KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

PROGRAMŲ INŽINERIJOS KATEDRA

AISTIS KARKAUSKAS

KLIŪČIŲ VENGIMO METODO TYRIMAS MOBILIOJE AUTONOMINĖJE PLATFORMOJE

Magistro darbas

Vadovas

prof. dr. R. Damaševičius

KAUNAS, 2014

KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

PROGRAMŲ INŽINERIJOS KATEDRA

AISTIS KARKAUSKAS

KLIŪČIŲ VENGIMO METODO TYRIMAS MOBILIOJE AUTONOMINĖJE PLATFORMOJE

Magistro darbas

Recenzentas

dr. D. Barisas

Vadovas

prof. dr. R. Damaševičius

Darbą atliko

IFM-2/2 gr. stud.

A. Karkauskas

KAUNAS, 2014

PATVIRTINIMAS APIE ATLIKTO DARBO SAVARANKIŠKUMĄ

Patvirtinu, kad įteikiamas baigiamasis darbas „Kliūčių vengimo metodo tyrimas mobilioje autonominėje platformoje“:

1. Autoriaus atliktas savarankiškai, jame nėra pateikta kitų autorių medžiagos kaip savos, nenurodant tikrojo šaltinio.
2. Nebuvo to paties autoriaus pristatytas ir gintas kitoje mokymo įstaigoje Lietuvoje ar užsienyje.
3. Nepateikia nuorodų į kitus darbus, jeigu jų medžiaga nėra naudota darbe.
4. Pateikia visą naudotos literatūros sąrašą.

Aistis Karkauskas		
(studento vardas, pavardė)	(data)	(parašas)

SANTRAUKA

Magistriniame darbe „Kliūčių vengimo metodo tyrimas mobilioje autonominėje platformoje“ aprašomos problemos iškilusios projektuojant bei realizuojant roboto valdymo sistemą. Nagrinėjami iškilusių problemų sprendimo būdai. Įvertinus padėti, pasiūlyti bei įgyvendinti nauji metodai.

Darbo metu buvo atlikta projektą įtakojančių veiksnių analizė. Išnagrinėti galimi problemų sprendimo metodai. Suprojektuota bei sukurta sistema, kuri leidžia valdyti robotą, patikrinti naudojamų metodų teisingumą pasitelkiant grafinę vartotojo sąsają.

Galiausiai buvo realizuoti komponentai bei metodai skirti vykdyti kliūčių atpažinimą bei vengimą. Atlikti eksperimentai parodė, kad įgyvendinti algoritmai veikia teisingai. Darbo rezultatų išvados buvo grindžiamos atliktų eksperimentų rezultatais.

SUMMARY

In “Research of obstacle avoidance in autonomous mobile platform” work a set of problems emerged while the main robot control system was under development. To solve the identified problems a detailed analysis of existing solutions was performed. Ultimately having unsatisfactory results new methods were introduced and implemented in the system.

At first there was a need to perform an analysis of factors which can affect the project. Conducted analysis results have contributed to the development of the robot control system, which was used to examine and verify implemented algorithms.

Finally, at the end of the work a set of experiments were performed to examine implemented obstacle detection and avoidance methods. The conclusions were made based on the results of conducted experiments.

TURINYS

LENTELIŲ SĄRAŠAS

PAVEIKSLŲ SĄRAŠAS

TERMINŲ IR SANTRUMPŲ ŽODYNAS

- UGV (angl. Unmanned Ground Vehicle) – žmogaus nepilotuojama transporto priemonė;
- DARPA – (angl. The Defense Advanced Research Projects Agency). JAV agentūra, atsakina už naujų technologijų, skirtų karinei pramonei vystimasi;
- CSUF – (angl. California State University, Fullerton). Tai viešas universitetas esantis Fullerton mieste, Kalifornijoje;
- GPS – (angl. Global Positioning System) . Visuotinė padėties nustatymo sistema arba globali pozicionavimo sistema;
- **.**NET – .NET Framework yra Microsoft Windows operacinės sistemos komponentas, sukurtas 2002 metais. Jis suteikia kitoms programoms galimybę naudotis daugybe jau paruoštu įvairių bibliotekų;
- RGB – spalvų maišymo sistema, kurioje naudojamos trys, žmogaus akių receptorius atitinkančios spalvos: raudona (Red), žalia (Green) ir Mėlyna (Blue);
- MRDS – (angl. Microsoft Robotics Developer Studio). Windows OS aplinkoje veikti pritaikyta sistema skirta roboto programinei įrangai kurti, valdyti bei vykdyti simuliacijas;
- PĮ – (angl. Software). Programinė įranga (informacijos apdorojimo sistemos programų, procedūrų, taisyklių visuma arba tos visumos dalis kartu su atitinkama dokumentacija;
- TOF – (angl. Time Of Flight). Apibūdina įvairius metodus skirtus matuoti objekto vidutinį greitį jam judant nustatytą atstumą;
- SDK (angl. Software development kit). Programinės įrangos įrankiai įgalinantys darbą su tam tikrais PĮ paketais ar karkasais;
- PCL (angl. Point Cloud Library). Atvirojo kodo tipo projektas skirtas 2D/3D vaizdų bei taškų debesų apdorojimui;
- PID valdiklis – (angl. Proportional-integral-derivative controller). Proporcingo vientiso darinio valdiklis. Toks valdiklis įvertina „klaidos“ vertę, kuri yra skirtumas tarp išmatuoto proceso kintamojo ir nustatytos norimos reikšmės. Po to, valdiklis bando minimizuoti klaidos reikšmę koreguodamas iėjimus.

