KAUNO TECHNOLOGIJOS UNIVERSITETAS

ELEKTROS IR ELEKTRONIKOS FAKULTETAS

Laurynas Petkevičius

ŽEMĖLAPIŲ, SKIRTŲ ROBOTŲ NAVIGACIJAI NEŽINOMOJE APLINKOJE, SUDARYMO SPALVOTAIS PETRI TINKLAIS GALIMYBIŲ TYRIMAS

Baigiamasis magistro projektas

Vadovas
Doc. dr. Virginijus Baranauskas

KAUNAS, 2015

KAUNO TECHNOLOGIJOS UNIVERSITETAS
ELEKTROS IR ELEKTRONIKOS FAKULTETAS
AUTOMATIKOS KATEDRA

ŽEMĖLAPIŲ, SKIRTŲ ROBOTŲ NAVIGACIJAI NEŽINOMOJE APLINKOJE, SUDARYMO SPALVOTAIS PETRI TINKLAIS GALIMYBIŲ TYRIMAS

Baigiamasis magistro projektas
Valdymo technologijos (621H66001)

Vadovas
Doc. Dr. Virginijus Baranauskas
2015-06-04

Recenzentas
Lekt. Gintautas Narvydas

Projektą atliko
Laurynas Petkevičius
2015-06-04

KAUNAS, 2015

KAUNO TECHNOLOGIJOS UNIVERSITETAS

Elektros ir elektronikos fakultetas

(Fakultetas)

Laurynas Petkevičius

(Studento vardas, pavardė)

Valdymo technologijos, 621H66001

(Studijų programos pavadinimas, kodas)

Baigiamojo projekto „Žemėlapių, skirtų robotų navigacijai nežinomoje aplinkoje, sudarymo spalvotais Petri tinklais galimybių tyrimas“

AKADEMINIO SĄŽININGUMO DEKLARACIJA

20 ___ m. _____ ____ d.

Kaunas

Patvirtinu, kad mano **Lauryno Petkevičiaus** baigiamasis projektas tema „Žemėlapių, skirtų robotų navigacijai nežinomoje aplinkoje, sudarymo spalvotais Petri tinklais galimybių 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)*

Petkevičius, L. Žemėlapių, skirtų robotų navigacijai nežinomoje aplinkoje, sudarymo spalvotais Petri tinklais galimybių 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 – ištirti spalvotų Petri tinklų galimybes, juos naudojant žemėlapių, skirtų roboto navigacija nežinomoje aplinkoje, sudarymui.

Šiame darbe tiriami sprendimai, leidžiantys mobiliam robotui sudaryti aplinkos žemėlapį nežinomoje aplinkoje, taip pat pasiūlytas naujas vektorinių žymių generavimo metodas žemėlapio sudarymui nežinomoje aplinkoje. Atlikus šios srities technologijų analizę surasta įvairių realizacijai tinkamų metodų. Išanalizuoti žemėlapio kūrimo būdai suskirstyti į dvi dalis: kai naudojama robotai realioje aplinkoje ir kai modeliuojama virtualioje aplinkoje. Ištirta ar galima vektorinių žymių generavimo metodą naudoti žemėlapio kūrimui nežinomoje aplinkoje, naudojant spalvotus Petri tinklus, kurie realizuoti programų pakete „CENTAURUS CPN“. Vektorinių žymių generavimo metodo tikslas yra aptikti statinių kliūčių pokyčius, fiksuojant statinių kliūčių kampų taškus, kurie vadinami trūkio taškais ir pagal juos sudaryti aplinkos žemėlapį. Aptinkant statinių kliūčių trūkio taškus fiksuojami statinių kliūčių kampai, angos ir kiti staigūs konfigūracijos pakitimai. Programinio paketo „CENTAURUS CPN“ aplinkoje buvo sukurtos keturios skirtingo sudėtingumo aplinkos, pagal kurias buvo sudarinėjami žemėlapiai. Sudaryti aplinkų žemėlapiai buvo lyginami tarpusavyje ir tiriama kiekvieno „CENTAURUS CPN“ programiniame pakete nurodomo parametro įtaka žemėlapio tikslumui bei išbaigtumui.

Reikšminiai žodžiai: žemėlapio sudarymas, CENTAURUS CPN, spalvoti Petri tinklai

Petkevičius, Laurynas. Investigation of Maps Creation Possibilities for Robot Navigation in Unknown Environment based on Colored Petri Nets. 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 work is to investigate maps creation possibilities for robot navigation in unknown environment based on colored Petri nets.

The presented work describes solutions for the mobile robot, which allows to build a map of an unknown environment, and also new method for building a map of an unknown environment based on vector marks generation was proposed. Mapping methods have been analyzed. Mapping techniques are divided into two parts: when robots are used in a real environment and when modeled in the virtual environment. Vectors marks generation method presented as an opportunity to map building of an unknown environment. Method based on colored Petri nets which are used in the program package Centaurus CPN. The aim of method is to detect changes in structures barriers, known as the break points, and build a map of the environment. There were created four different environments, which have been used to build maps. Different maps were compared between each other and obtained results were analyzed. Software Centaurus CPN allows to change map building parameters.

Keywords: map building, Centaurus CPN, colored Petri nets

Turinys

Įvadas

Spartus technologijų vystymasis, augimas ir įvairovė skatina atsisakyti žmogaus rankų darbo, pakeičiant autonominiais robotais, įvairiais technologiniais įrenginiais. Automatinio valdymo sistemų reikšmė šiuolaikinei visuomenei yra labai didelė. Mes taip priklausome nuo šių sistemų, gyvenimas be jų būtų tiesiog neįsivaizduojamas. Gamyklose dažnai naudojami įvairios konstrukcijos, plataus pritaikymo pramoniniai robotai, namuose - autonominiai robotai siurbliai, gatvėse netgi važinėja autonominiai automobiliai. Daugumai šių robotų reikalinga nustatyti esamą poziciją aplinkoje. Dažniausiai roboto aplinka yra žinoma, tačiau įvykus nenumatytiems pakitimams, įvykus avarijai, atsiradus kliūtims ir pan. aplinka tampa nežinoma ir robotas negali toliau vykdyti savo užduoties bei atlikti jam užduoto darbo. Norint išspręsti šią problemą nežinomoje aplinkoje reikalinga sudaryti žemėlapį, pagal kurį nenumatyti pakitimai būtų pašalinti arba sudarytos sąlygos jų išvengti ar apeiti tam, kad robotas galėtų toliau dirbti pagal paskirtį.

Kuriant žemėlapį nežinomoje aplinkoje susiduriama su greitaveikos bei tikslumo problemomis. Šios srities atlikti tyrimai bando spręsti žemėlapio kūrimo problemas analizuodami robotus realioje aplinkoje arba modeliuodami virtualioje aplinkoje. Šiame darbe nežinoma aplinka modeliuojama programinėje virtualioje aplinkoje, siekiant atlikti tyrimą ar spalvoti Petri tinklai leidžia analizuoti ir tyrinėti nežinomą aplinką.

1. Nežinomos aplinkos žemėlapio sudarymas

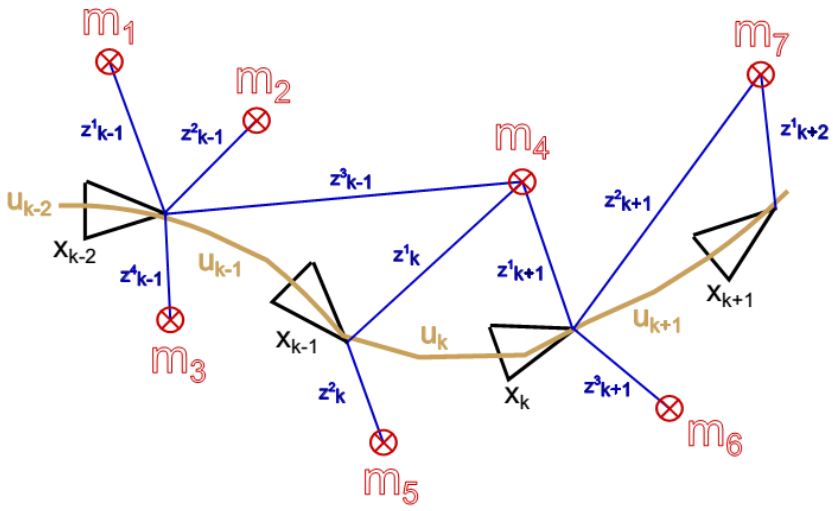
Išanalizavus kelis žemėlapio nežinomoje aplinkoje kūrimo būdus aptikta, kad dažniausiai žemėlapio kūrimo analizavimas orientuotas į naudojamą techninę įrangą arba modeliavimo metodus virtualioje aplinkoje. Išanalizuotos robotų žemėlapio nežinomoje aplinkoje sudarymo galimybės suskirstytos į:

- Žemėlapio kūrimas modeliuojant virtualioje aplinkoje
- Žemėlapio kūrimas naudojant robotus realioje aplinkoje

1.1. Žemėlapio kūrimas modeliuojant virtualioje aplinkoje

1.1.1. Ypatybėmis paremtas žemėlapio kūrimas

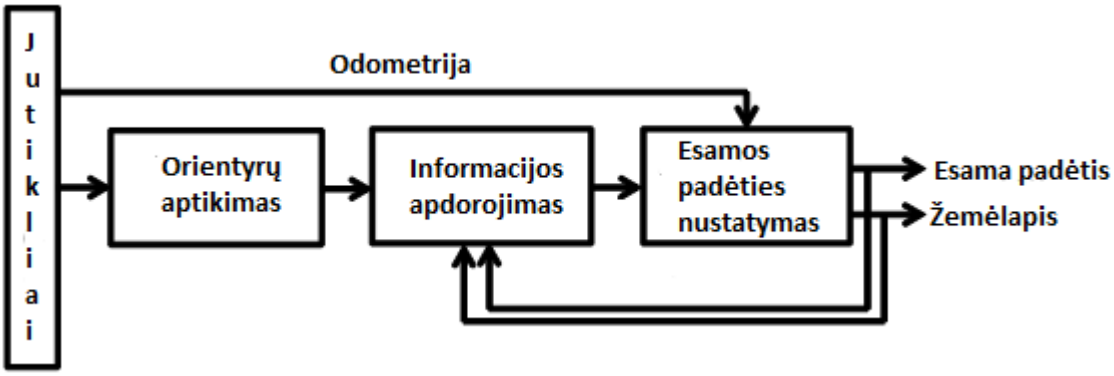
Žemėlapio kūrimui naudojama orientyrai aplinkoje, kurie nurodo robotui kelią. Tarkime robotas juda kaip nurodyta 1.1 pav.



1.1 pav. Roboto judėjimo trajektorija

Pradinė roboto poziciją žymėsime kaip nulinė. Robotas aplinkoje turi aptikti specifinius objektus vadinamus orientyrais, nustatyti atstumą tarp savo pozicijos x_k ir orientyro m_i ir apskaičiuoti kokiu kampu turės judėti. Kiekviename laiko žingsnyje k robotas juda pagal trajektoriją u_{k-1} ir atlieka skaičiavimus z_k pozicijoje x_k [1].

Yra trys užduotys nurodytos 1.2 pav. kurias privalo atitikti ypatybėmis paremtas žemėlapio kūrimas.



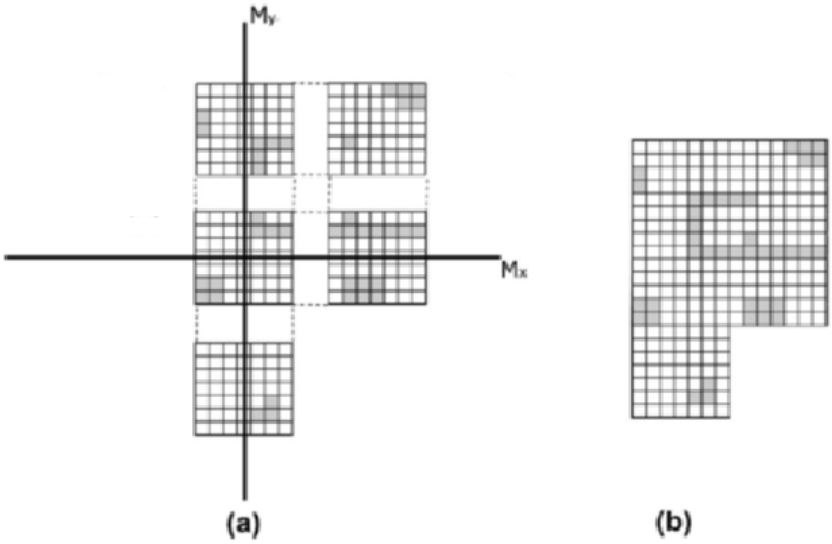
1.2 pav. Užduotys žemėlapio kūrimui

1. Robotas turi atpažinti specifinius objektus, kurie vadinami orientyrais. Orientyrus galima aptikti pasinaudojus lazerio nuotolio jutiklio arba kameros pagalba. Orientyrai gali būti kampai, linijos, medžiai ir t.t.
2. Informacijos apdorojimas. Aptikti orientyrai turi būti susieti su orientyrais žemėlapyje. Kadangi orientyrai nėra skirtingi informacijos apdorojime gali atsirasti didelių klaidų ir iškraipyti žemėlapį.
3. Esamos padėties nustatymas. Svarbiausi parametrai nustatant esamą padėtį yra konvergencija, tikslumas ir nuoseklumas.

1.1.2. Mobilaus roboto žemėlapio sudarymas naudojant skrydžio laiko kameras

Skrydžio laiko (angl. time-of-flight (ToF)) kamera yra naudojama žemėlapiams sudaryti. Kameros veikia vadinamojo skrydžio laiko (time-of-flight) principu: skaičiuojamas laikas, per kurį infraraudonasis spindulys nukeliauja nuo vieno objekto iki kito. Kamera verčia paveikslėlio pikselių (x, y, z) koordinates į žemėlapiui skirtas sudaryti koordinates. Aplinkos žemėlapiui sudaryti yra naudojamas užimtumo tinklelis. Kadangi aplinkos dydis nėra žinomas, negalima iškart sukurti fiksuoto dydžio užimtumo tinklelio. Šiai problema spręsti yra naudojami maži tinkleliai iš kurių, vėliau, yra sudaromas bendras užimtumo tinklas. Visi tinkleliai turi vienodus dydžius ir vienodą langelių skaičių. Kiekvienam tinklelyje langeliai turi užimtumo lygio tikimybę. Jei užimtumo lygio tikimybė yra lygi nuliui langelis yra laisvas, jei užimtumo lygio tikimybė lygi vienetui langelis yra pilnai užimtas [2].

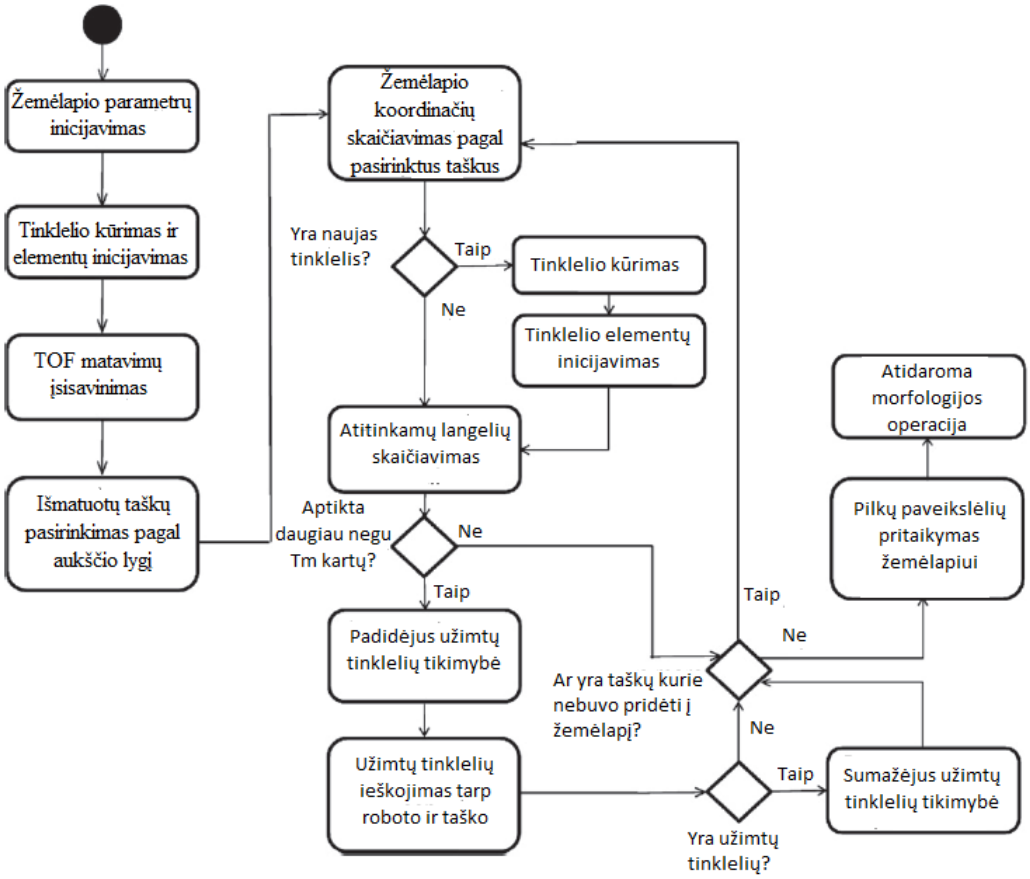
Sudarytas žemėlapis pateiktas 1.3 pav.



1.3 pav. Aplinkos modelis. (a) Naujų tinklelių pildymas į žemėlapį. (b) Gautas aplinkos žemėlapis

Žemėlapio kūrimo algoritmas pavaizduotas 1.4 paveikslėlyje.

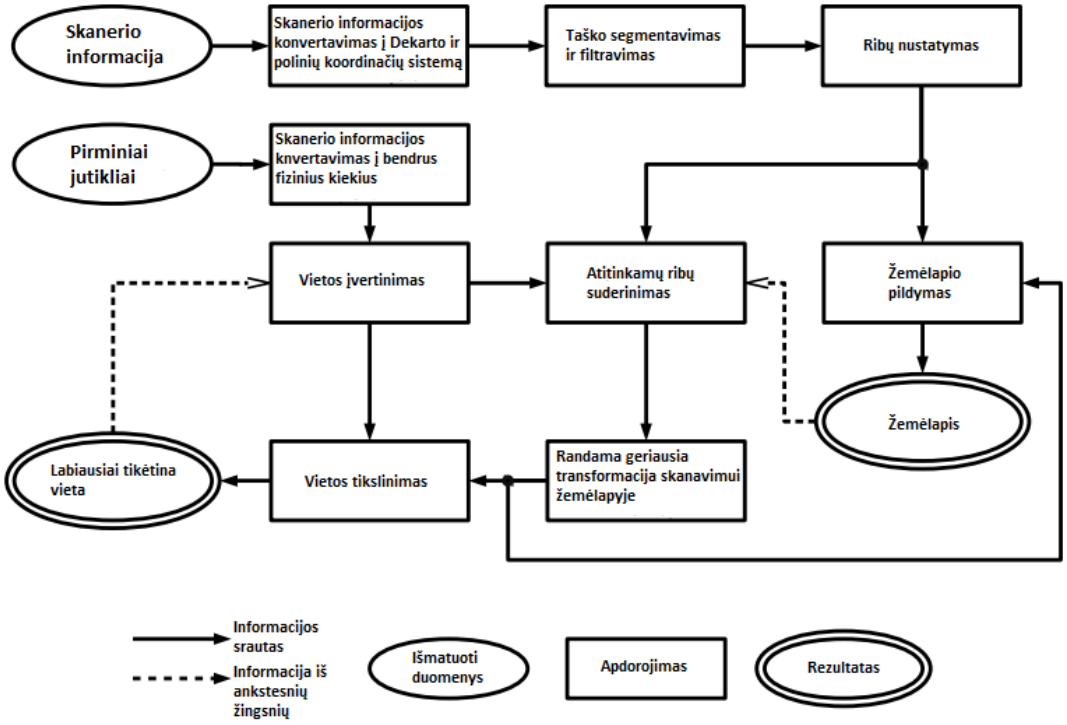
ToF kamera sukuria matricą, kurioje kiekviena pozicija turi savo trijų dimensijų koordinates (x, y, z) metrais. X koordinatė kinta horizontalioje ašyje, y kinta kartu vertikalioje ašyje, z yra atstumas nuo plokštumos, nustatytos pagal X ir Y ašis.



1.4 pav. Žemėlapio sudarymo algoritmas

1.1.3. Vektorinis žemėlapis mobiliajame robote

Visa skaičiavimų ir žemėlapio kūrimo diagrama nurodyta 1.5 paveikslėlyje.



1.5 pav. Skaičiavimo algoritmas

Kiekvienas stačiakampis reiškia tam tikrą algoritmą, kurio pagalba tikslinamas žemėlapis ir roboto pozicija. Pagrindinis tikslas esamoje vietoje yra testuoti informaciją, ribų nustatymus ir skenuoti. Pradžioje visų jutiklių informaciją reikia paversti duomenimis naudingais algoritmui. Naujo taško apdorojimui pirma reikia nufiltruoti triukšmus ir nukrypimus tam, kad nustatyti kuo tikslesnes objektu esančių aplink ribas. Tada nustatomos objektų esančių aplink ribos. Nustačius ribas žemėlapis papildomas nauja informacija. Norint sudaryti tikslų žemėlapį robotui būtina tiksliai žinoti vietą sukurtame žemėlapyje, todėl ši informacija turi pastoviai būti atnaujinama [3].

1.1.4. Nežinomos aplinkos tyrinėjimas naudojant 2D lazerinį skanerį

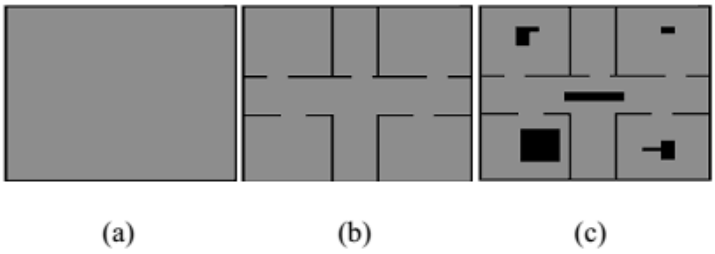
Pagrindinis darbo tikslas yra sukurti nežinomos aplinkos žemėlapį per kuo trumpesnį laiką. Žemėlapis susideda iš kvadratinių elementų, kurie gali būti laisvi, užimti arba nežinomi. Robotai pasirenka nežinomos aplinkos vietą, skenuoja aplinką 180^0 priekį savęs ir renka informaciją [4]. Robotai tirs aplinką pagal šiuos algoritmus:

- Algoritmas Nr. 1 Šiame algoritme robotas atsitiktinai pasirenka vieną iš nežinomų kvadratinių elementų.
- Algoritmas Nr. 2 Šiame algoritme robotas pasirenka priekyje esantį arčiausią nežinomą kvadratinį elementą.

- Algoritmas Nr. 3 Šiame algoritme robotas pasirenka priekyje esantį arčiausią nežinomą kvadratinį elementą, bet roboto pasirinktas elementas negali būti kito roboto skanavimo ribose.
- Algoritmas Nr. 4 Šiame algoritme robotas pasirenka priekyje esantį arčiausią nežinomą kvadratinį elementą, bet roboto pasirinktas elementas negali būti kito roboto pasirinkto elemento skanavimo ribose.
- Algoritmas Nr. 5 Šiame algoritme robotas pasirenka priekyje esantį arčiausią nežinomą kvadratinį elementą, bet roboto pasirinktas elementas negali būti kito roboto pasirinkto elemento skanavimo ribose. Tačiau jeigu pasirinktas elementas buvo nuskenuotas kito roboto jis pakeičia kryptį ir ieško kito elemento.

Tyrimo algoritmai išbandomi naudojantis programomis Wilensky, Netlog. Aplinka yra susideda iš m-n tinklelių, kurie sudaryti iš kvadratinių elementų [4]. Kiekvienas elementas turi savo reikšmę.

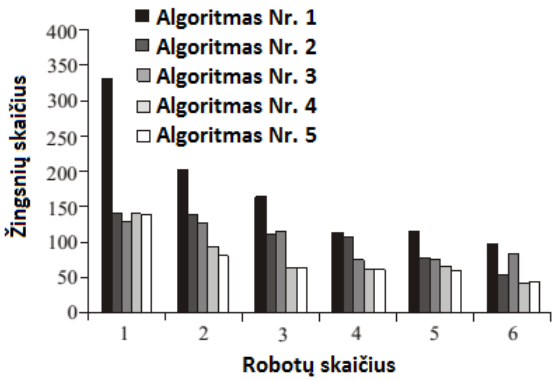
Tiriamieji aplinkos modeliai pateikti 1.6 paveikslėlyje.



