(find
pioneer_utils)/navigation/common/base_local_planner_params.yaml"
command="load"/>
              <rosparam file="$(find
pioneer_utils)/navigation/common/global_planner_params.yaml" command="load"
/>
              <rosparam file="$(find
pioneer_utils)/navigation/common/recovery_behaviors.yaml" command="load" />
              <rosparam>
                  planner_frequency: 1.0
              </rosparam>
              <param name="base_global_planner"
value="global_planner/GlobalPlanner"/>
          </node>
      </launch>
```

# 1. ŽEMĖLAPIO PARAMETRŲ NUSTATYMO FAILAS

```
      local_costmap:
        global_frame: /odom
        robot_base_frame: /base_link
        update_frequency: 20.0
        publish_frequency: 20.0
        static_map: false
        rolling_window: true
        width: 6.0
        height: 6.0
        resolution: 0.05
        max_obstacle_height: 0.5
        plugins:
            - {name: obstacle_layer_laser, type: "costmap_2d::VoxelLayer"}

            - {name: inflation_layer, type: "costmap_2d::InflationLayer"}
          obstacle_layer_laser:
          observation_sources: sick_lms1xx
          sick_lms1xx: {sensor_frame: laser, data_type: LaserScan, topic:
scan, marking: true, clearing: true, obstacle_range: 10.0, raytrace_range:
12.0, inf_is_valid: true}
        inflation_layer:
          inflation_radius: 0.2
```

# 1. BAZINIAI ROBOTO PARAMETRAI

```
TrajectoryPlannerROS:
  max_vel_x: 0.3
  min_vel_x: 0.05
  max_vel_theta: 0.3
  min_in_place_vel_theta: 0.1
  acc_lim_theta: 0.6
  acc_lim_x: 1.2
  acc_lim_y: 1.2
  holonomic_robot: false
  yaw_goal_tolerance: 3.1415
  sim_granularity: 0.025
  sim_time: 4.0
  meter_scoring: true
  pdist_scale: 0.9
  gdist_scale: 0.6
```

# 1. ŽEMĖLAPIO SUDARYMO *LAUNCH* FAILAS

```
<launch>
    <node pkg="amcl" type="amcl" name="amcl">

                <param name="odom_model_type" value="diff"/>
                <param name="odom_alpha5" value="0.1"/>
                <param name="transform_tolerance" value="0.2" />
                <param name="gui_publish_rate" value="10.0"/>
                <param name="laser_max_beams" value="30"/>
                <param name="min_particles" value="500"/>
                <param name="max_particles" value="5000"/>
                <param name="kld_err" value="0.05"/>
                <param name="kld_z" value="0.99"/>
                <param name="odom_alpha1" value="0.2"/>
                <param name="odom_alpha2" value="0.2"/>
                <!-- translation std dev, m -->
                <param name="odom_alpha3" value="0.8"/>
                <param name="odom_alpha4" value="0.2"/>
                <param name="laser_z_hit" value="0.5"/>
                <param name="laser_z_short" value="0.05"/>
                <param name="laser_z_max" value="0.05"/>
                <param name="laser_z_rand" value="0.5"/>
                <param name="laser_sigma_hit" value="0.2"/>
                <param name="laser_lambda_short" value="0.1"/>
                <param name="laser_lambda_short" value="0.1"/>
                <param name="laser_model_type"
                value="likelihood_field"/>
                <!-- <param name="laser_model_type" value="beam"/> -->
                <param name="laser_likelihood_max_dist" value="2.0"/>
                <param name="update_min_d" value="0.2"/>
                <param name="update_min_a" value="0.5"/>
                <param name="odom_frame_id" value="odom"/>
                <param name="resample_interval" value="1"/>
                <param name="transform_tolerance" value="0.1"/>
                <param name="recovery_alpha_slow" value="0.0"/>
                <param name="recovery_alpha_fast" value="0.0"/>
    </node>
</launch>
```