1. ĮVADAS

Mobilios autonominės platformos sistemos kūrimo pradžioje buvo identifikuotos problemos bei veiksniai, kurie gali įtakoti tam tikrų komponentų realizavimo ypatumus. Šiame darbe nagrinėjamos problemos yra susijusios su nuožulniųjų plokštumų, slenksčių bei tikrų kliūčių atpažinimu. Taip pat sprendžiamas roboto dydžio bei judėjimo trajektorijos priklausomumas nuo stebimų aplinkos objektų. Analizuojami paminėtų problemų sprendimai, pasiūlomi nauji metodai skirti sąlyginai apeiti iškilusias problemas.

1.1. Dokumento paskirtis

Šis dokumentas – tai Kauno technologijos universiteto Informatikos fakulteto Programų inžinerijos katedros studento Aisčio Karkausko magistrinis darbas.

Dokumente analizuojamos mobilios autonominės platformos judėjimo aplinkoje su statinėmis bei dinaminėmis kliūtimis problemos. Taip pat pateikiami šių problemų galimi sprendimo būdai bei realizuoto kliūčių vengimo metodo tyrimas.

Analitinėje dalyje analizuojamos įvairios problemos susijusios su kliūčių nustatymu. Identifikuojamos problemos, kurios yra būdingos sukurtai sistemai. Pateikiami įvairūs sprendimo būdai kiekvienai problemai.

Projektinėje dalyje pateikiamas studijų metu sukurtos programinės įrangos aprašymas bei struktūra.

Tiriamojoje dalyje pateikiami aprašomi sistemoje realizuoti metodai.

Galiausiai, eksperimentinių tyrimų dalyje pateikiami atlikti kliūčių vengimo metodo tyrmai.

2. ANALITINĖ DALIS

2.1. Sistemos apžvalga

Magistro darbo tikslas buvo sukurti programinę įrangą, kuri atlieka robotizuotos platformos valdymo funkciją, duomenų registravimą bei atvaizdavimą ekrane.

Dvi pagrindinės autonominės mobilios platformos veikimo funkcijos:

1) Gebėjimas nuvažiuoti į nustatytą tikslą
2) Žmonių atpažinimas ir sekimas

Dėl laiko stokos antrosios funkcijos realizavimas buvo atidėtas. Šiame darbe buvo įgyvendinta pirmoji veikimo funkcija – mobilios platformos gebėjimas nuvažiuoti į nustatytą tikslą.

2.2. Problemų analizė

Tikslo siekimo uždavinio įgyvendinimas reikalauja roboto lokalizacijos, žemėlapio kūrimo bei gautų duomenų iš jutiklių interpretavimo, apdorojimo problemos sprendimo.

Būtent pastaroji problema (duomenų apdorojimas) yra nagrinėjama šioje analitinėje dalyje. Verta paminėti, kad nagrinėjama informacija yra apie aplinkos objektus, kitaip – potencialias kliūtis. Kiti duomenys, kurie padeda apskaičiuoti roboto poziciją yra taip pat svarbūs sprendžiant lokalizacijos problemą. Sudarant žemėlapį yra reikalinga tiek roboto pozicija, tiek sudarytų objektų sąrašas.

Kliūčių vengimo strategijos pasirinkimas gali būti įtakojamas įvairių faktorių. Kiekvienas jų buvo išnagrinėtas atskirai:

2.2.1. Veikimo aplinka

Vienas iš esminių faktorių, kuriuos reikia apgalvoti kuriant UGV (angl. Unmanned Ground Vehicle) yra aplinka. Tai terpė, kurioje roboto platforma vykdo jai priskirtus veiksmus. Paprastai aplinka yra skirstoma į atvirąją (laukas) bei uždarąją (patalpa).