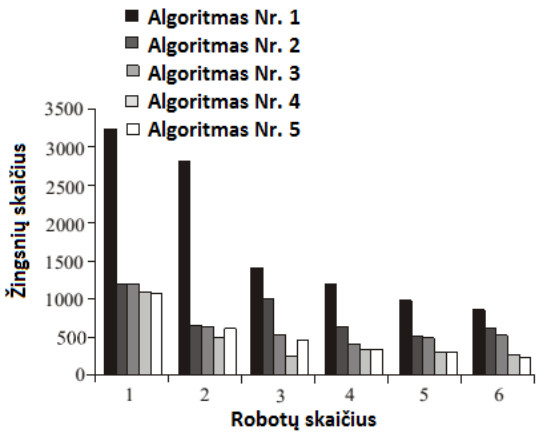
1.6 pav. Aplinkos modeliai

Tyrimo rezultatai pateikiami lyginant algoritmų tyrimo laikus (žingsnių skaičius)

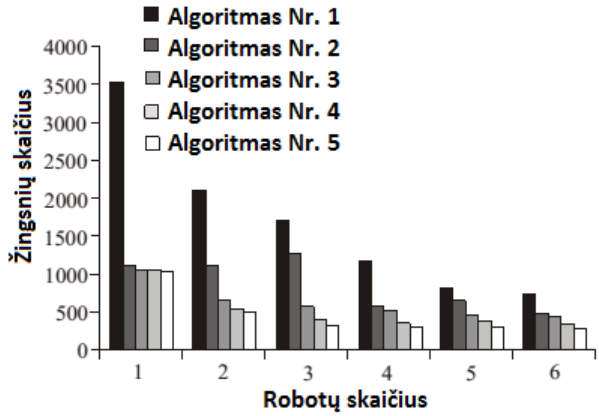
Tiriant aplinką nurodytą paveiksle 1.6, (a), (b), (c) dalyse laikas (žingsnių skaičius) ir robotų skaičius nurodyti 1.7, 1.8, 1.9 paveiksluose.



1.7 pav. Tyrimo rezultatai pagal 1.6 pav. a dalį



1.8 pav. Tyrimo rezultatai pagal 1.6 pav. b dalį



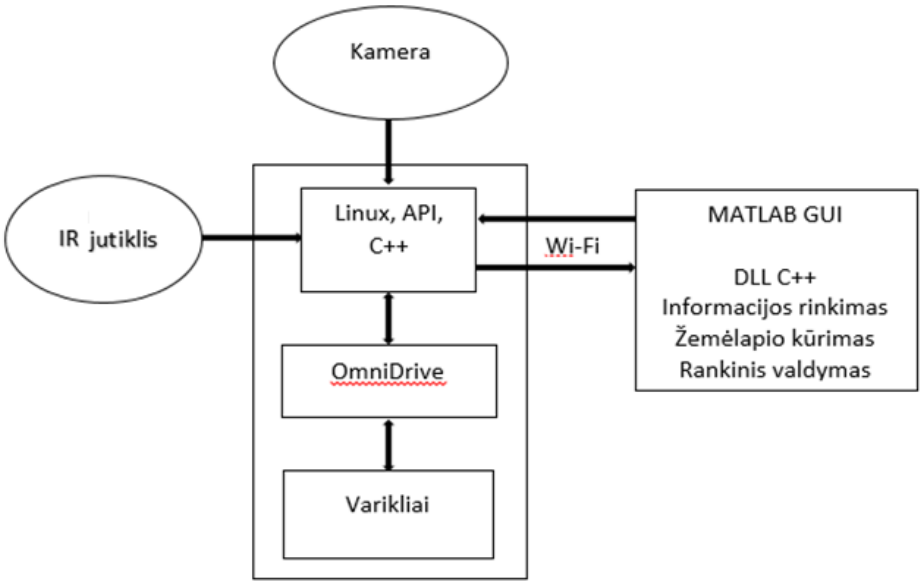
1.9 pav. Tyrimo rezultatai pagal 1.6 pav. c dalį

Iš tyrimo rezultatų matyti jog daugiausiai laiko užima pirmasis algoritmas, kai skenuojama pasirenkat atsitiktinai. Mažiausiai laiko užima kai robotas pasirenka priekyje esantį arčiausią nežinomą kvadratinį elementą, bet roboto pasirinktas elementas negali būti kito roboto pasirinkto elemento skanavimo ribose [4].

1.2. Žemėlapio kūrimas naudojant robotus realioje aplinkoje

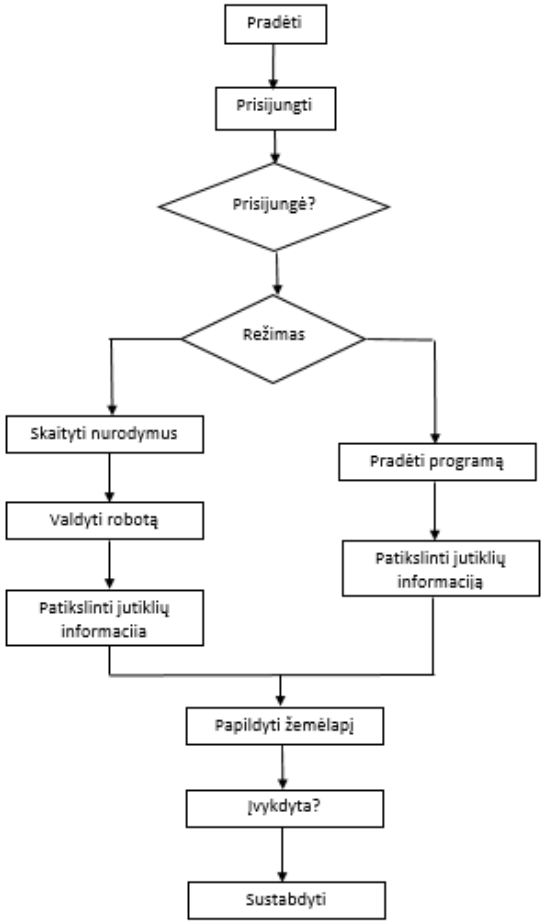
1.2.1. Mobilaus roboto žemėlapio sudarymas naudojant Festo Robotino mobilų robotą

Žemėlapiams sudaryti nežinomoje aplinkoje galima naudoti Festo Robotino mobilų robotą. 1.10 paveikslėlyje nurodyti techninė įranga naudojama žemėlapių sudarymui. Informacija surinkta iš jutiklių ir kameros siunčiama į kompiuterį, kuriame formuojamas žemėlapis [5].



1.10 pav. Principinė įrangos schema

Žemėlapio sudarymo algoritmas, kuris nurodytas 1.11 pav., yra paremtas Matlab programa, kuri veikia personaliniame kompiuteryje. Programa gauna informaciją apie Festo Robotino esamą vietą ir aplinką, bei siunčia duomenis, kuriuose nurodomi veiksmai kaip elgtis robotui. Atstumo informacijai yra naudojami IR jutikliai, kurie neleidžia atsitrenkti į kliūtis, bei fiksuoja informaciją apie aplinką. Festo Robotino robotas turi kamera, esančia roboto priekyje. Kameros pagalba galima matyti vaizdinę medžiagą roboto priekyje [5].



1.11 pav. Žemėlapio sudarymo algoritmas

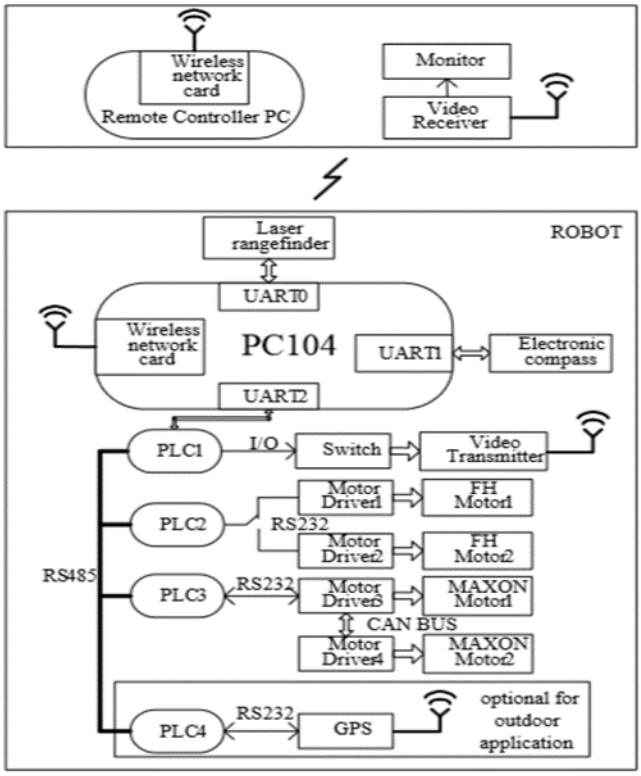
1.2.2. Autonominio mobilaus roboto žemėlapio sudarymas naudojant lazerinį nuotolio skenerį ir elektrinį kompasą

Šiame metode mobilus robotas nežinomoje aplinkoje žemėlapį sudaro iš tinklelių lazerinio nuotolio skaitytuvo ir elektroninio kompaso pagalba. 2D lazerinis nuotolio skaitytuvas URG-04LX įmontuotas roboto priekyje kaip pagrindinis duomenų surinkimo įrankis. Roboto viduje įmontuotas PC104 valdiklis operacijoms atlikti. Mobilaus roboto gale pasirinktas elektrinis kompasas XW-EC1700, kuris leidžia robotui orientuotis aplinkoje. Infraraudonųjų spindulių jutikliai išdėstyti žiedu aplink robotą užtikrina kliūčių išvengimą, trys kameros išdėstytos skirtingose vietose leidžia matyti realų vaizdą vartotojui. Robote taip pat yra įmontuotas ir GPS imtuvas-siųstuvas [6].

Roboto sistemoje vyrauja hierarchinė valdymo struktūra. PC104 naudojamas kaip centrinis valdiklis į kurį siunčiama visų jutiklių informacija. Programuojami valdikliai (PLC) esantys roboto gale atlieka skirtingas užduotis. PLC1 yra atsakingas už komunikaciją tarp PC104 ir žemesniame lygyje esančių valdiklių. PLC2 valdo varomuosius variklius ir infraraudonųjų spindulių pagalba

stebi aplinkoje esančias kliūtis. PLC3 yra atsakingas už variklius padėsiančius robotui kopti į viršų arba leistis žemyn. PLC4 gauna GPS informaciją dirbant lauke [6].

Roboto struktūrinė schema nurodyta 1.12 paveikslėlyje.



1.12 pav. Struktūrinė roboto schema

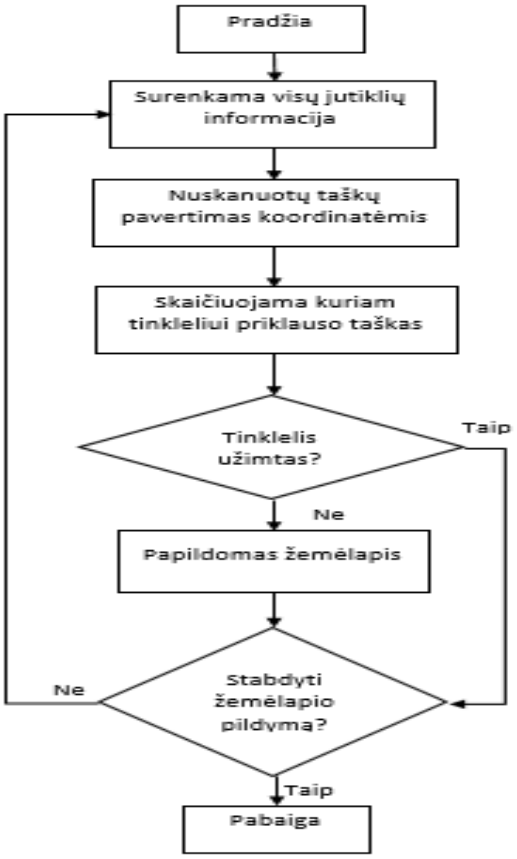
Lazerinis nuotolio skaitytuvas Hokuyo URG-04LX yra mažas, įperkamas ir tikslus. Veikimo diapazonas nuo 20 mm. Iki 5600 mm (rezoliucija 1 mm.). Matuoja 240^0 kampu (rezoliucija $0{,}36^0$). Duomenys yra siunčiami per RS232 arba USB sąsają.

Infraraudonųjų spindulių davikliai GP2Y0A02YK yra optoelektriniai prietaisai skirti kliūtims fiksuoti. Jų matavimo diapazonas yra nuo 20 cm iki 150 cm.

Žemėlapio tinklelio sudarymas procesas sudaromas sekančia tvarka:

1. Visų nuskenuotų taškų vertimas koordinatėmis. Kiekvienos nuotolinio lazerio nuskenuotos koordinatės turi būti paverčiamos į vietinę koordinačių sistemą. Pavertus tikrinama ar nuskenuotas taškas nepriklauso jau užpildytam tinkleliui žemėlapyje.
2. Skaičiuojama kuriam tinkleliui priklausys naujai nuskenuotas taškas.
3. Skaičiuojamas kelias kliūtims išvengti.

Žemėlapio papildymo algoritmas nurodytas 1.13 paveikslėlyje.

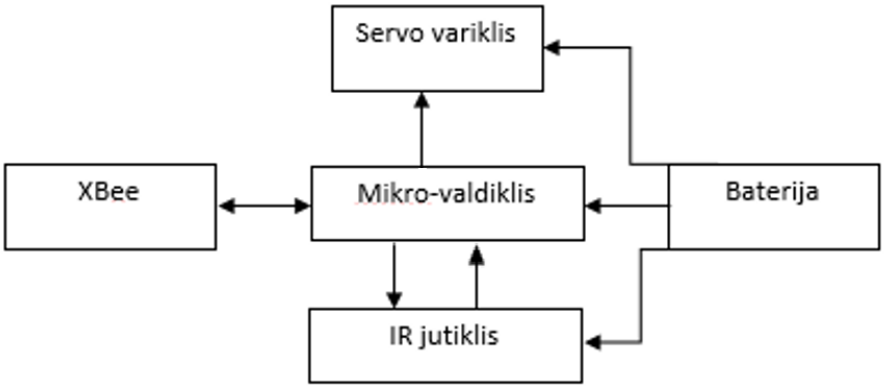


1.13 pav. Žemėlapio papildymo algoritmas

1.2.3. Mobilaus roboto žemėlapio kūrimas naudojant IR jutiklius

Šis metodas skirtas mobiliems robotams uždaroje aplinkoje. Robotas susideda iš 2 Xbee modulių, kurie siunčia ir gauna informaciją iš ir į kompiuterį, 3 IR jutiklių, kurių pagalba seka objekto kraštus ir pagal nuvažiuoto kelio trajektorijas sudarinėjamas žemėlapis kompiuteryje. Robotas važiuoja servo variklių pagalba. Pradinės koordinatės nustatomos rankiniu būdu. Žemėlapio sudarymas yra paremtas tinklelių kūrimo principu [7].

Roboto struktūrinė schema nurodyta 1.14 paveikslėlyje.



1.14 pav. Struktūrinė mobilaus roboto schema

Jutikliai yra visada įjungti kai žemėlapis yra sudarinėjamas ir jų reikšmės nuskaitomos robotui atlikus bent kokį judesį. Atstumas iki kliūčių yra matuojamas penkis kartus ir naudojamas gautų reikšmių vidurkis. Užfiksavus vidurkį jo reikšmė talpinama į žemėlapį.

Žemėlapio sudarymas vyksta kartojant septynis žingsnius.

1. Nuskaitoma jutiklių parametrai. Reikšmės keliamos į žemėlapį.
2. Jeigu nėra kliūčių mobilus robotas pasisuka 45^0 kampu aplink savo ašį. Po kiekvieno judesio jutiklių parametrai yra nuskaitomi ir jei yra užfiksuota kliūtis programa iš naujo atlieka pirmą žingsnį. Kliūties neužfiksavus robotas vėl sukasi aplink savo ašį 45^0 kampu ir skenuoja aplinką. Taip kartojama kol grįžtama į pirminę poziciją. Tada robotas juda į priekį po 5 cm kol aptinka kliūtį ir programa atlieka pirmą žingsnį.
3. Jei kliūties atstumas kairėje pusėje yra mažesnis nei 6 cm robotas pasisuka į dešinę 45^0 kampu ir programos žingsniai atliekami iš naujo. Jei sąlygos nėra tenkinamos programa, eina į 4 žingsnį.
4. Jei kliūties atstumas dešinėje pusėje yra mažesnis nei 6 cm robotas pasisuka į kairę 45^0 kampu ir programos žingsniai atliekami iš naujo. Jei sąlygos nėra tenkinamos programa, eina į 5 žingsnį.
5. Jei kliūties atstumas dešinėje pusėje yra didesnės nei 15 cm arba yra kliūčių kairėje pusėje ir priekyje, mobilus robotas pasisuka 90^0 į dešinę ir programa atliekama nuo pirmo žingsnio. Jei tai pasikartoja programa atlieka 6 žingsnį.
6. Jei kliūties atstumas kairėje pusėje yra didesnės nei 15 cm arba yra kliūčių dešinėje pusėje ir priekyje, mobilus robotas pasisuka 90^0 į kairę ir programa atliekama nuo pirmo žingsnio. Jei tai pasikartoja programa atlieka 7 žingsnį.
7. Mobilus robotas pajuda 5 cm į priekį.

Atlikus nežinomos aplinkos žemėlapio sudarymo darbų analizę pastebėta, kad yra opių problemų, kurios nėra iki galo išspręstos. Mokslininkai tiria žemėlapio sudarymo nežinomoje aplinkoje galimybę tiek dvimatėje, tiek trimatėje aplinkoje. Tolimesnis darbo tyrimo tikslas yra išanalizuoti ar spalvoti Petri tinklai gali būti naudojami problematikos sprendimui t.y. nežinomos aplinkos žemėlapio sudarymui dvimatėje virtualioje erdvėje.

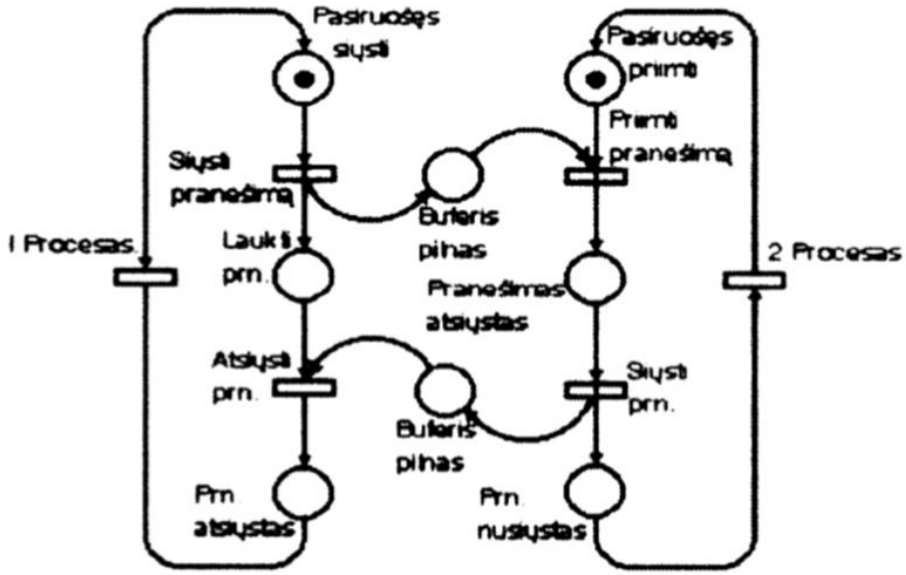
2. Tyrimų dalis

Petri tinklai – viena iš keleto matematinio modeliavimo kalbų, skirta paskirstytųjų sistemų aprašymui. Petri tinklas yra nukreiptas dviejų dalių grafas, kuriame mazgai atitinka perėjimus (t. y. įvykiai kurie gali įvykti - pažymėti stulpeliais) ir vietas (t. y. sąlygos - pažymėtos apskritimais). Rodyklės parodo, kurios vietos yra išankstinės sąlygos ir/ar posąlygos. Petri tinklus išrado Carl Adam Petri 1939 m. rugpjūtį, būdamas 13 metų norėdamas apibūdinti cheminį procesą [8].

Jų tolimesnis vystymasis vyko kilus poreikiui nustatyti procesų sinchronizaciją, asinchroninių ir lygiagrečių procesų radimą, konfliktinių situacijų nustatymą, o taip pat tikslu nustatyti bendrų resursų panaudojimą diskretinio veikimo sistemose. Ankstyvuosius ar pradinius Petri tinklų tyrimus atliko kompiuterių specialistai. Nuo 1970 metų Petri tinklų panaudojimu pramoninei automatikai susidomėjo specialistai, kurie turi inžinerinį išsilavinimą. Pastarieji ieškojo jų pritaikymo sistemoms „žmogus mašina“, nes pramoninės automatikos tolydinės bei diskretinės sistemos tapo tiek sudėtingos, kad esama tokių sistemų tyrimo metodika jau nesugebėdavo jų tinkamai analizuoti. Bet koks tinklas susideda iš dviejų pagrindinių elementų: mazgų ir ryšių. Petri tinklai modeliuoja sistemos būsenas. Ryšiai čia vaidina svarbų vaidmenį keičiant vieną būseną kita. Reikia pažymėti, kad pereigos vaizduoja įvykius sistemoje, o ryšiai atlieka valdančios informacijos ar materialų srautų judėjimą. Pavyzdžiui, roboto techninėje sistemoje, pradinė būklė bus, kai robotas yra pasiruošęs paimti gaminio komponentes, o komponentės yra paruoštos transportavimui. Kai įvykis startuos, galimos dvi situacijos: robotas paima komponentes ir yra pasiruošęs jas transportuoti arba robotas nesugeba ar negali paimti komponenčių. Visą tai nustato galimos dvi roboto būsenos – laisvas ar užimtas. Reikia pabrėžti, kad būsenos ir įvykiai leidžia labai puikiai nustatyti konfliktines situacijas, tai visos kitos problemos, kaip operacijų sinchronizacija, lygiagrečios operacijos ir draudžiamų veiksmų nustatymas nėra aiškiai gaunamas.

Pozicijos, perdavimo ir kryptiniai ryšiai Petri tinklą organizuoja kaip orientuotą grafų sistemą, kuri ir yra vadinama Petri tinklo struktūra. Tai įgalina panaudoti Petri tinklus struktūrinių schemų modeliavimui. Sistemos būsena yra nustatoma „būsenos žymėmis“, kurios yra fiksuojamos Petri tinklo pozicijose. Būsenos žymės yra vadinamos žetonais. vaizdavime pozicijos, kurios turi tokius žetonus, turi vieną iš priimtų žymėjimų. Istoriškai pirminiai žymėjimai buvo taškai, kur vieną žetoną atitinka vienas taškas, vėliau žymėjimas modifikavosi į žymėjimą skaitmenimis. Kai visos pozicijos prijungtos ryšiais prie pereigos įėjimų turi pakankama kiekį žetonų, vyksta pereigos atidarymas. Pereiga to rezultate generuoja per išeinančius ryšius žetonus kurie perduodami į atitinkamas pozicijas, pakeisdami esamą sistemos būseną kita būsena. Aukščiausias Petri tinklų modifikavimas pasiektas pasirodžius spalvotiems Petri tinklams[15].

Petri tinklų panaudojimo pavyzdys pateiktas 2.1 paveiksle.



2.1 pav. Petri tinklų panaudojimo pavyzdys.

Valdymo sistemų modeliuojant spalvotais Petri tinklais privalumai

Egzistuoja daugybė modeliavimo kalbų, kurias tiesiogiai palyginti būtų labai sunku arba kartais net neįmanoma. Žymiai lengviau atlikti netiesioginę analizę, kurios lemia jų taikymo privalumus projektuojant, specifikuojant ir analizuojant skirtingų tipų sistemas. Tačiau reiktų įžvelgti kitų modeliavimo kalbų savybes, kurios tinka nurodytiems reikalavimams tenkinti, kurių neturi spalvoti Petri tinklai. Štai kokius privalumus galima pastebėti naudojant spalvotuosius Petri tinklus:

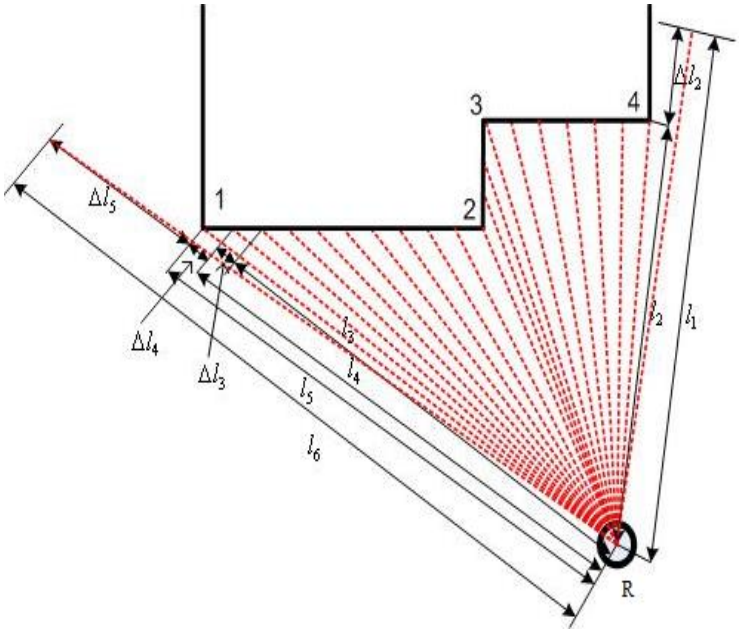
1. Spalvotieji Petri tinklai gali būti atvaizduoti grafiškai. Grafinė forma visuomet patraukli, nes ją lengva suprasti net žmonėms, neturintiems pakankamai žinių apie šio tipo tinklus. Taip yra dėl to, kad jie yra panašūs į dauguma neformaliųjų brėžinių, kuriuos naudoja projektuotojai ar inžinieriai kurdami bei analizuodami sistemą. Paprasta atvaizduoti sistemą, kaip būsenų, veiksmų ir srautų visumą, kurių analogai yra pozicijos, pareigos ir briaunos.
2. Spalvotieji Petri tinklai turi aiškiai apibrėžtą semantiką, kuri tiksliai nusako kiekvieną jų veikimą. Tokias semantiką leidžia konstruoti spalvotųjų Petri tinklų imitatorius ir kitus formaliosios analizės metodus.
3. Spalvotieji Petri tinklai yra labai bendri ir leidžia aprašyti daugybę skirtingų sistemų. Jie leidžia apibrėžti formaliąsias ir neformaliąsias sistemas, o jų taikymas apima programų sistemas, techninę įrangą. Spalvotieji Petri tinklai taip pat tinka sistemoms, turinčioms daug lygiagrečiai vykstančių procesų, aprašyti ir naudojami netgi šnekamajai kalbai analizuoti.

4. Spalvotieji Petri tinklai turi keletą galingų primityvų. Pavyzdžiui apibrėžimas yra nesudėtingas ir sudarytas iš gerai žinomų matematikos ir programavimo sąvokų leidžiančių juos greitai perprasti, sudaryti ir analizuoti.
5. Jie turi aiškų būsenų ir veiksmų aprašą. Skirtingai nuo daugelių sistemų aprašymo kalbų, vienu metu galinčių aprašyti tik būsenas arba veiksmus, čia tai galima atlikti iškarto arba atskirai.
6. Spalvotieji Petri tinklai turi semantiką, kuri remiasi tikru konkurentiškumu. Čia sąvokos „konfliktas“ ir „konkurentiškumas“ suprantamos labai paprastai ir natūraliai.
7. Galimas hierarchiškumas t.y. didelio tinklo sudarymas iš potinklių. Tai labai vertinga savybė projektuojant dideles ir sudėtingas sistemas.
8. Spalvotieji Petri tinklai apima valdymą, sinchronizavimą, manipuliavimą duomenimis. Bendroje erdvėja yra modeliuojama aplinka, veiksniai, sąlygos ir rezultatai. Dauguma kitų kalbų naudoja grafus, kurie modeliuoja veiksmų aplinką, o detalės nurodomo atskira.
9. Juose gali būti panaudota laiko sąvoka. Įmanoma panaudoti tą pačią modeliavimo kalbą specifikuojant funkcines ir veikimo savybes. Čia naudojamas globalinis laikas, kur kiekvienas žetonas be duomenų turi ir laiko arba trukmės ženklą. Šis laiko ženklas parodo kada žetonas yra pasiruošęs įveikti pereigą.
10. Spalvotieji Petri tinklai išlieka stabilios struktūros nežymiai keičiantis modeliuojamai sistemai. Praktika rodo, kad nedideli sistemos pakitimai beveik nepakeičia šių tinklų struktūros.
11. Jie leidžia atlikti interaktyvią imitaciją, kurios metu rezultatai pateikiami tiesiogiai spalvotųjų Petri tinklų diagramoje. Imitacija nesunkiai leidžia suderinti didelę ir sudėtingą sistemą. Nesudėtinga stebėti ir keisti judančių žetonų „pernešamą“ informaciją.
12. Spalvotieji Petri tinklai turi daug formaliųjų analizės metodų, kuriuos taikant galima įrodyti jų savybes. Yra keturi formaliosios analizės metodai: įvykių grafo konstravimas, sistemos variantų skaičiavimas ir interpretavimas, prastinimas, struktūrinių savybių patikra.
13. Spalvotieji Petri tinklai turi kompiuterinių įrankių, leidžiančių konstruoti, imituoti ir analizuoti. Tai suteikia galimybę valdyti net ir didelius tinklus, nedarant klaidų.

Dauguma šių spalvotų Petri tinklų privalumų galioja ir kitiems aukšto lygio tinklams ar modeliavimo kalboms. Tačiau juos taikant kartu su kitomis modeliavimo kalbomis t.y. vienas iš priimtinesnių būdų modeliuoti įvairaus lygio sistemas

2.1. Vektorinių žymių generavimo metodas

Pagal siūlomą metodą, aplinka apžvelgiama skeneriu. Skenavimo tikslas – aptikti statinių kliūčių pokyčius, fiksuojant statinių kliūčių kampų taškus, kurie toliau vadinami trūkio taškais. Aptinkant statinių kliūčių trūkio taškus fiksuojami statinių kliūčių kampai, angos ir kiti staigūs konfigūracijos pakitimai. Taigi, skenavimas pradedamas iš roboto esamos padėties. Skeneriui užduodamas skenavimo kampo dydis, bei skenerio skenavimo žingsnio dydis. Skenavimo kampo žingsnio dydžio parinkimas labai svarbus. Jei šis žingsnis bus pasirinktas per didelis, skeneris gali neaptikti kai kurių statinių kliūčių, taigi logiška šį žingsnį parinkti kuo mažesnį. Skeneriu skenuojant aplinką nustatytu kampu ir skenavimo žingsniu, aptinkamos statinės kliūtys, kurias reikia fiksuoti. Siūlomame metode įvedama sąvoka – trūkio taškas. Trūkio tašku vadinamas taškas, kuris aptinkamas skenuojant aplinką su kliūtimis ir jis sutampa su kraštiniu statinės kliūties briaunos tašku. Tačiau ne visi kraštiniai kliūčių taškai gali būti trūkio taškais. Trūkio taško ypatybė ta, kad už šio taško skenuojamoje aplinkoje iki sekančios bet kokio tipo kliūties yra skenerio spindulio ilgio šuolis, kuris nustatyta verte didesnis arba mažesnis už prieš tai, skenuojant nustatytais žingsniais, gautus atstumus [12]. Ši situacija iliustruojama 2.2 paveiksle.

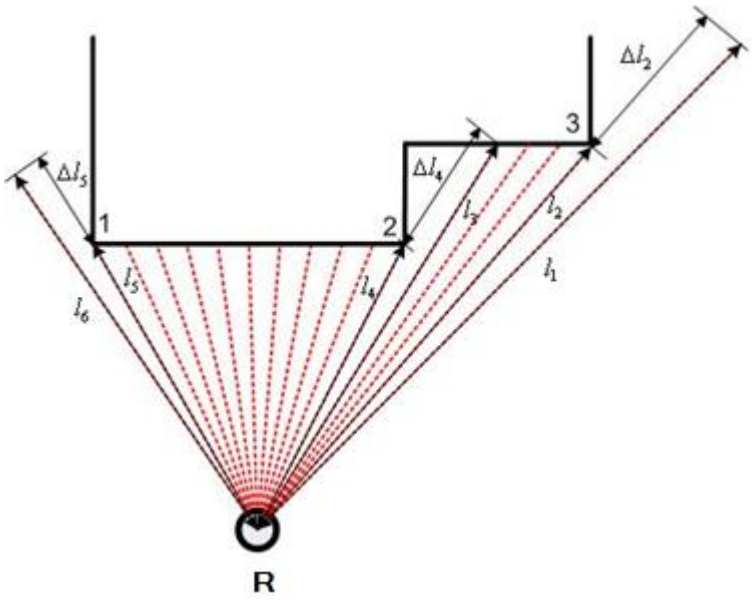


2.2 pav. Trūkio taškų radimas

2.2 paveiksle vaizduojamu atveju skenuojama aplinka, kurioje yra statinė kliūtis, turinti 4 kampus. Iš esamos padėties taško R, skenuojant aplinką nustatytu skenerio kampu α, pastoviu skenerio kampo žingsniu $\Delta\alpha$, atstumas iki kliūties yra l_1. Sekančiame skenerio žingsnyje atstumas iki kliūties yra l_2. Taigi, skenerio spindulio ilgio skirtumas yra Δl_2. $l_2 = l_1 - \Delta l_2$, o dydžio Δl_2 vertė didesnė nei nustatytoji galima ilgio paklaidos vertė Δl. Esant tokiai situacijai, kai $\Delta l_2 > \Delta l$, statinės

kliūties taškas „4“ yra trūkio taškas. Tęsiant tolesnį skenavimą, gaunami skenerio spindulių l_3, l_4 ir l_5 ilgių šuoliai Δl_3 ir Δl_4 savo vertėmis neviršija nustatytojo dydžio Δl. Sekančiame skenerio žingsnyje už skenerio spindulio ilgio l_5 gaunamas skenerio spindulio ilgio šuolis Δl_5. Gaunamas skenerio spindulio ilgis $l_6 = l_5 + \Delta l_5$. Dydis Δl_5 savo verte didesnis nei nustatytoji dydžio Δl vertė. Taigi, $\Delta l_5 > \Delta l$, o taškas „1“ (2.2 pav.) yra trūkio taškas. 2.2 paveiksle vaizduojamu atveju, vidiniai kliūties kampai, pažymėti „2“ ir „3“, nėra trūkio taškai.

Kintant roboto vietai, gali kisti trūkio taškų kiekis. 2.3 paveiksle iliustruojama analogiška statinė kliūtis kaip ir prieš tai nagrinėtame 2.2 paveiksle, tačiau, lyginant su 2.2 paveikslu, roboto esamos padėties taško R padėtis statinės kliūties atžvilgiu. Remiantis 2.2 paveiksle aprašytu metodu, nustatoma, jog už visų trijų statinės kliūties kampų (2.3 pav.) gaunami skenerio spindulio ilgių šuoliai.



2.3 pav. Trūkio taškų kiekio skirtumas

Pastebima, jog abiem atvejais trūkio taškai randami, tačiau jų kiekis priklauso nuo roboto padėties statinių kliūčių atžvilgiu. 2.2 ir 2.3 paveiksluose nagrinėtose situacijose statinės kliūtys stačiakampio ar kvadrato formos. Analogiška trūkio taškų paieška yra ir trikampio, rombo ar trapecijos formos statinėms kliūtims.

2.2. Vektorinių žymių savybės

Pastebima, kad vien trūkio taškų ar liestinių, norint nuskenuoti aplinką su kliūtimis, nepakanka. Reikia žinoti kryptį, kuria sekant galima patekti iš išeities taško į tikslą. Trūkio taškai

ir liestinės palengvina statinių kliūčių kampų radimą, tačiau dar reikalingas ir krypties nustatymas. Idėja formuluojama taip: skenuojama aplinka ir randami trūkio taškai, kuriuose žymima kryptis.

Įvedama sąvoka – vektorinė žymė. Virtualus vektorius, orientuotas išilgai artimiausio kelio fragmento, kurio viršūnė yra ties kliūties kampo viršūne (trūkio tašku) arba lietimosi taške, vadinamas vektorine žyme. Kitaip tariant, tai vektorių aibė, nukreipta išilgai kelio link roboto pradinės pozicijos galimo kelio posūkių taškuose. Iš dalies, šios žymės yra ne kas kita kaip krypties vektoriai. Tiesioginės krypties vektoriais gali būti vadinamos tik pirmosios žymės, nes jos tiesiogiai nukreiptos į roboto pradinės pozicijos tašką. Visos kitos vektorinės žymės, atsižvelgiant į statines kliūtis, nurodo kryptį, kuria sekant, roboto pradinės pozicijos taškas pasiekiamas trumpiausiu keliu [12].

Kliūtimis užimta sritis, kaip taškų visuma, pažymima S_{obst} , o laisva sritis - S_{free}, subsritis, kurios visi taškai yra matomi iš viršūnės ar lietimosi taško su kliūtimi, priklausanti keliui, $-S_v$. Aplinka suskaidyta k suberdvėmis, susietomis su k taškų, jei išpildoma sąlyga:

$$S_{free} = \bigcup_{n=1}^{k} S_{vn} \qquad (1)$$

Norint apibūdinti vektorinių žymių savybes, apibūdinamas “matomumas” – taškas p_d yra matomas iš taško p_0, jei egzistuoja tiesės atkarpa, jungianti šiuos taškus ir visi šios linijos taškai priklauso S_{free}:

$$\forall p \in l(p_0, p_d) \in S_{free} \qquad (2)$$

$l(p_0, p_d)$ – reiškia tiesę, jungiančią taškus p_0 ir p_d

Vektorinių žymių savybės ir matomumas:

1. Visos vektorinės žymės yra erdvėje S_{free}.
2. Bet kuri vektorinė žymė yra bitangentinio linijinio kelio fragmento pratęsimas.
3. Žymė matoma, jei matomas bent vienas žymės taškas.
4. Iš bet kurio S_{free} taško matoma bent viena vektorinė žymė. Ši savybė siejasi su (1) sąlyga.

Įrodyta, kad erdvė gali būti išskaidyta į iškiliuosius daugiakampius, suberdves. 1 vektorinių žymių matomumo sąlyga išpildoma, jei kiekvienoje suberdvėje yra nors viena vektorinė žymė. Iškiliųjų daugiakampių skaičius ir forma priklauso nuo S_{free} erdvės konfigūracijos – kampai, durys ir kt. Dalis S_{free} yra nematoma iš taško p_0, bet gali būti rasta skenuojant erdvę iš šio taško, bei

fiksuojant matomumo nutrūkimus – ilgio $l(p_0, p_d)$ šuolius. Vektorinių žymių rinkinys susiejamas su roboto pradinės pozicijos tašku ir struktūriškai apibrėžiamas kaip tvarkingas medis, nes yra tikimybė, kad egzistuoja keli optimalūs keliai, priklausomai nuo pradinio roboto taško, vedantys į tą patį tikslą. Vektorinė žymė turi būti nustatyto ilgio.

2.3. Vektorinių žymių svorinis koeficientas

Vektorinės žymės svoriniu koeficientu nustatomas suminis atstumas nuo vektorinės žymės galo taško iki roboto pradinės pozicijos taško. Kitaip tariant, svorinis koeficientas yra atstumas nuo roboto pradinės pozicijos taško iki vektorinės žymės galo.

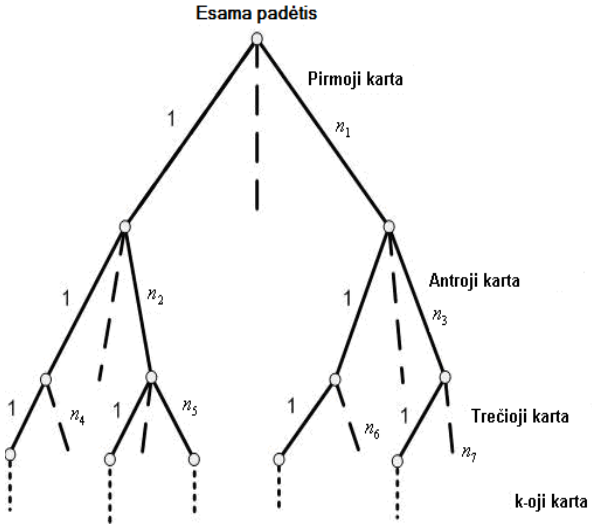
Pirmosios, iš pradinio roboto taško suformuotos, vektorinės žymės yra vyriausios kartos. Tolesnės vektorinės žymės yra jaunesniųjų kartų. Jaunesniųjų kartų vektorinių žymių svorinis koeficientas bus lygus vyresniųjų kartų žymių svorinių koeficientų ir atstumo iki jaunesnės kartos vektorinės žymės sumai. T.y.

$$\begin{aligned} D_1 &= D_{v1} + dd_1, \\ D_2 &= D_1 + D_{v2} + dd_2, \\ D_3 &= D_2 + D_{v3} + dd_3, \end{aligned} \tag{3}$$

$$D_i = D_1 + \sum_{l=2}^{i} (D_{vn} + dd_n), \quad i=2, 3, ...; \; n=2, 3, ...; \tag{4}$$

čia D_1 yra pirmosios vektorinės žymės svorinis koeficientas, D_{v1} – atstumas nuo pirmosios vektorinės žymės viršūnės taško iki roboto pradinės pozicijos taško, dd_1 - pirmosios vektorinės žymės ilgis, D_{v2} – atstumas nuo pirmosios kartos vektorinės žymės galo taško iki antrosios kartos vektorinės žymės viršūnės taško, dd_2 – antrosios vektorinės žymės ilgis, D_{vn} – *n*-tosios vektorinės žymės ilgis nuo jos viršūnės iki vyresnės kartos vektorinės žymės galo taško, dd_n – *n*-tosios vektorinės žymės ilgis, D_i- *i*-tosios vektorinės žymės svorinis koeficientas – atstumas iki roboto pradinės pozicijos taško.

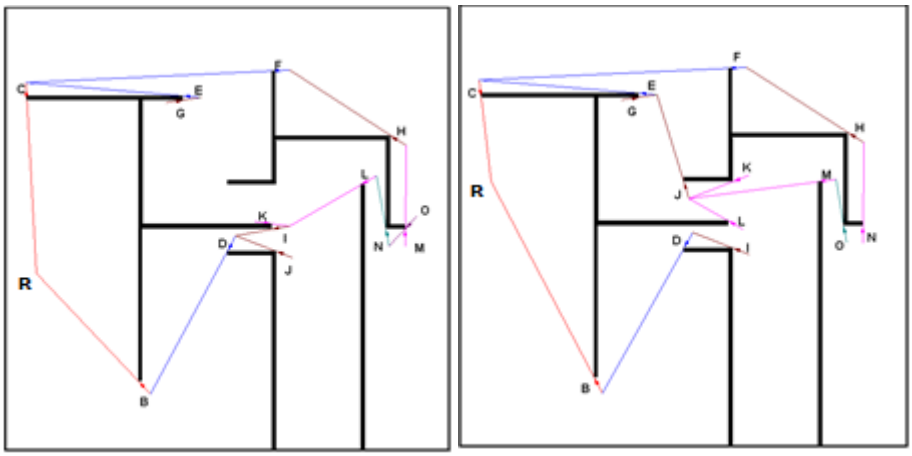
Vektorinių žymių medis sudaro žymių ryšius viena su kitomis. Pirmosios vektorinės žymės, kurios prasideda iš pradžios taško, yra vyriausios kartos. Kitos vektorinės žymės yra jaunesniųjų kartų. Pradedant esamos padėties tašku formuojama medžio struktūra, kuri vaizduojama 2.4 paveiksle.



2.4 pav. Vektorinių žymių medžio struktūra

2.4 paveiksle vaizduojama, kad vektorinių žymių medžio struktūrą gali sudaryti *k* kartų, kurių kiekis priklauso nuo skenuojamos teritorijos sudėtingumo. Kiekviena žymė turi savo svorinį koeficientą. Kaip jau minėta, šis svorinis koeficientas yra suminis atstumas iš vektorinės žymės galo iki roboto pradinės pozicijos taško. 2.4 paveikslas vaizduoja bendrąjį vektorinių žymių medžio atvejį.

Suformavus vektorinių žymių medį galima pasiekti tikslą iš bet kurios nagrinėtos aplinkos vietos. Taigi vektorinės žymės yra orientavimosi aplinkoje priemonė, rodanti trumpiausią kelią į roboto pradinės pozicijos tašką iš bet kurios nagrinėjamos aplinkos vietos. Keičiantis roboto pradinės pozicijos taško vietai, kinta ir vektorinių žymių medžio struktūra. Gali būti, kad esant vienai roboto pradinės pozicijos taško vietai bus pasirenkamos vienos vektorinės žymės, o esant kitai roboto pradinės pozicijos taško vietai pasirenkamos visai kitos vektorinės žymės. 2.5 paveiksle vaizduojamoje aplinkoje, keičiantis roboto R vietai, kinta vektorinių žymių medžio struktūra.



a) b)

2.5 pav. Skirtingos vektorinių žymių struktūros

2.5 paveiksle vaizduojami tokios pačios aplinkos atvejai, kuriuose keičiasi roboto esamos padėties R vieta. Paveiksle vaizduojama vektorinių žymių medžio struktūra. Lyginant 2.5 paveikslo a ir b dalyse vaizduojamus vektorinių žymių medžius, pastebima, kad pirmosios ir antrosios kartos vektorinės žymės abiem atvejais suformuojamos panašiai. Iš roboto pradinės pozicijos taško R, tuose pačiuose trūkio taškuose suformuojamos pirmos kartos vektorinės žymės B ir C. Antrosios kartos vektorinės žymės D, E ir F, 2.5 paveiksle abiem vaizduojamais atvejais taip pat suformuojamos panašiai. Žymių medžio skirtumai pastebimi tik pradedant trečiąją žymių karta. Trečiosios kartos vektorinės žymės G ir H, abiem atvejais, suformuojama panašiai. Skirtumas pastebimas formuojant vektorines žymes iš žymės D galo taško. 2.5 paveiksle vaizduojamu a atveju, iš žymės D galo, randami du trūkio taškai, kuriuose suformuojamos žymės J ir I. Paveikslo b dalyje, iš žymės D galinio taško randamas vienas trūkio taškas, kuriame suformuojama žymė I. Šiuo atveju, vektorinė žymė J suformuojama iš žymės E galo taško (2.5 pav. b dalis). 2.5 paveikslo b dalyje matoma, kad iš vektorinės žymės J galo taško randami trys trūkio taškai, kuriuose suformuojamos vektorinės žymės K, L ir M. Šiuo atveju, iš žymės M galo taško, randamas vienas trūkio taškas, kuriame suformuojama vektorinė žymė O. 2.5 paveikslo b dalyje, vektorinė žymė N suformuojama tęsiant tolesnį skenavimą iš žymė H galo taško. 2.5 paveikslo a dalyje, iš vektorinės žymės I galo, randami du trūkio taškai, kuriuose suformuojamos vektorinės žymės K ir L. Žymė N suformuojama tęsiant skenavimą iš žymės L galo. 2.5 paveikslo a dalyje viename trūkio taške suformuojamos dvi žymės M ir O dėl to, kad atstumai nuo šių žymių iki roboto pradinės pozicijos taško R skiriasi nežymiai. Žymė O suformuojama tęsiant skenavimą iš žymės N galo taško, o žymė M – iš žymės H galo taško. Abiem 2.5 paveiksle vaizduojamais atvejais suformuojamas skirtingos konfigūracijos vektorinių žymių medis. Skirtingą vektorinių žymių medžio struktūros suformavimą, keičiantis roboto vietai, lemia vektorinių žymių svoriniai koeficientai.

2.4. Vektorinių žymių generavimas

Jei aplinka padalinta į suberdves, svarbu rasti trūkio taškus ir juos susieti su vektorinėmis žymėmis.

Žinant vektorinių žymių savybes bei jų matomumą, galima sudaryti šių žymių formavimo algoritmą.

Pirmiausia suformuojamos svarbiausios algoritmo dalys:

1. Sistemoje yra esamos padėties taškas, iš kurio ir turi būti pradedamas skenavimas.
2. Skenuojant ir radus trūkio taškus, fiksuojamos vektorinės žymės.
3. Tęsiamas skenavimas, formuojant jaunesniųjų kartų vektorines žymes.

4. Skenuojama tol, kol skenuojamoje teritorijoje nelieka nė vienos tame pačiame trūkio taške naujai suformuotos vektorinės žymės, turinčios didesnį svorinį koeficientą.

Aplinkos modelis sudaromas virtualioje aplinkoje su suformuota aplinkos situacija. Aplinkos skenavimas atliekamas imituojamu virtualiu skeneriu.

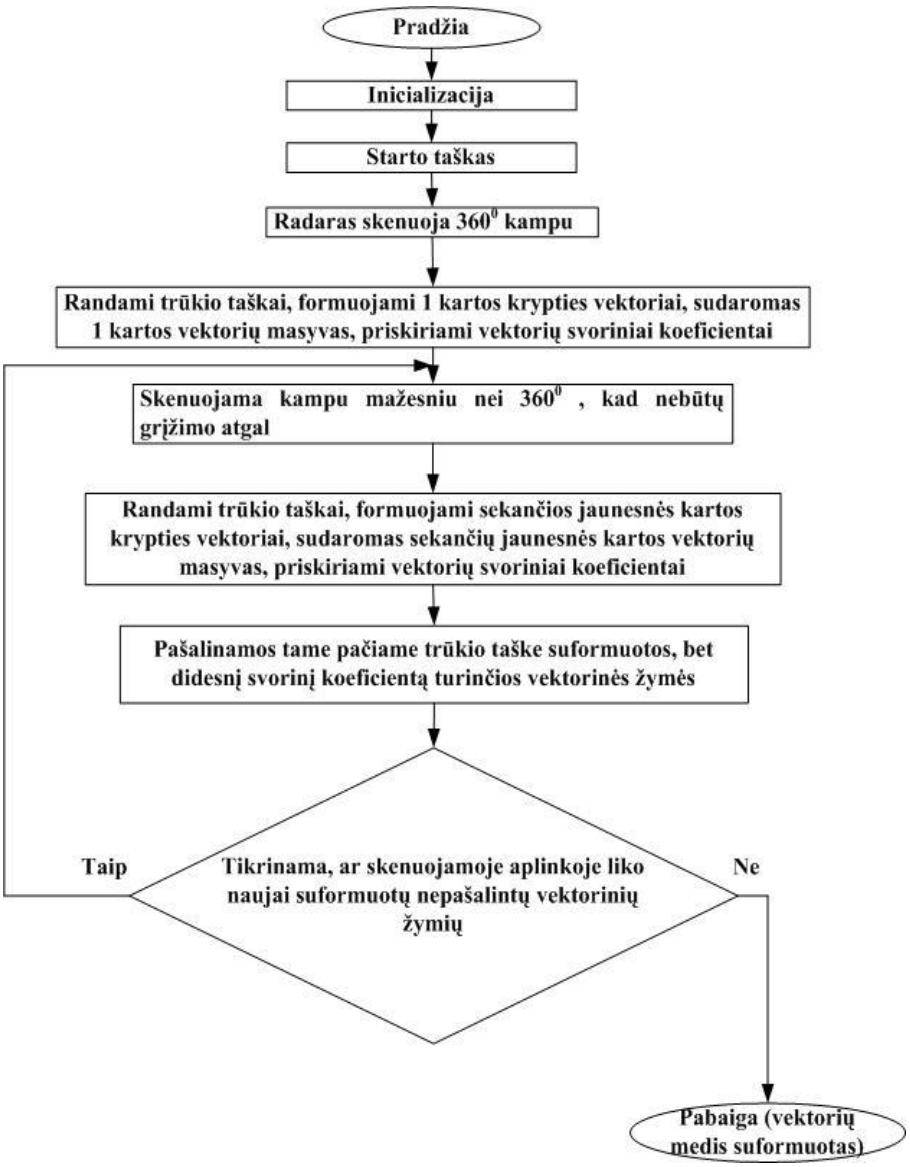
Skenavimas pradedamas iš esamos padėties taško, formuojama medžio struktūra, kurios pavyzdys vaizduojamas 2.4 paveiksle. Pirmiausia turi būti skenuojama visa erdvė aplink robotą, todėl skeneris skenuoja 360^0 kampu, nustatytu skenerio kampo žingsniu. Fiksuojami trūkio taškai. Radus trūkio taškus, formuojami visi pirmos kartos krypties vektoriai. Šie vektoriai yra nukreipti į roboto pradinės pozicijos tašką. Tolesniame žingsnyje sudaromas pirmos kartos vektorių masyvas, šiems vektoriams priskiriami svoriniai koeficientai.

Vykdant tolesnį skenavimą iš pirmos kartos vektorinių žymių galo, skenuojama kampu, kuris yra mažesnis nei 360^0, nustatytu skenerio kampo žingsniu. Skenuojama kampu mažesniu nei 360^0 todėl, kad nebūtų grįžimo atgal. Tai reiškia, kad tęsiant tolesnį skenavimą 360^0 kampu, bus skenuojama aplinka, kuri buvo skenuota prieš tai. Tokiu būdu bus randami tie patys trūkio taškai, kuriuose jau suformuotos vektorinės žymės. Gaunama situacija, kad dubliuojamos vektorinės žymės, kurios, vykdant tolesnius skaičiavimus, bus pašalinamos, dėl turimo didesnio svorinio koeficiento, tačiau tam bus gaištamas laikas.

Randami trūkio taškai ir formuojamos jaunesnės kartos vektorinės žymės. Vektorinėms žymėms priskiriami svoriniai koeficientai. Sudaromi sekančių jaunesnių kartų vektorinių žymių masyvai, kuriuose nurodoma roboto pradinės pozicijos taško koordinatės, vektorinių žymių svoriniai koeficientai ir žymės kartos numeris.

Skenuojant ir skaičiuojant vektorinių žymių svorinius koeficientus skenuojamoje teritorijoje tame pačiame trūkio taške neturi likti naujai suformuotų, didesnį svorinį koeficientą turinčių, vektorinių žymių. Šis reikalavimas susieja vektorinių žymių matomumo 4 sąlygą ir aprašomąjį algoritmą.

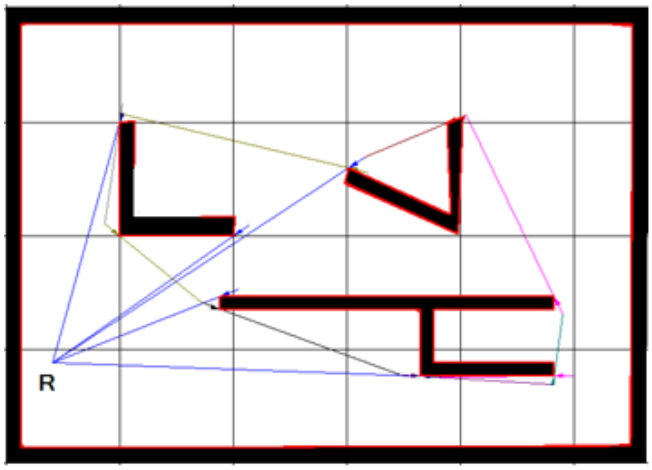
Pagal aprašytą vektorinių žymių formavimo tvarką sudaromas vektorinių žymių skaičiavimo algoritmas, kurio blokinė schema vaizduojama 2.6 paveiksle.



2.6 pav. Vektorinių žymių formavimo algoritmas

Vektorinių žymių formavimo algoritmas realizuojamas programinėje priemonėje, kuri modifikuota taip, kad yra pritaikyta šio vektorinių žymių generavimo algoritmo realizavimui. Teoriškai aprašyti bei išnagrinėti modeliai realizuojami su programine priemone.

Teoriškai, skenuojant aplinką bei aptinkant kliūtis, fiksuojami visi trūkio taškai bei vektorinės žymės. Perteklinės, tame pačiame trūkio taške suformuotos vektorinės žymės, turinčios didesnį svorinį koeficientą, vėliau pašalinamos. Taigi, šiuo atveju nagrinėjant teoriškai, bus vaizduojamos visos rastos vektorinės žymės. 2.7 paveiksle pavaizduota sistema, kurią sudaro trys kliūtys.



2.7 pav. Teorinis vektorinių žymių formavimas.

2.7 paveiksle vaizduojamu atveju robotas žymimas R. Skenuojant iš roboto pradinio taško 360^0 kampu, nustatytu skenerio kampo žingsniu, randami trūkio taškai. Šiuose taškuose formuojamos vyriausios (pirmos) kartos vektorinės žymės. Atliekant tolesnį skenavimą iš suformuotų vektorinių žymių galo taškų, randami kiti trūkio taškai, kuriuose suformuotos kitos vektorinės žymės. Šios vektorinės žymės yra antrosios kartos ir yra nukreiptos ne į roboto pradinės pozicijos tašką, kaip pirmosios kartos vektorinės žymės, bet į atitinkamą pirmosios kartos vektorinę žymę. Jaunesniųjų kartų vektorinės žymės nukreiptos į matomą vyresnės kartos vektorinę žymę. Taip tęsiama tolesnis skenavimas.

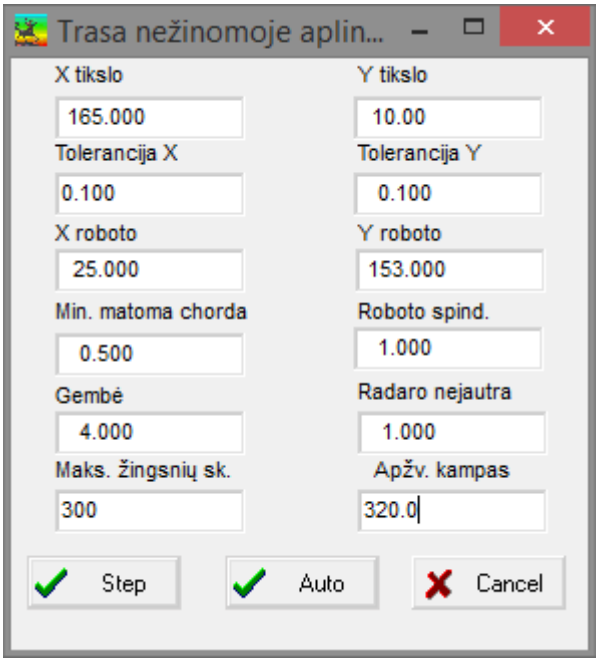
2.5. Programavimo paketas Centaurus CPN

Valdymo sistemų įvairovė ir efektyvaus sprendimo koncepcija bei atsiradusios problemos buvo pagrindinis stimulas, vadinamiesiems klasikiniams spalvotiems Petri tinklams pakisti į kitus, sudėtingesnės struktūros ir didesnių galimybių, Petri tinklus, kurie realizuoti programų pakete Centaurus CPN.

Aprašytasis analitinis trūkio taškų radimo bei vektorinių žymių generavimo pavyzdys yra nesudėtingos aplinkos konfigūracijos. Net ir esant nesudėtingai aplinkos konfigūracijai viename trūkio taške gali būti suformuotos kelios vektorinės žymės. Aprašytas vektorinių žymių generavimo metodas realizuojamas programine priemone. Programinė priemonė „CENTAURUS CPN“, modifikuojama siekiant įgyvendinti modifikuoto vektorinių žymių generavimo algoritmo veikimą spalvotų Petri tinklų bazėje.

Programinės priemonės aplinkoje sudaromas aplinkos žemėlapis. Norint suformuoti vektorinių žymių medį, turi būti nurodomi tam tikri aplinkos skenavimo parametrai. Šie parametrai, jei vektorinių žymių medis sudaromas rankiniu būdu, gali būti keičiami kiekvieną kartą, kai tik formuojamas medis. Formuojant vektorinių žymių medį nurodomi skenavimo

parametrai yra: roboto koordinatės, radaro nejautros dydis, minimali matoma chorda, roboto spindulys, maksimalus žingsnių skaičius, skenerio apžvalgos kampas, gembė. Minėti parametrai įvedami programinės priemonės lange, pateiktame 2.8 paveiksle.



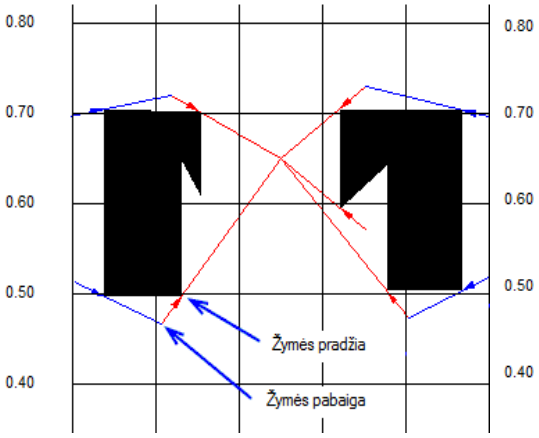
2.8 pav. Vektorinių žymių medžio formavimo papildomas parametrų nustatymo langas programinėje priemonėje

Svarbus faktorius yra vektorinės žymės ilgis (gembė). Šis ilgis įtakoja žymės svorinį koeficientą, kuris skaičiuojamas nuo žymės galo iki roboto pradinės pozicijos taško. Skaičiuojant vektorinės žymės svorinį koeficientą, įskaičiuojamas ir žymės ilgis. Ilginant žymę, gali būti gaunama situacija, kad žymė patenka į statinės kliūties zoną. Tai yra, atstumas tarp dviejų statinių kliūčių yra mažesnis, nei nustatytas vektorinės žymės ilgis. Siekiant išvengti šios situacijos, perskaičiuojamas vektorinės žymės ilgis. Naudojamasi euristiniu sąryšiu tarp vektorinės žymės dydžio l su šuolio ilgiu Δl skenerio kontūre rasto trūkio taške, pasiūlytas algoritmas:

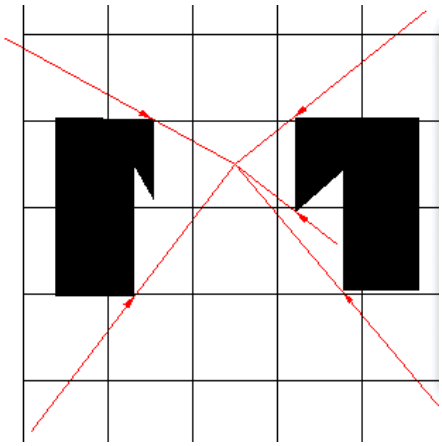
$$l = \begin{cases} 2 \cdot g, \ jei \Delta l \geq 4 \cdot g, \\ 0.25 \cdot \Delta l, \end{cases} \qquad (5)$$

čia g yra skenerio atstumo nejautra.

Pateikti du pavyzdžiai, kaip atrodo vektoriniai medžiai priklausomai nuo vektorinės žymės ilgio, kiek jis išsikiša už nustatytos kliūties, ir kaip tas įtakoja pačiam vektoriniam medžiui. 2.9 pav. pavyzdyje parodyta mėlynomis rodyklėmis, kaip reikia suprasti žymės ilgį. Palyginimui pateiktas 2.10 paveikslas su penkiais kartais ilgesnėmis vektorinėmis žymėmis.



2.9 pav. Vektorinės žymės



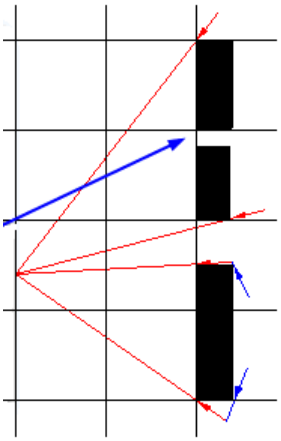
2.10 pav. Ilgesnės vektorinės žymės

Galima atkreipti dėmesį į tai, kad jei vektoriaus išsikišimas kliudytų statinę kliūtį, išsikišimo ilgis automatiškai nustatomas trumpesnis, kad nekliudytų statinės kliūties.

Kitas nurodomas parametras yra kiek vektorinių žymių kartų daugiausia gali būti suformuota arba maksimalus žingsnių skaičius. Jei aplinkos konfigūracija sudėtinga, gali būti atvejų, jog rankiniu būdu nurodyto kartų skaičiaus nepakanka, norint pilnai suformuoti vektorinių žymių medį. Atliekant eksperimentinius tyrimus galima varijuoti žymių kartų kiekiu, norint stebėti kurios žymės sudaromos, nurodant tam tikrą kartų kiekį. Jei įvedamas kartų skaičius yra didesnis nei reikiamas, vektorinių žymių medis baigiamas formuoti, kai nelieka nė vienos naujai suformuotos vektorinės žymės.

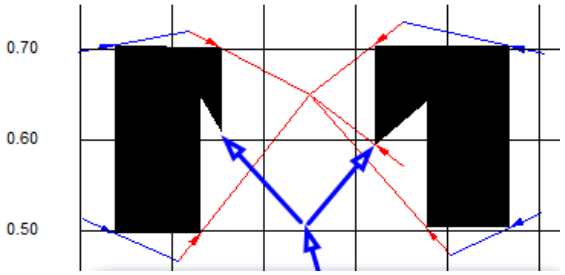
Radaro apžvalgos kampas parenkamas atsižvelgiant į prieš tai aprašytas rekomendacijas. Pasirinkus per mažą apžvalgos kampą, priklausomai nuo vektorinės žymės ilgio, bei statinių kliūčių išsidėstymo aplinkoje, randami ne visi trūkio taškai. Jei šio kampo dydis parenkamas didesnis nei 320^0, atsiranda papildomų skaičiavimų bei vektorinių žymių atmetinėjimo problema. Dėl šių priežasčių ilgėja programos darbo laikas. Suformuojamos vektorinės žymės, kurių svoriniai koeficientai dideli. Vėliau, šios žymės, algoritmo veikimo metu yra pašalinamos kaip perteklinės. Per didelis skenavimo kampas sudaro papildomus skaičiavimus formuojant vektorinių žymių medį.

2.11 paveiksle mėlyna rodykle parodyta, kad nustatyta per didelė minimali matomos chordos reikšmė, neberanda angos tarp statinių kliūčių, kurios dydis yra mažesnis už nustatytą chordos reikšmę.



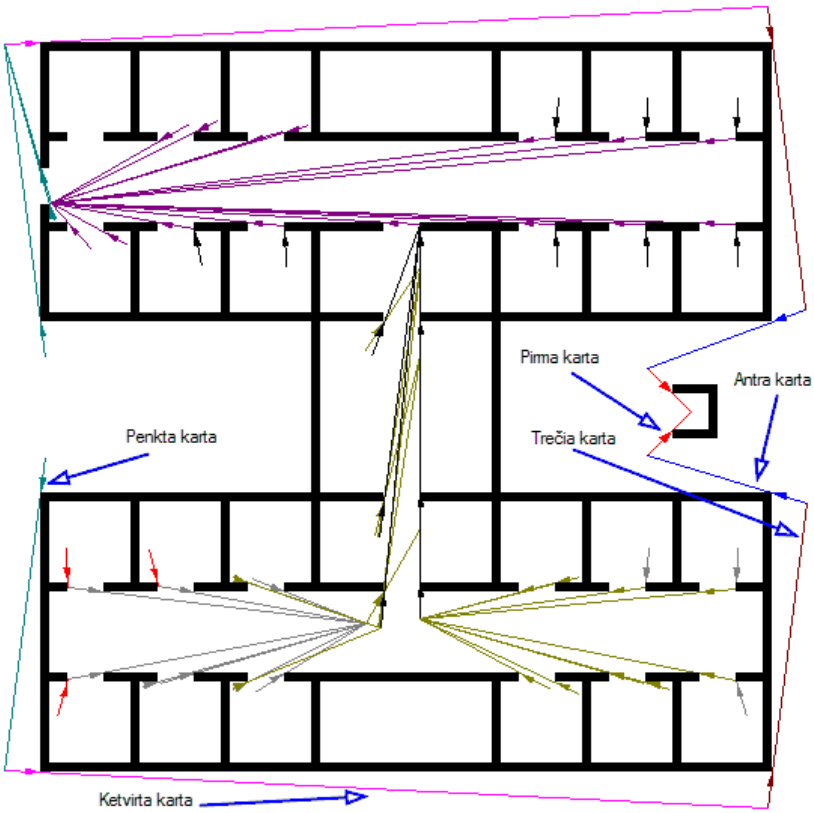
2.11 pav. Minimalios chordos įtaka žemėlapiui

Radaro atstumo nejautra rodo, kokiam gretimų radaro spindulių ilgio šuoliui pradėti fiksuoti trūkio taškus. Pateiktame pavyzdyje 2.12 pav. parodyta, kaip šis parametras įtakoja vektorinio medžio paiešką.



2.12 pav. Atstumo nejautros įtaka žemėlapiui

Nurodžius aplinkos skenavimo parametrus galima generuoti vektorinius medžius. Sugeneruoto vektorinio medžio pavyzdys pavaizduotas paveiksle 2.13.



2.13 pav. Vektorinis medis

Generuojama tiek vektorinių medžių, kiek reikia kad neliktų nematomų teritorijų. Vienas vektorinis medis susideda iš eilės kartų, kur priklausomybė yra ta, kad naujai generuotas vektorius priklauso žemesnei kartai ir t.t.

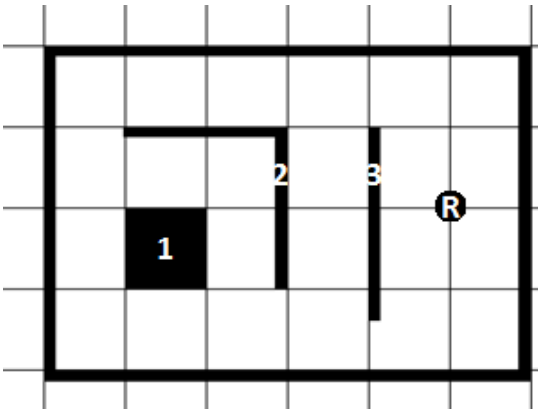
Vektorius yra charakterizuojamas:

- Karta
- Koordinatėmis
- Svoriniu koeficientu - atstumu iki roboto pradinės pozicijos taško einant pagal vektorinio medžio vektorius.

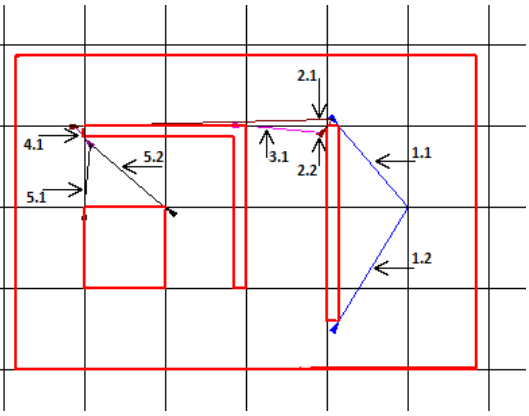
2.6. Modeliavimo rezultatai

Programinio paketo „CENTAURUS CPN“ aplinkoje sudaroma aplinkos, pagal kurias bus tiriami sudarytieji žemėlapiai. Norint išsiaiškinti kokią įtaką žemėlapio išbaigtumui ir tikslumui turi parameterai: radaro nejautros dydis, minimali matoma chorda, maksimalus žingsnių skaičius, skenerio apžvalgos kampas, gembė sukurti keturi skirtingų lygių aplinkos modeliai. Pirmasis pateiktas 2.14 paveikslėlyje yra nesudėtingos konfiguracijos, antrasis pateiktas 2.21 pav. ir trečiasis pateiktas 2.27 pav vidutinio sunkumo, ketvirtasis pateiktas 2.35 paveiksle sunkios konfigūracijos. Sudaryti aplinkų žemėlapiai yra lyginami ir tiriama kiekvieno parametros įtaka žemėlapio tisklumui bei išbaigtumui.

Programinio paketo „CENTAURUS CPN“ aplinkoje sudaroma aplinka pavazduota 2.14 pav. ir pažymima statinės kliūtys numeriais „1“, „2“, „3“. Roboto pradinė pozicija aplinkoje pažymėta raide R.



2.14 pav. Sudaryta aplinka



2.15 pav. Aplinkos žemėlapis

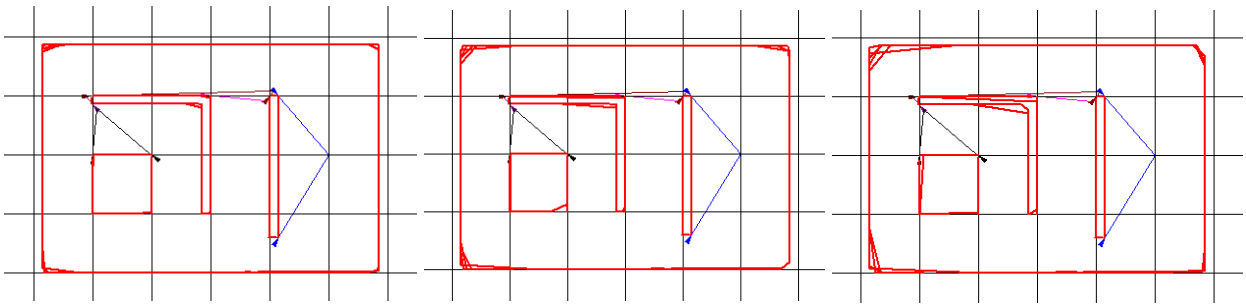
Minimali matoma chorda, gembė, radaro nejautra, maksimalus žingsnių skaičius ir apžvalgos kampas, tai parametrai, kurie gali įtakoti vektorinių žymių medžio struktūrą bei žemėlapio tikslumą. Paveiksle 2.15 parodyta kaip atrodo sudarytas aplinkos žemėlapis pagal 2.1 lentelėje pateiktus duomenis.

2.1 lentelė. Aplinkos žemėlapio sudarymo duomenys

Minimali matoma chorda	Gembė	Radaro nejautra	Maksimalus žingsnių skaičius	Apžvalgos kampas
1	2	1	100	330^{o}

Sudaromas vektorinių žymių medis atsižvelgiant į skenavimo metu aptiktas statines kliūtis. Sudarytame aplinkos žemėlapyje pažymimos vektorinės šakos, pirmasis skaičius žymi kartos eilę, antrasis šakos skaičių kartoje. Paveikslėliuose 2.16 – 2.20 parodyta kaip kiekvienas iš parametrų keičia sudaromą žemėlapį. Parametrai keičiami po vieną, kelis kartus. Statinių kliūčių žymėjimas ir roboto pradinė pozicija nesikeičia.

Programinio paketo „CENTAURUS CPN“ aplinkoje atliekami modeliavimo eksperimentai keičiant apžvalgos matymo kampo žingsnio (minimalios matomos chordos) reikšmę. Aplinkos žemėlapyje, lyginant su 2.15 paveikslėlyje gautu aplinkos žemėlapiu atsiranda netikslumų. Gauti eksperimentiniai rezultatai vaizduojami 2.16 pav.



a) b) c)

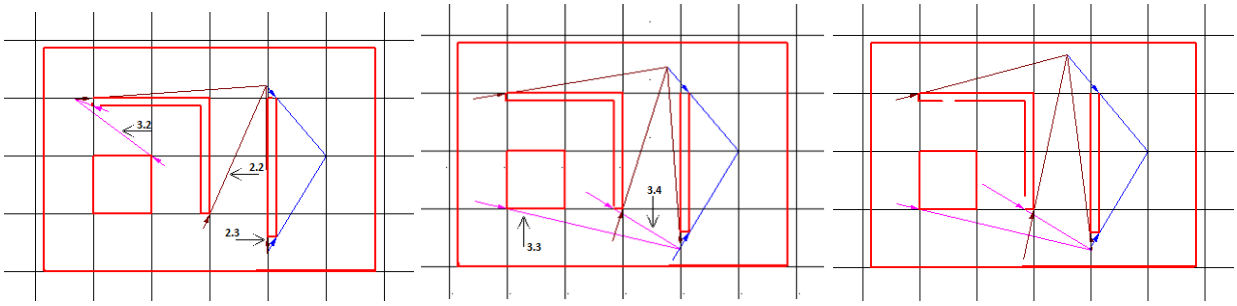
2.16 pav. Eksperimentiniai rezultatai keičiant minimalios matomos chordos reikšmę

2.16 paveikslo a dalyje chordos ilgis – 10, b dalyje – 20, c dalyje – 30.

Kaip matoma iš paveikslo 2.16 a dalies pakeitus reikšmę į 10 nedideli netikslumai matomi kontūro ir 2 statinės kliūties kampuose. Didinant apžvalgos matymo kampo žingsnį nuo 10 iki 20 minėti netikslumai buvę a dalyje išryškėja b dalyje, taip pat atsiranda naujų 1 statinėje kliūtyje dešiniajame apatiniame kampe. Nustačius minimalios matomos chordos reikšmę – 30 kontūro kairiuosiuose kampuose bei 2 statinėje kliūtyje nepilnai suformuojamas kliūčių kontūrai, taip pat nepilnai suformuojama 1 statinės kliūties kairioji kraštinė. Vektorinių šakų išsidėstymas bei jų kiekis nepakito.

Iš paveiksle 2.16 pateiktų eksperimentinių rezultatų nustatyta, kad per didelė minimalios matomos chordos reikšmė ženkliai įtakoja modeliavimo rezultatus, t. y. neberandamos tikslios vietos kur yra statinių kliūčių kampai, dėl per didelio laipsnių šuolio užfiksuojamos klaidingos vektorinių žymių vietos ir sudaromas klaidingas aplinkos žemėlapis. Palyginus 2.15 paveikslėlį su 2.16 pav. a dalimi žemėlapis skiriasi 6 %, su b dalimi 23 %, su c dalimi 34 %.

Programinio paketo „CENTAURUS CPN“ aplinkoje parametras vektorinės žymės ilgis (gembė) yra svarbus faktorius, kuris įtakoja vektorinio medžio išsidėstymą. Gauti eksperimentiniai rezultatai vaizduojami 2.17 pav.



a) b) c)

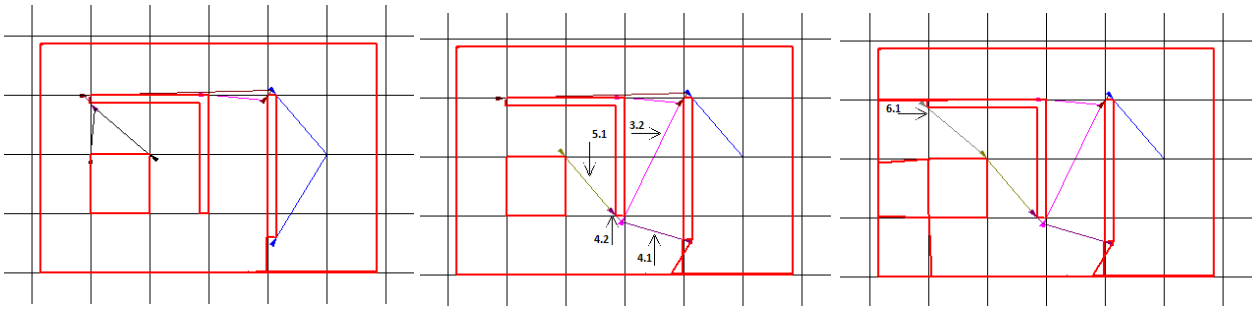
2.17 pav. Eksperimentiniai rezultatai keičiant vektorinės žymės ilgį

2.17 paveikslo a dalyje gembė – 5, b dalyje – 10, c dalyje – 15.

Kaip matoma iš paveikslo 2.17 a dalies pakeitus reikšmę iš 2 į 5 pasikeičia vektorinis medis. 2.17 pav. a dalyje numeriais 2.2, 2.3, 3.2 pažymėta naujai atsiradusios vektorinio medžio atšakos. Lyginant su 2.15 pav. sudarytu žemėlapiu išnyko šakos 3.1, 5.1, 5.2. Padidinus gembės reikšmę iki 10 išnyksta 3.2 šaka atsiradusi 2.17 pav. c dalyje ir 4.1 atsiradusi 2.15 pav., trečios kartos šakos pažymėtos 3.3 ir 3.4 susiformuoja kitoje vietoje, 3 statinės kliūties apačioje. Nustačius vektorinės žymės ilgį – 15 vektorinės šakų skaičius ir išsidėstymas nepakinta lyginant su b dalimi, tačiau nepilnai sudaromas 2 statinės kliūties kontūras.

Iš paveiksle 2.17 pateiktų eksperimentinių rezultatų nustatyta, kad vektorinės žymės ilgis, kiek ji išsikiša už nustatytos kliūties, įtakoja vektorinio medžio struktūrą ir modeliavimo rezultatus. Palyginus 2.15 paveikslėlį su 2.17 pav. a dalimi žemėlapis skiriasi 1 %, su b dalimi 1 %, su c dalimi 2 %.

Programinio paketo „CENTAURUS CPN“ aplinkoje atliekami modeliavimo eksperimentai keičiant radaro nejautrą. Aplinkos žemėlapyje, lyginant su 2.15 paveikslėlyje gautu aplinkos žemėlapiu atsiranda menamų statinių kliūčių. Eksperimentiniai rezultatai pateikti paveiksle 2.18.



a) b) c)

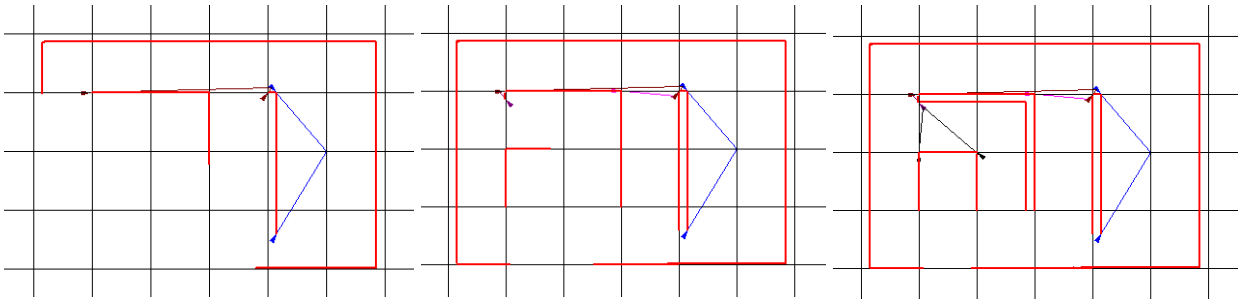
2.18 pav. Eksperimentiniai rezultatai keičiant radaro nejautrą

4.5 paveikslo a dalyje radaro nejautra – 11, b dalyje – 14, c dalyje – 17

Kaip matoma iš paveikslo 2.18 a dalies padidinus radaro nejautrą iki 11, 3 statinės kliūties apačioje gaunama menama statinė kliūtis, kurios sukurtoje 2.15 pav. aplinkoje nėra. Pakeitus reikšmę į 14 atsiranda dar viena menama statinė kliūtis, 3 statinės kliūties apačioje, taip pat pasikeičia vektorinio medžio išsidėstymas, išnyksta šakos 1.2, 4.1, 5.1, 5.2 pavaizduotos 2.15 pav. ir atsiranda naujų šakų 3.2, 4.1, 4.2, 5.1, jos pažymėtos 2.18 pav. b dalyje. Nustačius reikšmę – 17 susiformuoja menamos statinės kliūtys 1 statinės kliūties karėje ir apačioje, 2 statinės kliūties karėje ir išlieka 3 statinės kliūties apačioje. Atsiranda nauja vektorinė šaka 6.1 lyginant su b dalimi 2.18 pav.

Iš paveiksle 2.18 pateiktų eksperimentinių rezultatų nustatyta, kad gretimų radaro spindulių ilgio šuolis yra per mažas, kad būtų užfiksuoti reikalingi trūkio taškai. Pasikeičia vektorinis medis, žemėlapis gaunamas su menamomis statinėmis kliūtimis. Palyginus 2.15 paveikslėlį su 2.18 pav. a dalimi žemėlapis skiriasi 3 %, su b dalimi 6 %, su c dalimi 20 %.

Programinio paketo „CENTAURUS CPN“ aplinkoje keičiamas parametras yra kiek vektorinių žymių kartų daugiausia gali būti suformuota (maksimalus žingsnių skaičius), eksperimento rezultatai pateikti paveiksle 2.19.



a) b) c)

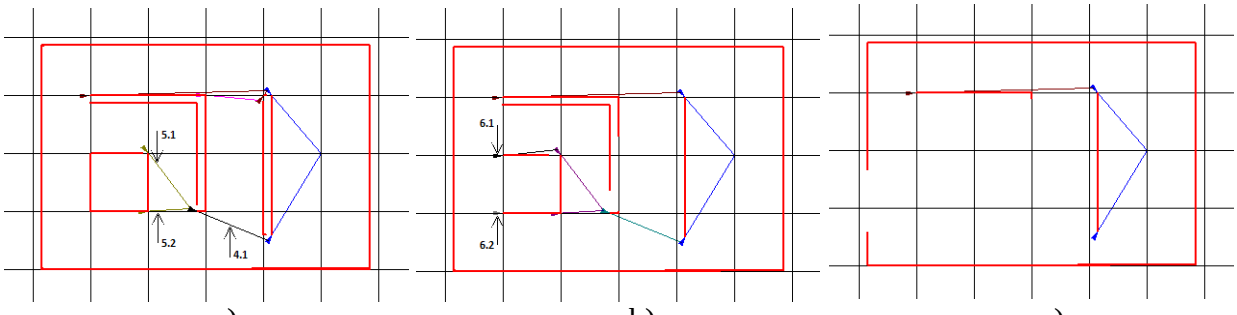
2.19 pav. Eksperimentiniai rezultatai keičiant maksimalų žingsnių skaičių

2.19 paveikslo a dalyje maksimalus žingsnių skaičius – 2, b dalyje – 5, c dalyje – 7

Kaip matoma iš paveikslo 2.19 a dalies nustačius 2 žingsnius žemėlapio kontūras, 2, 3 statinės kliūtys nėra pilnai išbaigtos, 1 statinė kliūtis neužfiksuota. Padidinus reikšmę iki 5 žemėlapio kontūras, 1, 2 statinės kliūtys yra neišbaigtos. Pakeitus žingsnių skaičių į 7 nepilnai sudaromas žemėlapio kontūras ir 1 statinės kliūties apatinė kraštinė.

Iš paveiksle 2.19 pateiktų eksperimentinių rezultatų nustatyta, kad nurodytų kartų skaičiaus nepakanka, norint pilnai suformuoti vektorinių žymių medį ir sudaryti išsamų žemėlapį. Palyginus 2.15 paveikslėlį su 2.19 pav. a dalimi žemėlapis skiriasi 38 %, su b dalimi 15 %, su c dalimi 8 %.

Programinio paketo „CENTAURUS CPN“ aplinkoje atliekami modeliavimo eksperimentai keičiant radaro apžvalgos matymo kampą. Lyginant su 2.15 paveikslėlyje gautu aplinkos žemėlapiu suformuojamas neišbaigtas aplinkos žemėlapis. Eksperimentiniai rezultatai pateikti paveiksle 2.20.



a) b) c)

2.20 pav. Eksperimentiniai rezultatai keičiant radaro apžvalgos kampą

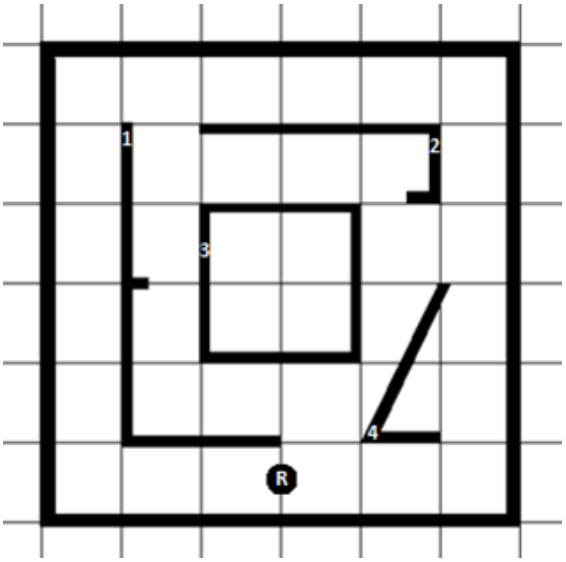
2.20 paveikslo a dalyje radaro apžvalgos kampas – 250º, b dalyje – 170º, c dalyje – 120º

Kaip matoma iš paveikslo 2.20 a dalies sumažinus radaro apžvalgos kampą iki 250º nepilnai sudaromas 2 statinės kliūties kontūras, taip pat pasikeičia vektorinis medis., t. y. išnyksta šakos

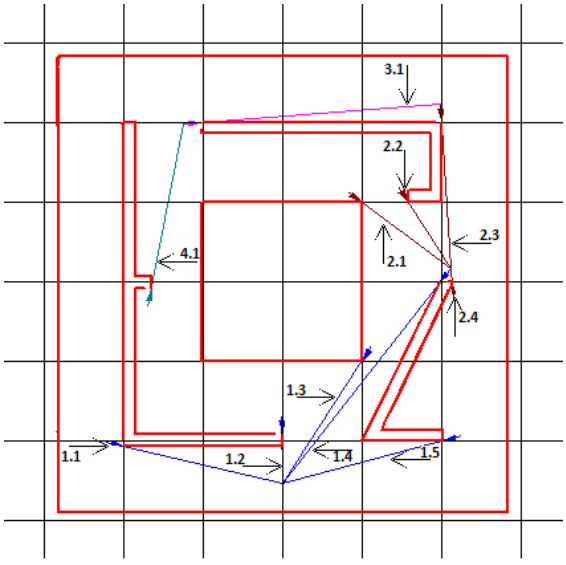
4.1, 5.1, 5.2 vaizduotos 2.15 pav., bei atsiranda naujos šakos 4.1, 5.1, 5.2 nurodytos 2.20 a dalyje. Pakeitus reikšmę į 170 ^{o} nepilnai užfiksuojami 1, 2 ,3 statinių kliūčių kontūrai, palyginus su 2.20 pav. a dalimi atsiranda 2 naujos šakos 6.1, 6.2. Sumažinus parametrą iki 120^{o} žemėlapio kontūras, 2, 3 statinės kliūtys nepilnai užfiksuojamos, 1 statinė kliūtis neužfiksuojama. Lyginant su 2.15 pav. išlieka tik 1.1, 1.2, 2.1 šakos.

Iš paveiksle 2.20 pateiktų eksperimentinių rezultatų nustatyta, kad pasirinkus per mažą apžvalgos kampą randami ne visi trūkio taškai ir sudaromas klaidingas aplinkos žemėlapis. Palyginus 2.15 paveikslėlį su 2.20 pav. a dalimi žemėlapis skiriasi 1 %, su b dalimi 20 %, su c dalimi 38 %.

Programinio paketo „CENTAURUS CPN“ aplinkoje sudaroma aplinka pavaizuota 2.21 pav ir pažymima statinės kliūtys numeriais „1“, „2“, „3“, „4“. Roboto pradinė pozicija aplinkoje pažymėta raide R.



2.21 pav. Sudaryta aplinka



2.22 pav. Aplinkos žemėlapis

2.22 paveiksle parodyta kaip atrodo sudarytas aplinkos žemėlapis pagal lentelėje 2.2 pateiktus duomenis.

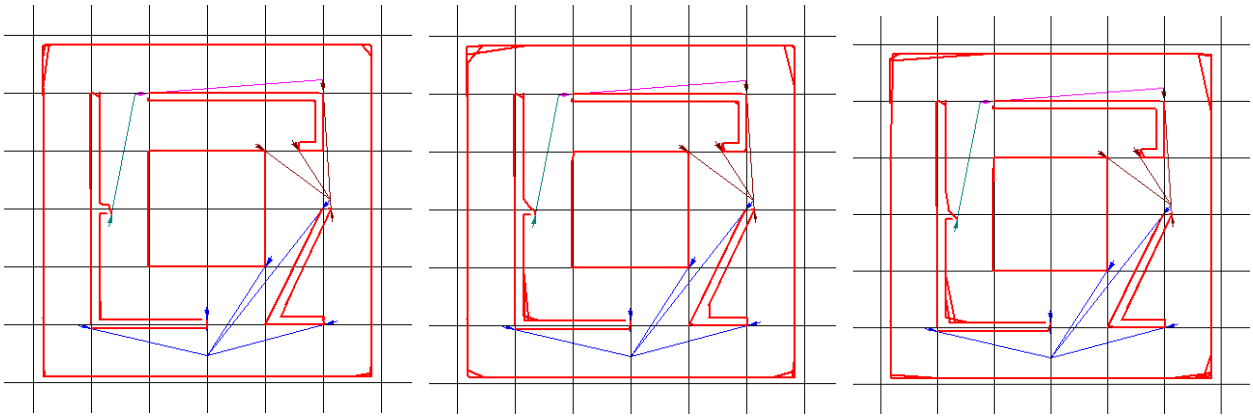
2.2 lentelė. Aplinkos žemėlapio sudarymo duomenys

Minimali matoma chorda	Gembė	Radaro nejautra	Maksimalus žingsnių skaičius	Apžvalgos kampas
1	4	1	100	300^{o}

Sudaromas vektorinių žymių medis atsižvelgiant į skenavimo metu aptiktas statines kliūtis. Sudarytame aplinkos žemėlapyje pažymimos vektorinės šakos, pirmasis skaičius žymi kartos eilę, antrasis šakos skaičių kartoje. Paveiksluose 2.23 – 2.27 parodyta kaip kiekvienas iš parametrų

keičia sudaromą žemėlapį. Parametrai keičiami po vieną, kelis kartus. Statinių žymių numeravimas ir roboto pradinė pozicija nesikeičia.

Programinio paketo „CENTAURUS CPN“ aplinkoje atliekami modeliavimo eksperimentai keičiant apžvalgos matymo kampo žingsnį (minimalią matomą chordą). Aplinkos žemėlapyje, lyginant su 2.22 paveikslėlyje gautu aplinkos žemėlapiu atsiranda netikslumų. Gauti eksperimentiniai rezultatai vaizduojami 2.23 pav.



a) b) c)

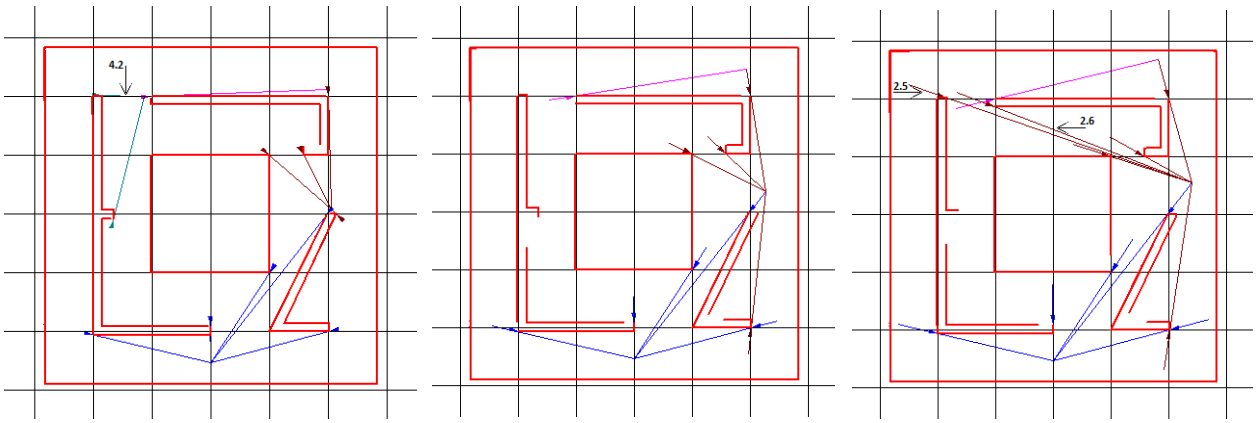
2.23 pav. Eksperimentiniai rezultatai keičiant apžvalgos matymo kampo žingsnio reikšmę.

2.23 paveikslo a dalyje chordos ilgis – 9, b dalyje – 14, c dalyje – 19.

Kaip matoma iš 2.23 paveikslo a dalies pakeitus apžvalgos matymo kampo žingsnio reikšmę iš 1 į 9 nedideli netikslumai matomi žemėlapio kontūro viršutiniuose kairiajame ir dešiniajame kampuose. Didinant apžvalgos matymo kampo žingsnį iki 14 netikslumai atsiranda visuose žemėlapio kontūro kampuose bei 1 ir 4 statinėse kliūtyse. Nustačius žingsnio reikšmę – 19 išlieka netikslumai žemėlapio kontūro kampuose ir 1 statinėje kliūtyje.

Iš paveiksle 2.23 pateiktų eksperimentinių rezultatų nustatyta, kad per didelė minimalios matomos chordos reikšmė ženkliai įtakoja modeliavimo rezultatus, t. y. neberandamos tikslios vietos kur yra statinių kliūčių kampai, dėl per didelio laipsnių šuolio užfiksuojamos netikslios vektorinių žymių vietos ir sudaromas klaidingas aplinkos žemėlapis. Palyginus 2.22 paveikslėlį su 2.23 pav. a dalimi žemėlapis skiriasi 10 %, su b dalimi 20 %, su c dalimi 30 %.

Programinio paketo „CENTAURUS CPN“ aplinkoje eksperimentiškai modeliuojant parametras vektorinės žymės ilgis (gembė) yra svarbus faktorius, kuris įtakoja vektorinio medžio išsidėstymą. Gauti eksperimentiniai rezultatai vaizduojami 2.24 pav.



a) b) c)

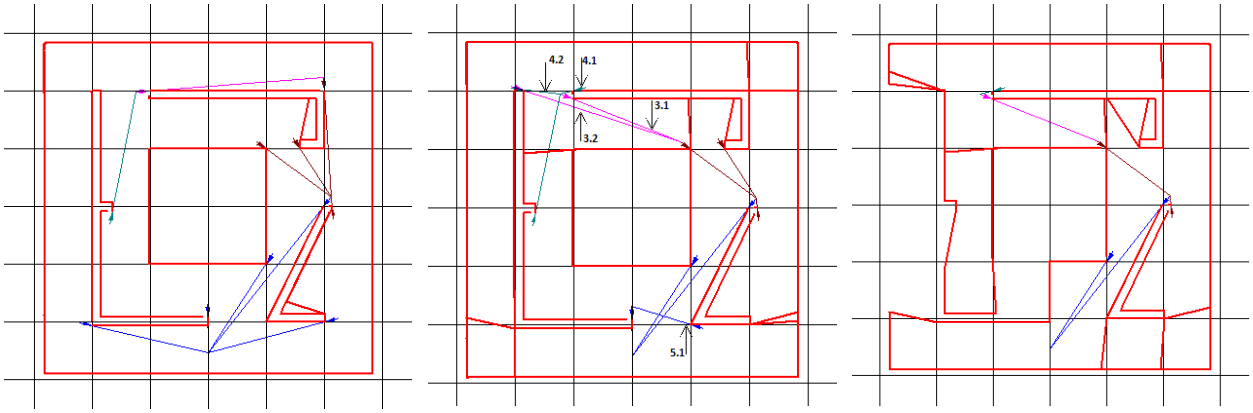
2.24 pav. Eksperimentiniai rezultatai keičiant gembės reikšmę.

2.24 paveikslo a dalyje vektorinės žymės ilgis ilgis – 2, b dalyje – 8, c dalyje – 12.

Kaip matoma iš 2.24 paveikslo a dalies pakeitus gembės reikšmę į 2 pasikeičia vektorinis medis, 4.2 pažymėta naujai atsiradusi vektorinio medžio atšaka, taip pat pilnai nesudaromas 2 statinės kliūties kontūras. Padidinus gembės reikšmę iki 8 lyginant su 2.22 aplinkos žemėlapiu išnyksta šaka 4.1, 1 ir 4 statinių kliūčių kontūrai yra nepilnai sudaryti. Nustačius vektorinės žymės ilgį – 12 atsiranda dvi naujos vektorinio medžio šakos 2.5 ir 2.6, taip pat padidėja 1 ir 4 statinių kliūčių neužfiksuoto kontūro plotas.

Iš paveiksle 2.24 pateiktų eksperimentinių rezultatų nustatyta, kad vektorinės žymės ilgis, kiek ji išsikiša už nustatytos kliūties, įtakoja vektorinio medžio struktūrą ir modeliavimo rezultatus. Palyginus 2.22 paveikslėlį su 2.24 pav. a dalimi žemėlapis skiriasi 2 %, su b dalimi 5 %, su c dalimi 6 %.

Programinio paketo „CENTAURUS CPN“ aplinkoje atliekami modeliavimo eksperimentai keičiant radaro nejautrą. Aplinkos žemėlapyje, lyginant su 2.22 paveikslėlyje gautu aplinkos žemėlapiu atsiranda menamų statinių kliūčių. Eksperimentiniai rezultatai pateikti paveiksle 2.25.



a) b) c)

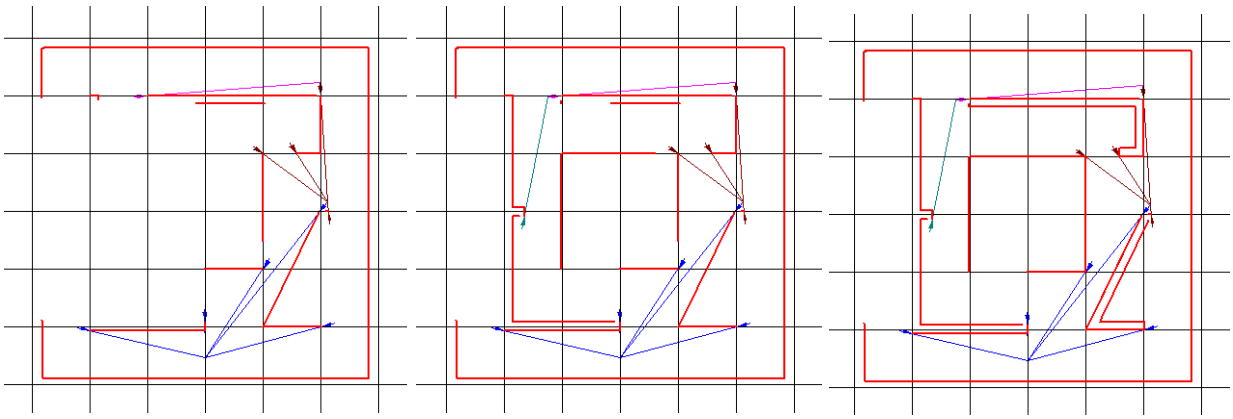
2.25 pav. Eksperimentiniai rezultatai keičiant radaro nejautrą.

2.25 paveikslo a dalyje radaro nejautra – 12, b dalyje – 15, c dalyje – 18.

Kaip matoma iš 2.25 paveikslo a dalies padidinus radaro nejautrą iki 12, statinėse kliūtyse 2 ir 4 suformuojama menamos statinės kliūtis, kurios sukurtoje aplinkoje nėra. Pakeitus reikšmę į 15 atsiranda daugiau menamų statinių kliūčių: prie 1 statinės kliūtis, prie 2 statinės kliūties dešinio krašto, prie 3 statinės kliūties viršutinių dešinio ir kairiojo kampo, prie 4 statinės kliūties apatinio dešinio kampo, taip pat pasikeičia vektorinio medžio išsidėstymas , palyginus su 2.22 pav. vaizduojamu aplinkos žemėlapiu išnyksta šakos 1.1, 1.5, 2.3, 3.1, taip pat atsiranda naujų šakų 3.1, 3.2, 4.1, 4.2, 5.1. Nustačius reikšmę – 18 susiformuoja menamos statinės kliūtys 1 statinės kliūties karėje ir apačioje, 2 statinės kliūties dešinėje, 3 statinės kliūties viršutiniuose dešiniajame ir kairiajame kampuose, 4 statinės kliūties apačioje. Išnyksta vektorinės šakos 1.1, 1.2, 1.5, 2.2, 2.3, 3.1, 4.1 pav. lyginant su 2.22 pav. užfiksuotomis vektorinėmis šakomis.

Iš paveiksle 2.25 pateiktų eksperimentinių rezultatų nustatyta, kad gretimų radaro spindulių ilgio šuolis yra per mažas, kad būtų užfiksuoti reikalingi trūkio taškai. Pasikeičia vektorinis medis, žemėlapis gaunamas su menamomis statinėmis kliūtimis. Palyginus 2.22 paveikslėlį su 2.25 pav. a dalimi žemėlapis skiriasi 4 %, su b dalimi 16 %, su c dalimi 45 %.

Programinio paketo „CENTAURUS CPN“ aplinkoje keičiamas parametras yra kiek vektorinių žymių kartų daugiausia gali būti suformuota (maksimalus žingsnių skaičius), eksperimento rezultatai pateikti paveiksle 2.26.



a) b) c)

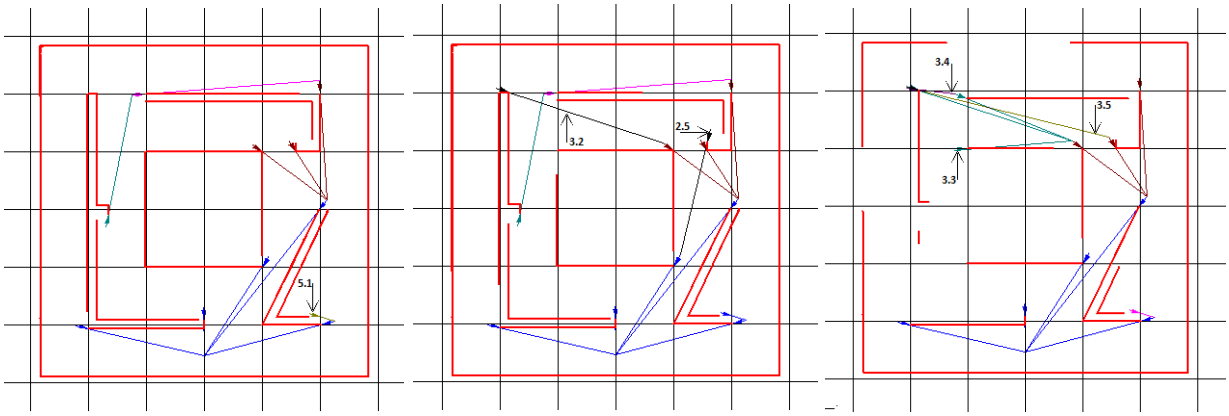
2.26 pav. Eksperimentiniai rezultatai keičiant maksimalų žingsnių skaičių.

2.26 paveikslo a dalyje maksimalus žingsnių skaičius – 3, b dalyje – 5, c dalyje – 8.

Kaip matoma iš 2.26 paveikslo a dalies nustačius 3 žingsnius žemėlapio kontūras, 1, 2, 3, 4 statinės kliūtys nėra pilnai išbaigtos. Padidinus reikšmę iki 5 žemėlapio kontūras, 1, 2, 3, 4 statinės kliūtys yra neišbaigtos. Pakeitus žingsnių skaičių į 8 nepilnai sudaromas žemėlapio kontūras, 1, 3 statinės kliūtys.

Iš 2.26 paveiksle pateiktų eksperimentinių rezultatų nustatyta, kad nurodytų kartų skaičiaus nepakanka, norint pilnai suformuoti vektorinių žymių medį ir sudaryti išsamų žemėlapį. Palyginus 2.22 paveikslėlį su 2.26 pav. a dalimi žemėlapis skiriasi 50 %, su b dalimi 30 %, su c dalimi 18 %.

Programinio paketo „CENTAURUS CPN“ aplinkoje atliekami modeliavimo eksperimentai keičiant apžvalgos matymo kampą. Lyginant su 2.22 paveikslėlyje gautu aplinkos žemėlapiu suformuojamas neišbaigtas aplinkos žemėlapis. Eksperimentiniai rezultatai pateikti 2.27 paveiksle.



a) b) c)

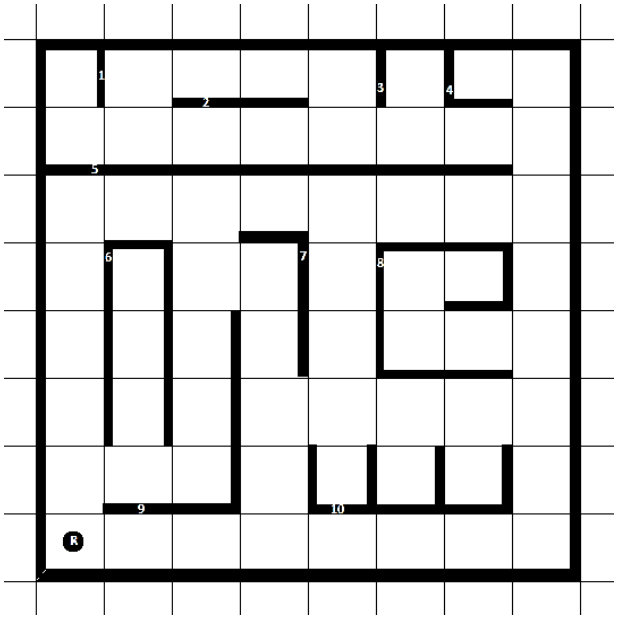
2.27 pav. Eksperimentiniai rezultatai keičiant radaro apžvalgos kampą.

2.27 paveikslo a dalyje apžvalgos kampas – 260º, b dalyje – 190 º, c dalyje – 120º.

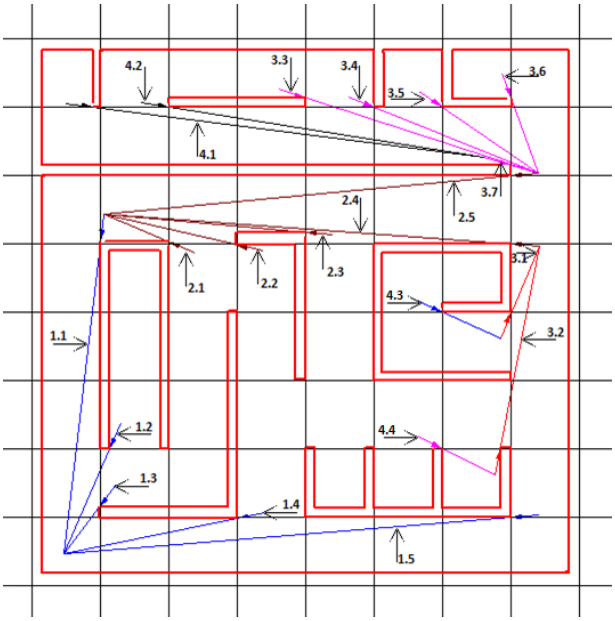
Kaip matoma iš 2.27 paveikslo a dalies sumažinus radaro apžvalgos kampą iki 260º nepilnai sudaromas 1, 2, 4 statinės kliūties kontūras, taip pat pasikeičia vektorinis medis, atsiranda nauja šaka 5.1. Pakeitus reikšmę į 190 º nepilnai užfiksuojami 1, 2 ,3, 4 statinių kliūčių kontūrai, atsiranda 2 naujos šakos 2.5, 3.2. Sumažinus parametrą iki 120º žemėlapio kontūras, 1, 2, 3, 4 statinės kliūtys nepilnai užfiksuojamos. Lyginant su 2.22 pav. vektorinio medžio šakomis išnyksta 2.4, 3.1, 4.1 ir atsiranda naujų šakų 3.3, 3.4, 3.5.

Iš 2.27 paveiksle pateiktų eksperimentinių rezultatų nustatyta, kad pasirinkus per mažą apžvalgos kampą randami ne visi trūkio taškai ir sudaromas klaidingas aplinkos žemėlapis. Palyginus 2.22 paveikslėlį su 2.27 pav. a dalimi žemėlapis skiriasi 3 %, su b dalimi 12 %, su c dalimi 38 %.

Programinio paketo „CENTAURUS CPN“ aplinkoje sudaroma aplinka pateikta 2.28 pav. ir pažymima statinės kliūtys numeriais „1“, „2“, „3“, „4“, „5“, „6“, „7“, „8“, „9“, „10“. Roboto pradinė pozicija aplinkoje pažymėta raide R.



2.28 pav. Sudaryta aplinka



2.29 pav. Aplinkos žemėlapis

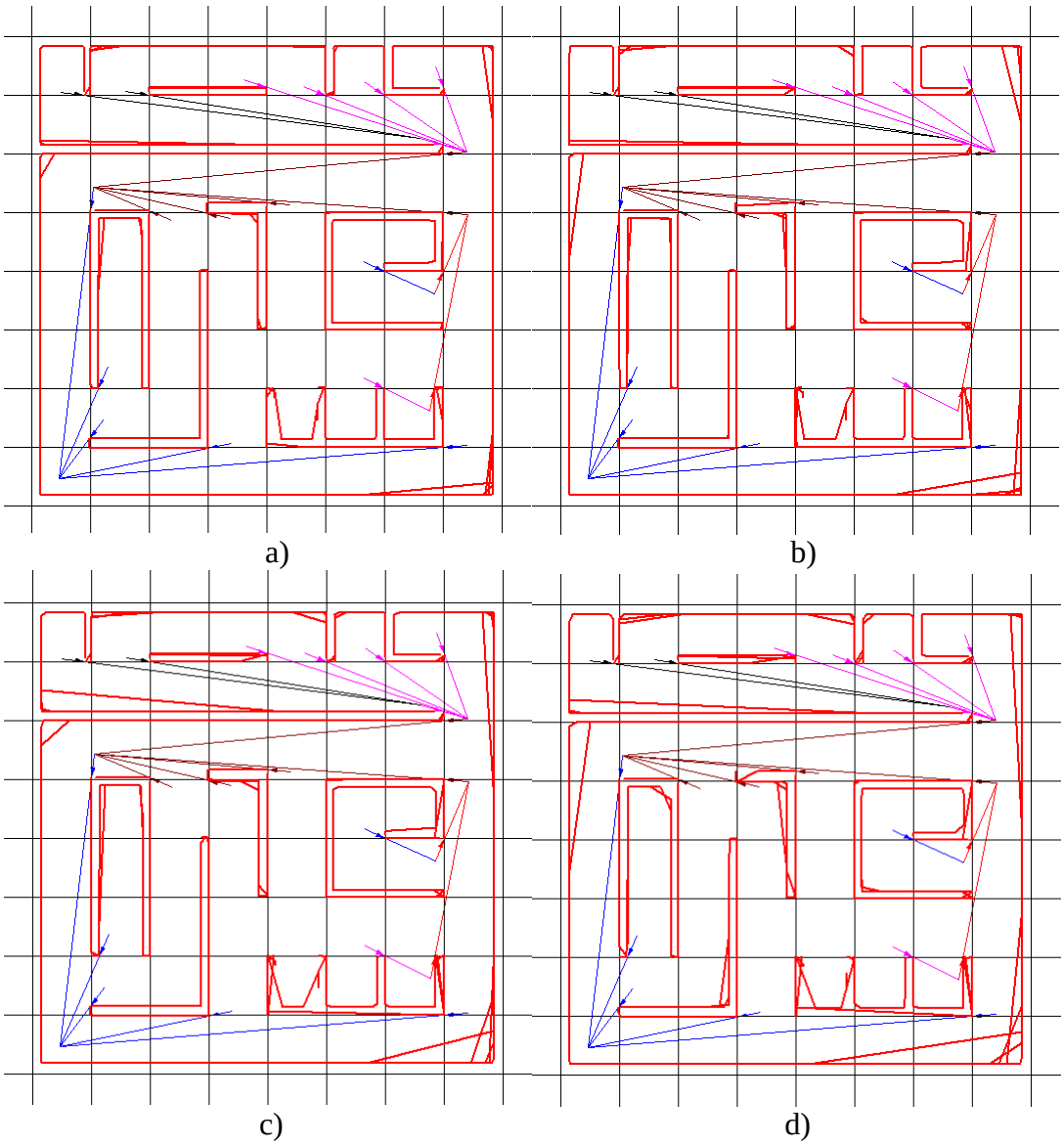
2.29 paveiksle parodyta kaip atrodo sudarytas aplinkos žemėlapis pagal lentelėje 2.3 pateiktus duomenis.

2.3 lentelė. Aplinkos žemėlapio sudarymo duomenys

Minimali matoma chorda	Gembė	Radaro nejautra	Maksimalus žingsnių skaičius	Apžvalgos kampas
1	7	1	100	340^{o}

Sudaromas vektorinių žymių medis atsižvelgiant į skenavimo metu aptiktas statines kliūtis. Sudarytame aplinkos žemėlapyje pažymimos vektorinės šakos, pirmasis skaičius žymi kartos eilę, antrasis šakos skaičių kartoje. Paveikslėliuose 2.30 – 2.34 parodyta kaip kiekvienas iš parametrų keičia sudaromą žemėlapį. Parametrai keičiami po vieną, kelis kartus. Statinių žymių numeravimas ir roboto pradinė pozicija nesikeičia.

Programinio paketo „CENTAURUS CPN“ aplinkoje atliekami modeliavimo eksperimentai keičiant apžvalgos matymo kampo žingsnį (minimalią matomą chordą). Aplinkos žemėlapyje, lyginant su 2.29 paveikslėlyje gautu aplinkos žemėlapiu atsiranda netikslumų. Gauti eksperimentiniai rezultatai vaizduojami 2.30 pav.



2.30 pav. Eksperimentiniai rezultatai keičiant apžvalgos matymo kampo žingsnio reikšmę.

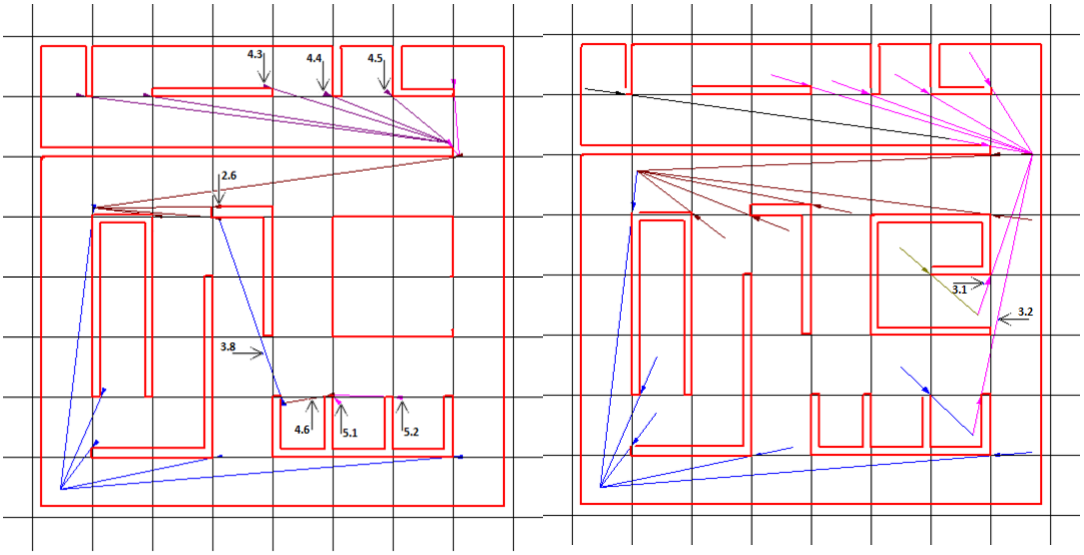
2.30 paveikslo a dalyje chordos ilgis – 10, b dalyje – 15, c dalyje – 20. D dalyje - 25.

Kaip matoma iš 2.30 paveikslo a dalies keičiant apžvalgos matymo kampo žingsnio reikšmę į 10 nedideli netikslumai matomi žemėlapio kontūro dešiniuose kampuose, 1 statinės kliūties dešiniajame kampe, 5 statinės kliūties dešiniuose kampuose, 6, 7, 10 statinėse kliūtyse. Didinant apžvalgos matymo kampo žingsnį nuo 10 iki 15 minėti netikslumai 2.30 pav. a dalyje. išryškėja, taip pat atsiranda naujų, 8 statinėje kliūtyje dešinėje. Nustačius minimalios matomos chordos reikšmę – 20 netikslumai dar labiau išryškėja lyginant su 2.30 pav. c dalimi. Pakeitus apžvalgos matymo kampo žingsnio reikšmę į 25 žemėlapio kontūre ir visose statinėse kliūtyse matomi netikslumai.

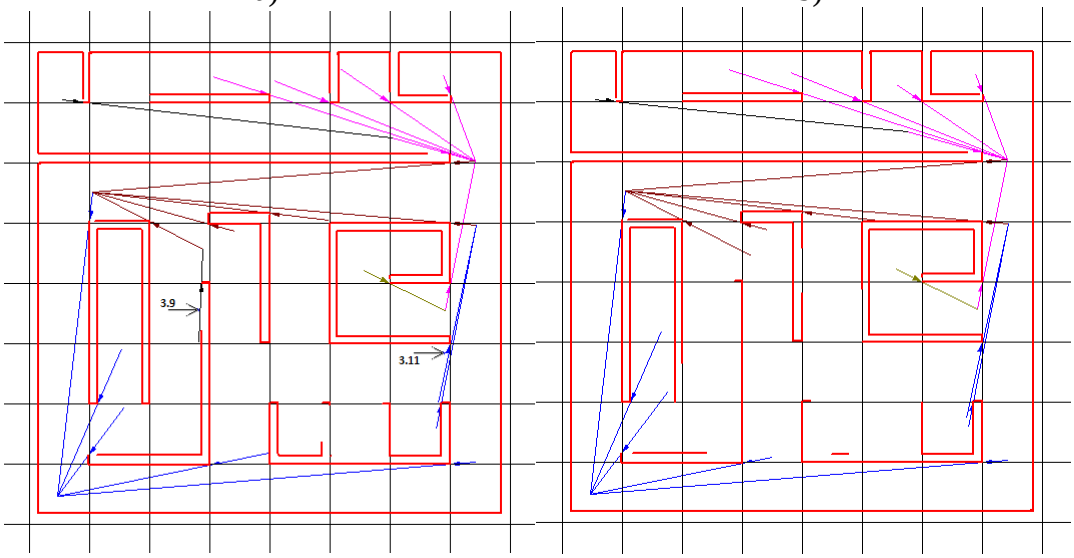
Iš 2.30 paveiksle pateiktų eksperimentinių rezultatų nustatyta, kad per didelė minimalios matomos chordos reikšmė ženkliai įtakoja modeliavimo rezultatus, t. y. neberandamos tikslios vietos kur yra statinių kliūčių kampai, dėl per didelio laipsnių šuolio užfiksuojamos netikslios

vektorinių žymių vietos ir sudaromas klaidingas aplinkos žemėlapis. Palyginus 2.29 paveikslėlį su 2.30 pav. a dalimi žemėlapis skiriasi 18 %, su b dalimi 25 %, su c dalimi 33 %, su d dalimi 43 %.

Programinio paketo „CENTAURUS CPN“ aplinkoje parametras vektorinės žymės ilgis (gembė) yra svarbus faktorius, kuris įtakoja vektorinio medžio išsidėstymą. Gauti eksperimentiniai rezultatai vaizduojami 2.31 pav.



a) b)



c) d)

2.31 pav. Eksperimentiniai rezultatai keičiant gembės reikšmę.

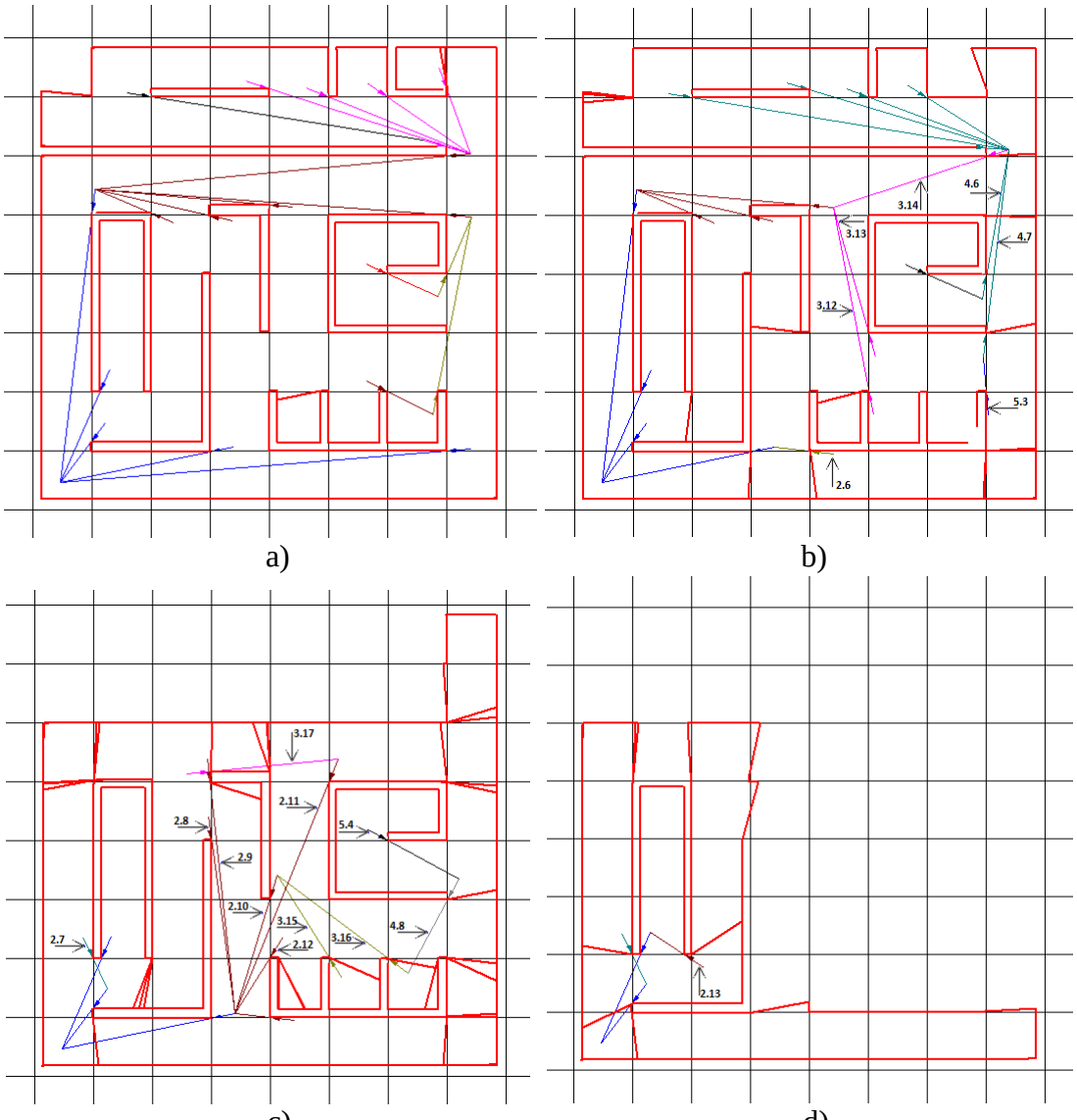
2.31 paveikslo a dalyje vektorinės žymės ilgis – 2, b dalyje – 12, c dalyje – 17, d dalyje - 22.

Kaip matoma iš 2.31 paveikslo pakeitus gembės reikšmę į 2 pasikeičia vektorinis medis, numeriais 2.6, 3.8, 4.3, 4.4, 4.5, 5.1, 5.2 pažymėta naujai atsiradusios vektorinio medžio atšakos. Lyginant su 2.29 sudarytu žemėlapiu išnyko šakos 2.3, 2.4, 3.1, 3.2, 3.3, 3.4, 3.5, 4.3, 4.4. Taip pat nepilnai sudaromas 8 statinės kliūties kontūras. Padidinus gembės reikšmę iki 12 lyginant su 2.31 pav. a dalimi išnyksta 4.2 šaka, šakos 3.1, 3.2 pakeičia pradžios taško vietą, 6 statinės kliūties

viršutiniame kairiajame kampe nepilnai suformuojamas kontūras. Pakeitus gembės reikšmę į 17 lyginant su 2.31 pav. b dalimi 3.9, 3.11 pažymėta naujai atsiradusios vektorinio medžio atšakos, taip pat nepilnai sudaromas 9, 10 statinių kliūčių kontūrai. Nustačius vektorinės žymės ilgį – 22 vektorinės šakų skaičius ir išsidėstymas nepakinta lyginant su 2.31 pav. c dalimi., tačiau nepilnai sudaromas 6, 7, 8, 9, 10 statinių kliūčių kontūrai.

Iš 2.31 paveiksle pateiktų eksperimentinių rezultatų nustatyta, kad vektorinės žymės ilgis, kiek ji išsikiša už nustatytos kliūties, turi didelę reikšmę vektorinio medžio struktūrai ir modeliavimo rezultatams. Palyginus 2.29 paveikslėlį su 2.31 pav. a dalimi žemėlapis skiriasi 8 %, su b dalimi 1 %, su c dalimi 5 %, su d dalimi 10 %.

Programinio paketo „CENTAURUS CPN“ aplinkoje atliekami modeliavimo eksperimentai keičiant radaro nejautrą. Aplinkos žemėlapyje, lyginant su 2.29 paveikslėlyje gautu aplinkos žemėlapiu atsiranda menamų statinių kliūčių. Eksperimentiniai rezultatai pateikti paveiksle 2.32.



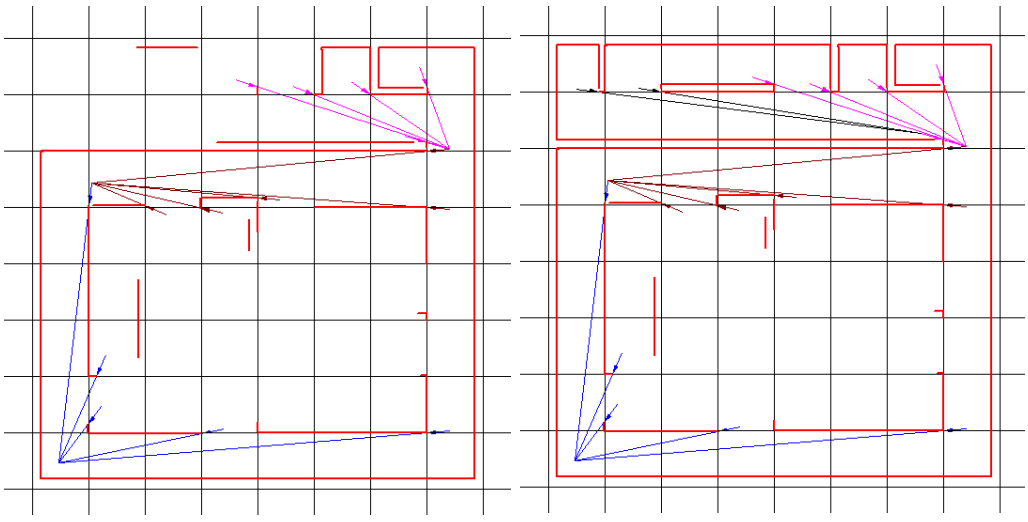
2.32 pav. Eksperimentiniai rezultatai keičiant radaro nejautrą.

2.32 paveikslo a dalyje radaro nejautra – 13, b dalyje – 15, c dalyje – 17, d dalyje - 19.

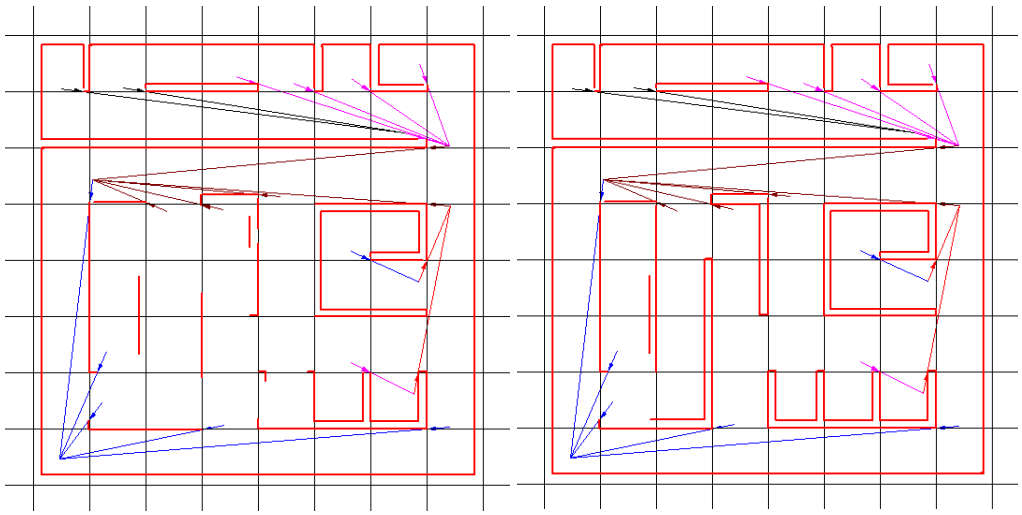
Kaip matoma iš 2.32 paveikslo padidinus radaro nejautrą iki 13, statinėse kliūtyse 1, 4, 10 gaunamos menamos statinė kliūtis, kurių aplinkoje nėra, lyginant su 2.29 pav. išnyksta 4.1 vektorinė šaka. Pakeitus reikšmę į 15 atsiranda menamos statinės kliūtys šalia 5, 6, 7, 8, 9 statinių kliūčių, taip pat nepilnai sudaromas 10 statinės kliūties kontūras. Pasikeičia vektorinio medžio išsidėstymas, palyginus 2.29 pav. pateiktu vektoriniu medžiu išnyksta šakos 1.5, 2.4, 2.5, 3.1, 3.2, 3.6, 4.4, 5.2, taip pat atsiranda naujų šakų 2.6, 3.12, 3.13, 3.14, 4.6, 4.7, 5.3. Nustačius reikšmę – 17 susiformuoja menamos statinės kliūtys šalia visų statinių kliūčių, 1, 2, 3 statinių kliūčių neužfiksuojama, nepilnai suformuojamas žemėlapio kontūras. Atsiranda naujos vektorinė šakos 2.7 – 2.12, 3.15 – 3.17, 4.8, 5.4. Įvedus radaro nejautrą – 19 išlieka 5, 6, 7, 9, 10 statinių kliūčių fragmentai su menamomis statinėmis kliūtimis, nepilnai suformuojamas žemėlapio kontūras, atsiranda nauja vektorinė šaka 2.13.

Iš paveiksle 2.32 pateiktų eksperimentinių rezultatų nustatyta, kad gretimų radaro spindulių ilgio šuolis yra per mažas, kad būtų užfiksuoti reikalingi trūkio taškai. Pasikeičia vektorinis medis, žemėlapis gaunamas su menamomis statinėmis kliūtimis. Palyginus 2.29 paveikslėlį su 2.32 pav. a dalimi žemėlapis skiriasi 5 %, su b dalimi 15 %, su c dalimi 46 %, su d dalimi 75 %.

Programinio paketo „CENTAURUS CPN“ aplinkoje keičiamas parametras yra kiek vektorinių žymių kartų daugiausia gali būti suformuota (maksimalus žingsnių skaičius), eksperimento rezultatai pateikti paveiksle 2.33.



a) b)



c) d)

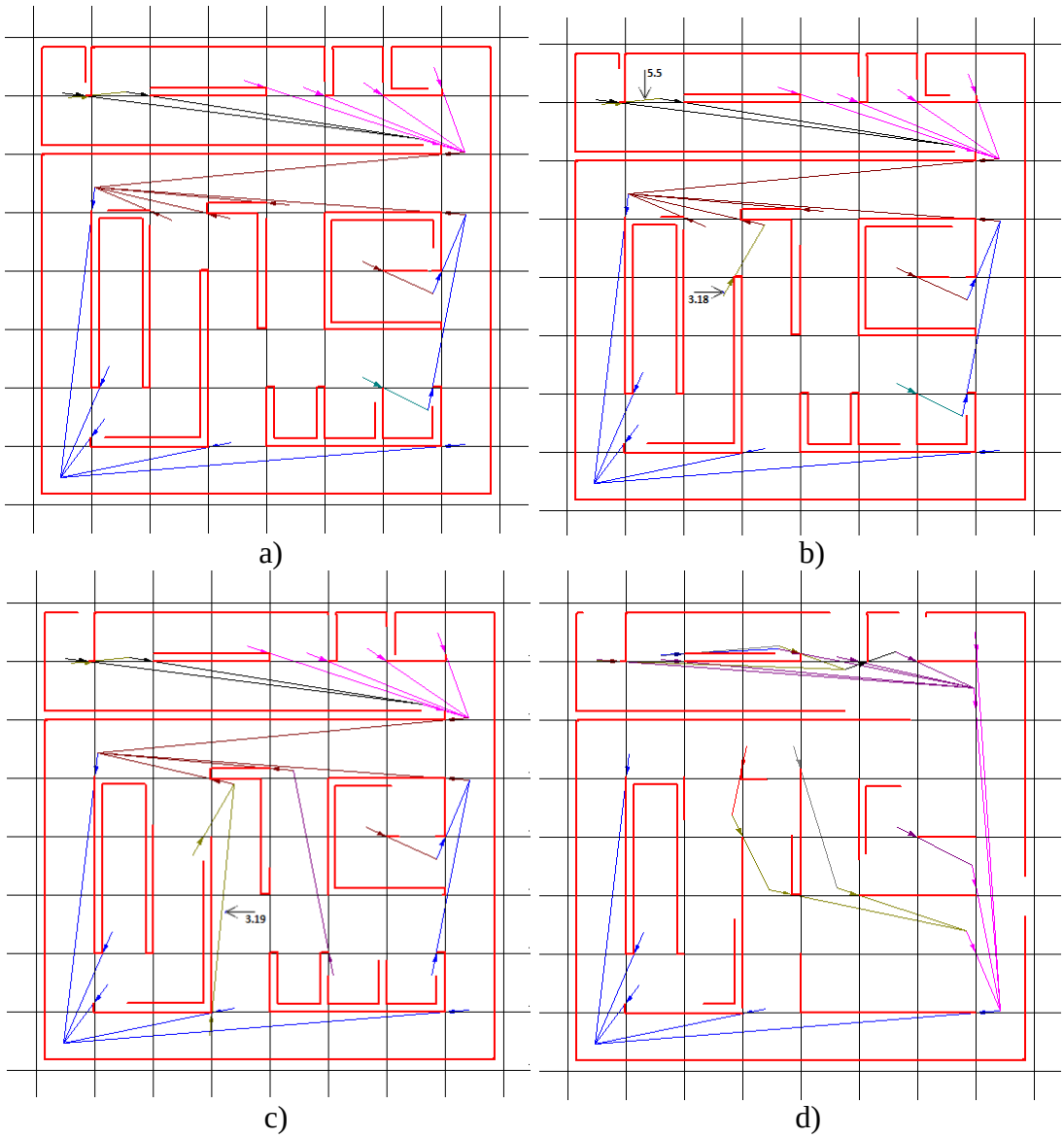
2.33 pav. Eksperimentiniai rezultatai keičiant maksimalų žingsnių skaičių.

2.33 paveikslo a dalyje maksimalus žingsnių skaičius – 5, b dalyje – 10, c dalyje – 15, d dalyje - 20.

Kaip matoma iš 2.33 paveikslo nustačius 5 vektorinius žingsnius žemėlapio kontūras, 2 - 10 statinės kliūtys nėra pilnai išbaigtos, 1 statinė kliūtis neužfiksuota. Padidinus reikšmę iki 10 visos statinės kliūtys yra neišbaigtos. Pakeitus žingsnių skaičių į 15, 6 - 10 statinių kliūčių kontūrai nėra pilnai išbaigti. Įvedus maksimalių žingsnių skaičių – 20 neišbaigti 6, 9 statinių kliūčių kontūrai.

Iš paveiksle 2.33 pateiktų eksperimentinių rezultatų nustatyta, kad nurodytų kartų skaičiaus nepakanka, norint pilnai suformuoti vektorinių žymių medį ir sudaryti išsamų žemėlapį. Palyginus 2.29 paveikslėlį su 2.33 pav. a dalimi žemėlapis skiriasi 50 %, su b dalimi 37 %, su c dalimi 12 %, su d dalimi 6 %.

Programinio paketo „CENTAURUS CPN“ aplinkoje atliekami modeliavimo eksperimentai keičiant radaro apžvalgos matymo kampą. Lyginant su 2.29 paveikslėlyje gautu aplinkos žemėlapiu suformuojamas neišbaigtas aplinkos žemėlapis. Eksperimentiniai rezultatai pateikti paveiksle 2.34.



a) b)

c) d)

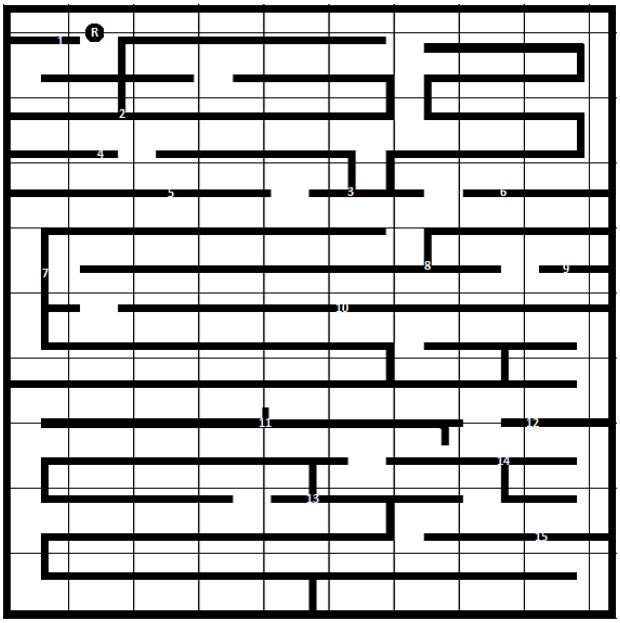
2.34 pav. Eksperimentiniai rezultatai keičiant radaro apžvalgos kampą.

2.34 paveikslo a dalyje apžvalgos kampas – 290º, b dalyje – 240º, c dalyje – 190º, d dalyje - 140º.

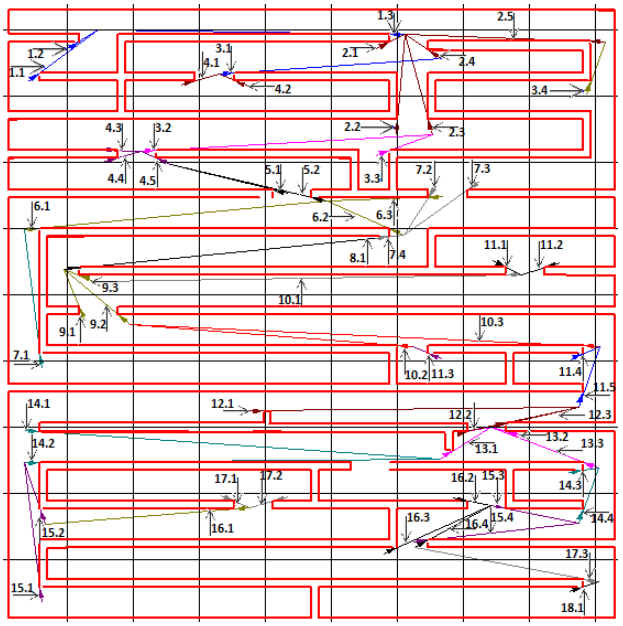
Kaip matoma iš paveikslo 2.34 a dalies sumažinus radaro apžvalgos kampą iki 290º nepilnai sudaromas visų statinių kliūčių kontūrai. Pakeitus reikšmę į 140 º atsiranda 2 naujos vektorinės šakos 3.18, 5.5. Sumažinus parametrą iki 190º žemėlapio kontūras nepilnai užfiksuojamas. Lyginant su 2.29 pav. išnyksta vektorinio medžio šakos 2.1, 4.4, atsiranda nauja šaka 3.19.

Iš paveiksle 2.34 pateiktų eksperimentinių rezultatų nustatyta, kad pasirinkus per mažą apžvalgos kampą randami ne visi trūkio taškai ir sudaromas klaidingas aplinkos žemėlapis. Palyginus 2.29 paveikslėlį su 2.34 pav. a dalimi žemėlapis skiriasi 4 %, su b dalimi 7 %, su c dalimi 20 %, su d dalimi 35 %.

Programinio paketo „CENTAURUS CPN“ aplinkoje sudaroma aplinka 2.35 pav. ir pažymima statinės kliūtys numeriais „1“, „2“, „3“, „4“, „5“, „6“, „7“, „8“, „9“, „10“, „11“ , „12“, „13“, „14“, „15“. Roboto pradinė pozicija aplinkoje pažymėta raide R.



2.35 pav. Sudaryta aplinka



2.36 pav. Aplinkos žemėlapis

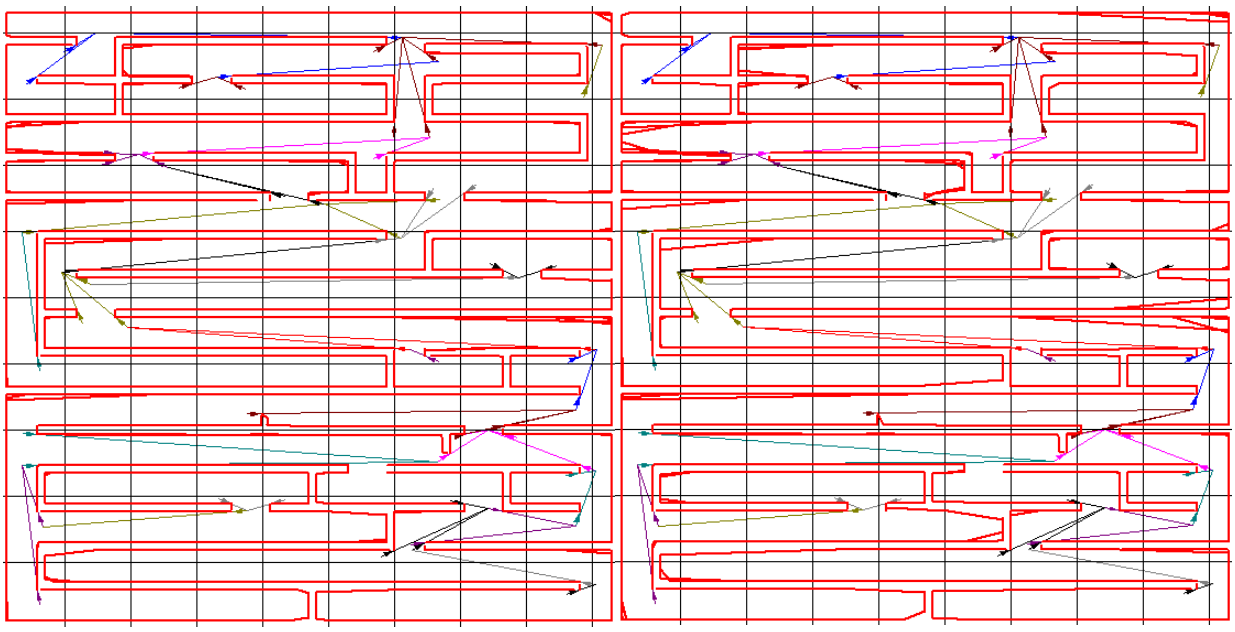
2.36 pav. parodyta kaip atrodo sudarytas aplinkos žemėlapis pagal 2.4 lentelėje pateiktus duomenis

2.4 lentelė. Aplinkos žemėlapio sudarymo duomenys

Minimali matoma chorda	Gembė	Radaro nejautra	Maksimalus žingsnių skaičius	Apžvalgos kampas
0,5	4	1	200	340^{o}

Sudaromas vektorinių žymių medis atsižvelgiant į skenavimo metu aptiktas statines kliūtis. Sudarytame aplinkos žemėlapyje pažymimos vektorinės šakos, pirmasis skaičius žymi kartos eilę, antrasis šakos skaičių kartoje. Paveiksluose 2.37 – 2.41 parodyta kaip kiekvienas iš parametrų keičia sudaromą žemėlapį. Parametrai keičiami po vieną, kelis kartus. Statinių žymių numeravimas ir roboto pradinė pozicija nesikeičia.

Programinio paketo „CENTAURUS CPN“ aplinkoje atliekami modeliavimo eksperimentai keičiant apžvalgos matymo kampo žingsnį (minimalią matomą chordą). Aplinkos žemėlapyje, lyginant su 2.36 paveikslėlyje gautu aplinkos žemėlapiu atsiranda netikslumų. Gauti eksperimentiniai rezultatai vaizduojami 2.37 pav.



a) b)

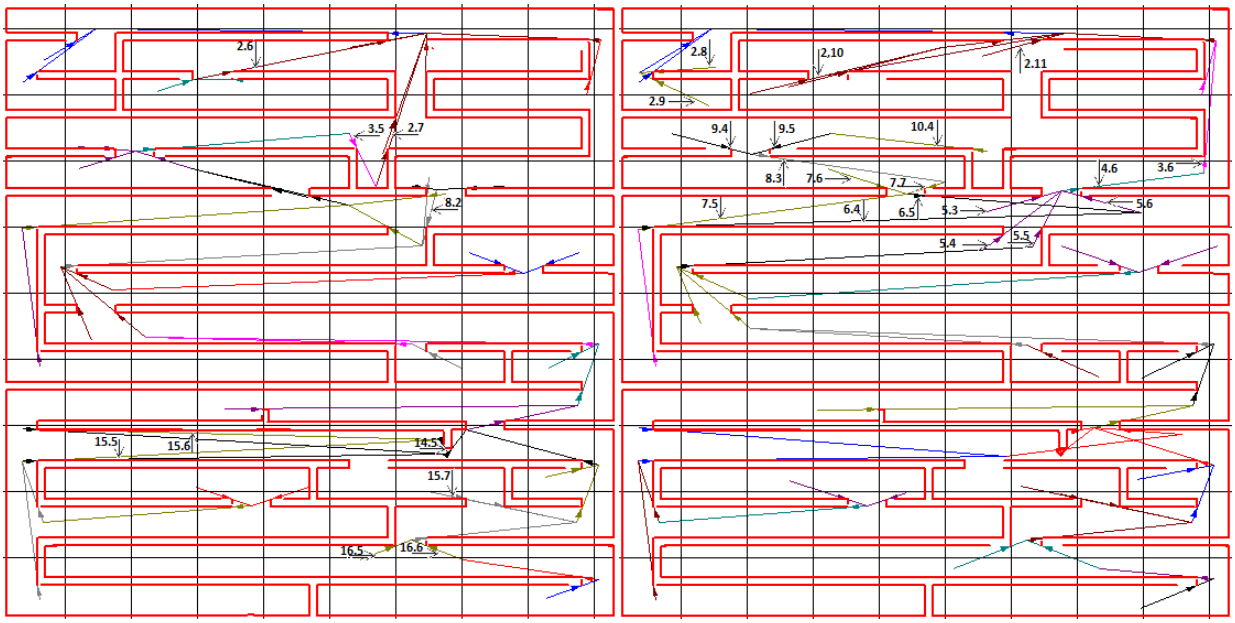
2.37 pav. Eksperimentiniai rezultatai keičiant apžvalgos matymo kampo žingsnio reikšmę.

2.37 paveikslo a dalyje chordos ilgis – 4, b dalyje – 8.

Kaip matoma iš 2.37 a dalies paveikslo keičiant apžvalgos matymo kampo žingsnio reikšmę į 4 nedideli netikslumai matomi žemėlapio kontūro dešiniajame viršutiniame kampe, 2, 4, 5, 7, 10, 13, 15 statinėse kliūtyse. Didinant apžvalgos matymo kampo žingsnį iki 8 minėti netikslumai a dalyje išryškėja, taip pat atsiranda naujų 3, 8 statinėse kliūtyse.

Iš paveiksle 2.37 pateiktų eksperimentinių rezultatų nustatyta, kad per didelė minimalios matomos chordos reikšmė ženkliai įtakoja modeliavimo rezultatus, t. y. neberandamos tikslios vietos kur yra statinių kliūčių kampai, dėl per didelio laipsnių šuolio užfiksuojamos netikslios vektorinių žymių vietos ir sudaromas klaidingas aplinkos žemėlapis. Palyginus 2.36 paveikslėlį su 2.37 pav. a dalimi žemėlapis skiriasi 15 %, su b dalimi 20 %.

Programinio paketo „CENTAURUS CPN“ aplinkoje parametras vektorinės žymės ilgis yra svarbus faktorius, kuris įtakoja vektorinio medžio išsidėstymą. Gauti eksperimentiniai rezultatai vaizduojami 2.38 pav.



a) b)

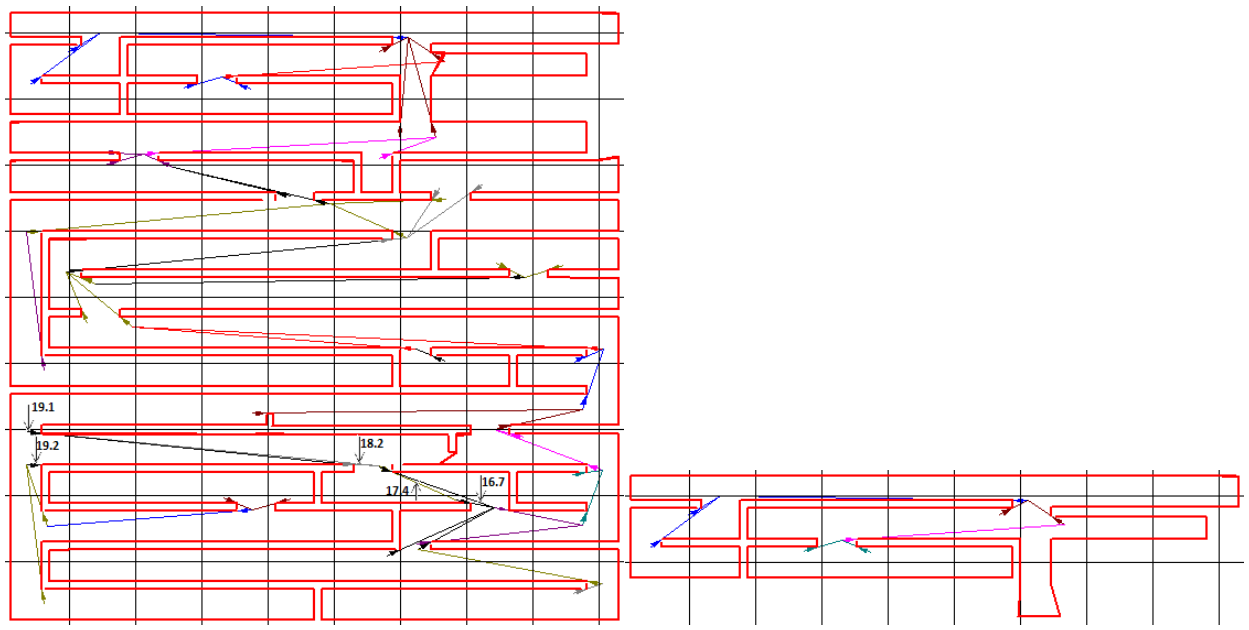
2.38 pav. Eksperimentiniai rezultatai keičiant vektorinės žymės ilgį.

2.38 paveikslo a dalyje vektorinės žymės ilgis ilgis – 10, b dalyje – 16.

Kaip matoma iš 2.38 a dalies paveikslo pakeitus gembės reikšmę į 2 pasikeičia vektorinis medis, numeriais 2.6, 2.7, 3.5, 8.2, 14.5, 15.5, 15.6, 16.5, 16.6 pažymėta naujai atsiradusios vektorinio medžio atšakos. Lyginant su 2.36 pav. sudarytu žemėlapiu išnyko šakos 2.3, 3.3, 7.3, 12.2, 16.2, 16.3, 16.4, taip pat nepilnai sudaromas 3 statinės kliūties kontūras. Padidinus gembės reikšmę iki 16 lyginant su 2.38 pav. a dalimi vektorinėmis šakomis 2.8 - 2.11, 3.6, 4.6, 5.3 – 5.6, 6.4, 6.5, 7.5 – 7.7, 8.3, 9.4, 9.5, 10.4 pažymėta naujai atsiradusios vektorinio medžio atšakos, taip pat nepilnai sudaromas 2, 4, 7, 11, 14, 15 statinių kliūčių kontūrai.

Iš paveiksle 2.38 pateiktų eksperimentinių rezultatų nustatyta, kad vektorinės žymės ilgis, kiek ji išsikiša už nustatytos kliūties, įtakoja vektorinio medžio struktūrą ir modeliavimo rezultatus. Palyginus 2.36 paveikslėlį su 2.38 pav. a dalimi žemėlapis skiriasi 11 %, su b dalimi 4 %.

Programinio paketo „CENTAURUS CPN“ aplinkoje atliekami modeliavimo eksperimentai keičiant radaro nejautrą. Aplinkos žemėlapyje, lyginant su 2.36 paveikslėlyje gautu aplinkos žemėlapiu atsiranda menamų statinių kliūčių. Eksperimentiniai rezultatai pateikti paveiksle 2.39.



a) b)

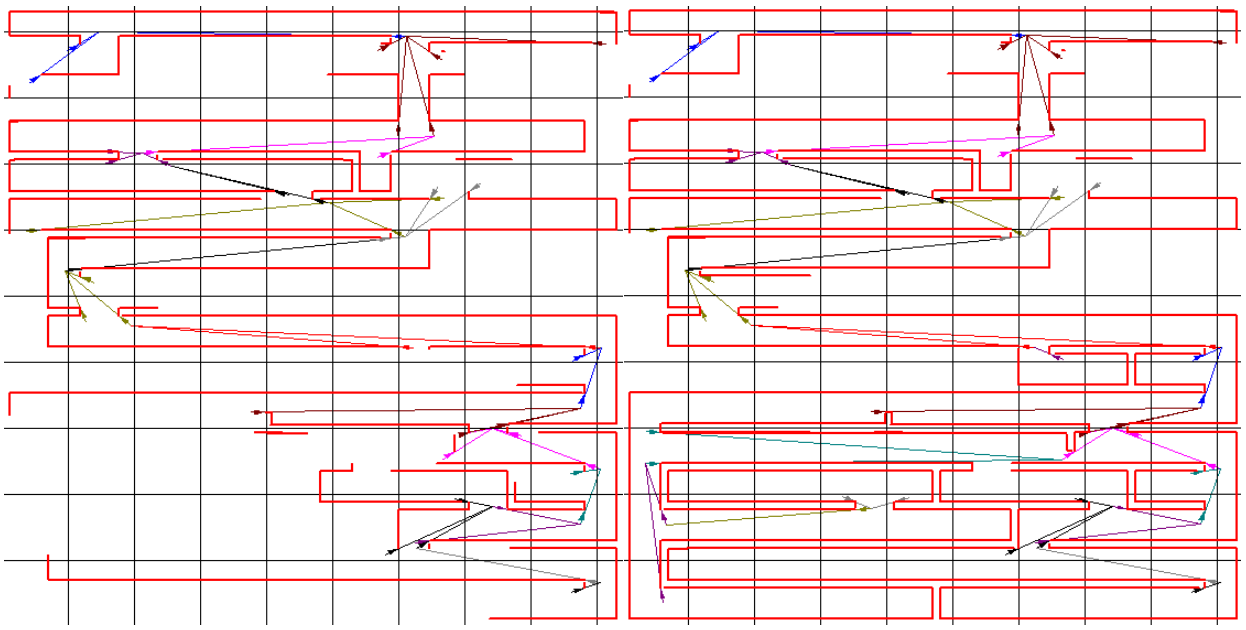
2.39 pav. Eksperimentiniai rezultatai keičiant radaro nejautrą.

2.39 paveikslo a dalyje radaro nejautra – 8, b dalyje – 9.

Kaip matoma iš 2.39 paveikslo a dalies padidinus radaro nejautrą iki 8, statinėse kliūtyse 3, 11 gaunamos menamos statinės kliūtys, kurių aplinkoje nėra, nepilnai suformuojamas žemėlapio kontūras, išnyksta 2.5, 3.4, 12.1, 12.2, 13.1, 14.1, 14.2 vektorinės šakos lyginant su 4.36 pav., taip pat atsiranda naujų šakų 16.7, 17.4, 18.2, 19.1, 19.2. Pakeitus reikšmę į 9 atsiranda menamos statinės kliūtys tarp 2 ir 3 statinių kliūčių.

Iš paveiksle 2.39 pateiktų eksperimentinių rezultatų nustatyta, kad gretimų radaro spindulių ilgio šuolis yra per mažas, kad būtų užfiksuoti reikalingi trūkio taškai. Pasikeičia vektorinis medis, žemėlapis gaunamas su menamomis statinėmis kliūtimis, taip pat neužfiksavus reikiamų trūkio taškų gaunamas nepilnas aplinkos žemėlapis. Palyginus 2.36 paveikslėlį su 2.39 pav. a dalimi žemėlapis skiriasi 14 %, su b dalimi 86 %.

Programinio paketo „CENTAURUS CPN“ aplinkoje keičiamas parametras yra kiek vektorinių žymių kartų daugiausia gali būti suformuota (maksimalus žingsnių skaičius), eksperimento rezultatai pateikti paveiksle 2.40.



a) b)

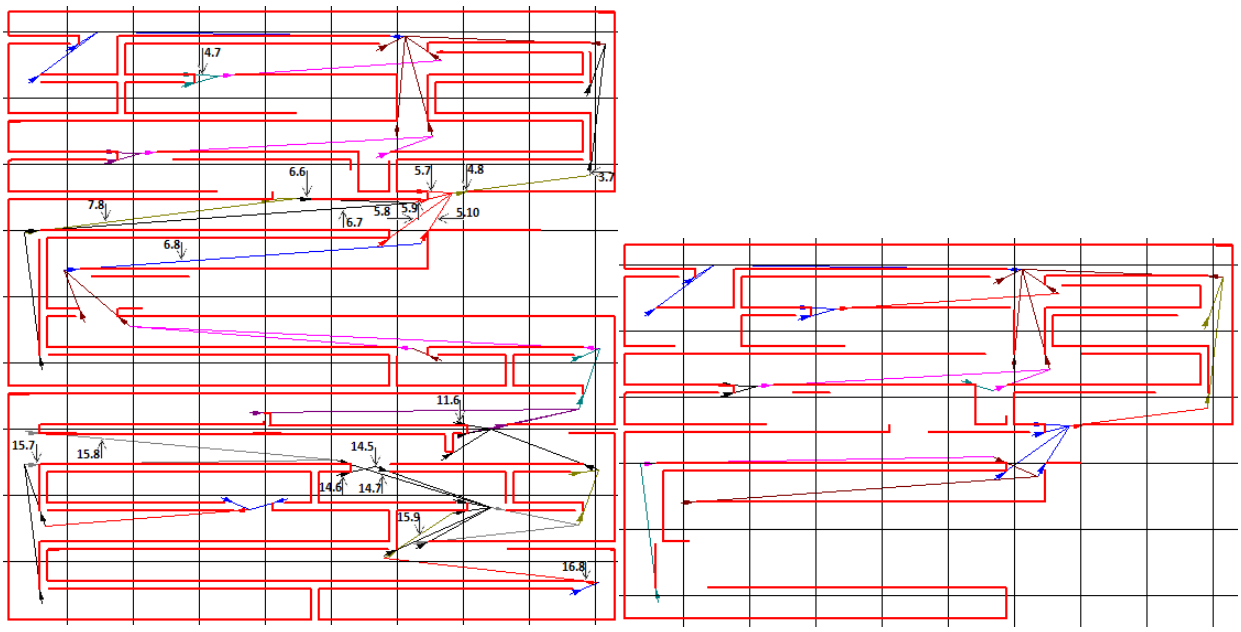
2.40 pav. Eksperimentiniai rezultatai keičiant maksimalų žingsnių skaičių.

2.40 paveikslo a dalyje maksimalus žingsnių skaičius – 20, b dalyje – 40.

Kaip matoma iš 2.40 paveikslo a dalies nustačius 20 vektorinius žingsnius pilnas yra tik 4 statinės kliūties kontūras, 9 statinė kliūtis neužfiksuota. Padidinus reikšmę iki 40 pilni kontūrai suformuojami 11, 12, 13, 15 statinių kliūčių.

Iš paveiksle 2.40 pateiktų eksperimentinių rezultatų nustatyta, kad nurodytų kartų skaičiaus nepakanka, norint pilnai suformuoti vektorinių žymių medį ir sudaryti išsamų žemėlapį. Palyginus 2.36 paveikslėlį su 2.40 pav. a dalimi žemėlapis skiriasi 45 %, su b dalimi 25 %.

Programinio paketo „CENTAURUS CPN“ aplinkoje atliekami modeliavimo eksperimentai keičiant radaro apžvalgos matymo kampą. Lyginant su 2.36 paveikslėlyje gautu aplinkos žemėlapiu suformuojamas neišbaigtas aplinkos žemėlapis. Eksperimentiniai rezultatai pateikti paveiksle 2.41.



a) b)

2.41 pav. Eksperimentiniai rezultatai keičiant radaro apžvalgos kampą.

2.41 paveikslo a dalyje apžvalgos kampas – 300º, b dalyje – 200º.

Kaip matoma iš 2.41 paveikslo a dalies sumažinus radaro apžvalgos kampą iki 300º nepilnai sudaromas žemėlapio ir 2, 3, 5 – 8, 10, 12 - 15 statinių kliūčių kontūrai, 9 statinės kliūties neužfiksuojama. Lyginant su 2.36 pav. vektoriniu medžiu išnyksta vektorinio medžio šakos 4.2, 4.5, 5.1, 5.2, 6.1 - 6.3, 7.2 – 7.4, 8.1, 9.3, 10.1, 11.1, 11.2, 14.1, 14.2, 17.3 ir atsiranda naujų šakų 3.7, 4.7, 4.8, 5.7 – 5.10, 6.6 – 6.8, 7.8, 11.6, 14.5 – 14.7, 15.7 – 15.9, 16.8. Pakeitus reikšmę į 200º 10 – 15 statinės kliūtys neužfiksuojamos, pilnai sudaromas tik 4 statinės kliūties kontūras, taip pat sumažėja vektorinio medžio šakų.

Iš paveiksle 2.41 pateiktų eksperimentinių rezultatų nustatyta, kad pasirinkus per mažą apžvalgos kampą randami ne visi trūkio taškai ir sudaromas nepilnas, klaidingas aplinkos žemėlapis. Palyginus 2.36 paveikslėlį su 2.41 pav. a dalimi žemėlapis skiriasi 12 %, su b dalimi 60 %.

Atlikus modeliavimo eksperimentus programinio paketo „CENTAURUS CPN“ aplinkoje iš surinktų duomenų pateiktų 2 skyriaus 2.14 – 2.41 paveiksluose galima teigti, kad spalvotus Petri tinklus ir pasiūlytą vektorinių medžių generavimo metodiką galima pritaikyti nežinomos aplinkos skenavimui bei žemėlapio sudarymui.

Išvados ir rezultatai

1. Atlikus literatūros analizę nustatyta, kad egzistuoja kelios robotų žemėlapio nežinomoje aplinkoje sudarymo galimybės. Dažniausiai jos skirstomos į dvi dalis: kai naudojami robotai realioje aplinkoje ir kai modeliuojama virtualioje aplinkoje. Nustatyta, kad mobilaus roboto žemėlapio sudarymas reikalauja didelio tikslumo ir greitaveikos.
2. Eksperimentiškai nustatyta, kad žemėlapio nežinomoje aplinkoje sudarymui gerai tinka vektorinių žymių generavimo metodas. Ištyrus pastebėta, kad statinių kliūčių kampuose fiksuojami skenerio spindulio ilgio šuoliai, fiksuojant šias vietas kaip trūkio taškus. Sukurto metodo esmė yra aptikti statinių kliūčių pokyčio vietas. Trūkio taškuose formuojamos virtualios vektorinės žymės. Skenuojant virtualią aplinką sudaromas vektorinių žymių medis pagal kurį sudarinėjamas aplinkos žemėlapis.
3. Programinio paketo „CENTAURUS CPN“ aplinkoje grafiškai sudarytos aplinkos ir jas analizuojant ištirta kaip kiekvienas iš parametrų, kurie gali įtakoti vektorinių žymių medžio struktūrą bei žemėlapio tikslumą, keičia sudaromą žemėlapį. Ištirta, kad priklausomai nuo aplinkos konfiguracijos sudėtingumo, kintant apžvalgos matymo kampo žingsniui nuo 4 iki 30, aplinkos žemėlapiuose nesutapimai su sudaryta aplinka kinta nuo 6 % iki 43 %, didėjant vektorinės žymės ilgiui nuo 2 iki 22, nesutapimai padidėja nuo 1 % iki 11 %, keičiant radaro nejautrą nuo 8 iki 19, neatitikimai keičiasi nuo 3 % iki 86 %, sumažinus maksimalų žingsnių skaičių nuo 40 iki 2, nesutapimai padidėja nuo 6 % iki 50 %, didėjant radaro apžvalgos matymo kampui nuo 120^{o} iki 300^{o}, neatitikimai mažėja nuo 60 % iki 1 %.
4. Atlikus tyrimą nustatyta, kad vektorinių žymių generavimo metodą galima naudoti žemėlapio kūrimui nežinomoje aplinkoje prieš tai, rankiniu būdu įvedus radaro nejautros dydį, minimalios matomos chordos reikšmę, maksimalų žingsnių skaičių, skenerio apžvalgos kampą, gembės reikšmę. Įvedus tinkamus parametrus gaunamas tikslus ir išbaigtas žemėlapis.

Literatūros sąrašas

1. Erik Zamora Gómez: *Map-building and planning for autonomous navigation of a mobile robot,* [žiūrėta:2015-01-18]. Prieiga per internetą: http://www.ctrl.cinvestav.mx/~yuw/pdf/ErikZG.pdf
2. Sergio Almansa-Valverde a, José Carlos Castillo a, Antonio Fernández-Caballero: *Mobile robot map building from time-of-flight camera* [žiūrėta:2014-05-16]. Prieiga per internetą: http://www.sciencedirect.com/science/article/pii/S0957417412002485
3. Ales Jelinek: *Vector Maps in Mobile Robotics* [žiūrėta:2014-11-18]. Prieiga per internetą: http://robotics.fel.cvut.cz/pair14/wp-content/uploads/2014/09/pair14_submission-jelinek.pdf
4. Mohammad Al Khawaldah: *Multi-robot Exploration of Unknown Environment Using 2D Laser Scanner* [žiūrėta:2014-05-16]. Prieiga per internetą: http://maxwellsci.com/print/rjaset/v7-5057-5062.pdf
5. Ciprian-Florin Ciuea, Adrian-Vasile Duka, Stelian-Emilian Olteam: *Automatic mapping of an enclosure using a mobile robot* [žiūrėta:2014-10-17]. Prieiga per internetą: http://www.sciencedirect.com/science/article/pii/S2212017313006403
6. Zhu Jianguo, Gao Junyao, Li Kejie, Ren Peijie, Yin Qiang: *Map Building for Indoor Tracked Autonomous Mobile Robot with Laser Rangefinder and Electronic Compass* [žiūrėta:2014-05-08]. Prieiga per internetą: http://www.ipcsit.com/vol51/043-A679.pdf
7. Senka Krivic, Aida Mrzic, Nedim Osmic: *Building Mobile Robot and Creating Applications for 2D Map Building and Trajectory Control* [žiūrėta:2014-05-08]. Prieiga per internetą: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5967338&tag=1
8. Justė Matusevičienė: *Mobilaus roboto grįžtamojo ryšio sistemos sukūrimas* [žiūrėta:2014-04-18]
9. Roland, Siegwart; Illah R., Nourbakhsh: *Introduction to autonomous mobile robots*. Cambridge, Massachusetts; London, England, [žiūrėta:2014-11-18]. Prieiga per internetą: http://home.deib.polimi.it/gini/robot/docs/siegwart.pdf
10. Cuesta, Federico; Ollero, Anibal. *Intelligent mobile robot navigation // Springer Tracts in Advanced robotics, Volume 16* [žiūrėta:2014-11-23]
11. J. Elseberg, R. T. Creed, and R. Lakaemper, *A line segment based system for 2D global mapping*, 2010 IEEE International Conference on Robotics and Automation, [žiūrėta:2014-11-18].
12. V. Baranauskas, *ROBOTŲ NAVIGACIJA TRUMPIAUSIO KELIO PAIEŠKOS METODU*, [žiūrėta:2014-04-23]

13. Arbnor Pajaziti, Petrit Avdullahu, *SLAM – Map Building and Navigation via ROS*, [žiūrėta:2014-05-18]. Prieiga per internetą: http://ijisae.atscience.org/article/download/1065000115/pdf_2
14. Jan Elseberg, Ross T. Creed, Rolf Lakaemper, *A Line Segment Based System for 2D Global Mapping*, [žiūrėta:2014-05-18]. Prieiga per internetą: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.473.4460&rep=rep1&type=pdf
15. S. Bartkevičius, V. Mačerauskas, K. Šarkauskas, *Spalvotųjų Petri tinklų taikymas valdymo sistemoms modeliuoti*, [žiūrėta:2014-04-28]. Prieiga per internetą: http://www.ee.ktu.lt/journal/2003/4/Bartkevicius.pdf
16. S. Bartkevičius, K. Šarkauskas, *Programų paketas CENTAURUS. Modeliavimas, identifikavimas, optimizavimas*, [žiūrėta:2014-04-28]. Prieiga per internetą: https://www.ebooks.ktu.lt/einfo/170/programu-paketas-centaurus-modeliavimas-identifikavimas-optimizavimas/
17. Michael Montemerlo, Sebastian Thrun, Daphne Koller, Ben Wegbreit, FastSLAM: *A Factored Solution to the Simultaneous Localization and Mapping Problem*, [žiūrėta:2014-11-01]. Prieiga per internetą: http://ai.stanford.edu/~koller/Papers/Montemerlo+al:AAAI02.pdf
18. Maki K. Habib, *Real Time Mapping and Dynamic Navigation for Mobile Robots*, [žiūrėta:2014-10-20]. Prieiga per internetą: http://cdn.intechopen.com/pdfs-wm/4239.pdf
19. Gamini Dissanayake, Hugh DurrantWhyte, Tim Bailey, *Computationally effcient Solution to the Simultaneous Localisation and Map Building SLAM Problem*, [žiūrėta:2014-10-20]. Prieiga per internetą: http://static.aminer.org/pdf/PDF/000/351/774/a_computationally_efficient_solution_to_the_simultaneous_localisation_and_map.pdf
20. Megan R. Naminski, *An Analysis of Simultaneous Localization and Mapping (SLAM) Algorithms*, [žiūrėta:2014-04-20]. Prieiga per internetą: http://digitalcommons.macalester.edu/cgi/viewcontent.cgi?article=1030&context=mathcs_honors