Atvirojoje aplinkoje arba kitaip lauko sąlygomis iškyla sekančios problemos:

- Nestruktūrizuota aplinka (nelygus paviršius, sunkiai identifikuojami objektai, pvz.: žolė, krūmai)
- Oro sąlygų poveikis (drėgmė, temperatūra, saulės skleidžiama šviesa)
- Dinaminių objektų įvairumas (žmonės, gyvūnai, mašinos ir t.t.)

Uždara patalpa yra sąlyginai draugiškesnė aplinka, tačiau ir čia susiduriama su tam tikromis problemomis:

- Dažnai pasitaikantys signalą iškraipantys veiksniai bei objektai (pvz.: metaliniai daiktai klaidina kompaso parodymus, stogas slopina GPS signalą.
- Nestandartiniai objektai (pvz.: stiklinės pertvaros, kurių negali užfiksuoti jutikliai naudojantys infraraudonuosius spindulius; laiptai, kurie reikalauja specialaus jutiklių išdėstymo)

Išanalizavus dažniausiai pasitaikančias aplinkos problemas buvo nuspręsta patikrinti, kurios problemos bus būdingos kuriamam UGV. Pagal reikalavimus, UGV turi funkcionuoti tiek patalpoje, tiek lauke. Tačiau laukas šiuo atveju būtų labiau struktūrizuotas (aikštelės tipas, lygus grindinys). Dažnai aikštelėse yra įrengtos rampos, skirtos neįgaliesiems arba dviračių vairuotojams). Jomis galėtų pasinaudoti ir roboto platforma, tačiau problema yra rampos atpažinimas. Rampa neturėtų būti klasifikuojama kaip kliūtis. Atvirkštinio atvejo, kai reikia judėti iš aukštesnės plokštumos į žemesnę nenagrinėjame. Taip yra todėl, kad roboto platforma, pagal reikalavimus turės lokalų žemėlapį, kuriame būtų sužymėtos kliūtys, pavyzdžiui laiptai.

2.2.2. Kainos ir kokybės santykis

Labai didelės agentūros ir kompanijos kaip NASA ar Google, kuri neseniai įsigijo DARPA atitinkamai gali skirti milžiniškas lėšas įvairiems projektams susijusiais su robotika. Kokybė šiuo atveju priklauso nuo investuojamų pinigų kiekio. Projektai išties priverčia stebėtis [1].

Šio projekto prototipui bei kartu galutiniam produktui tikimasi sunaudoti kuo mažiau kaštų, bet sukurti roboto komponentus, kurie užtikrintų reikiamą kokybę. Pavyzdžiui, lazerinio distancijos matuoklio duomenų praktiškai nereikia apdoroti, tačiau šie įrenginiai žymiai brangesni nei panašaus pobūdžio gylio kameros.

2.2.3. Naudojami jutikliai

Jutiklių parinkimas vyksta dviem etapais. Pirmiausiai yra išanalizuojamos aplinkos problemos, kurios įtakoja kuriamą UGV. Pavyzdžiui, rampos atpažinimui neužtektų lazerio tiriančio tik vieną horizontalią liniją erdvėje. Tačiau lauko sąlygomis neveiktų ir gylio kamera, kurios veikimas paremtas infraraudonaisiais spinduliais. Antras etapas yra jutiklių analizės sudarymas pagal kainą bei tinkamumo kriterijų tam tikrai aplinkos problemai spręsti.

2.2.4. Sistemos architektūra

Sistemos architektūros pasirinkimas taip pat įtakoja roboto elgsenos apsprendžiančius algoritmus. Plačiau apie mobilių roboto platformų architektūras žiūrėti sekančiame skyriuje.

2.2.5. Roboto mechaninė dalis

Reikia atkreipti dėmesį į roboto mechaninę dalį. Pavyzdžiui, kiek robotas turės ratų, jų techninė specifikacija. Pagal tai sprendžiamas roboto platformos valdymas, kokia trajektorija ji gali judėti. Kuriamas UGV turės 4 ratus, kurie galės atlikti pilną apsisukimą bei kiekvienas iš jų bus valdomas atskirų variklių. Roboto platforma galės judėti įvairiomis kryptimis. Problemai išspręsti reikia panaudoti algoritmą, kuris filtruotų kliūtis patenkančias į UGV judėjimo trajektoriją.

2.3. Sprendimų apžvalga

1) Jutiklių analizė

Buvo atlikta vaizdo kamerų sugebančių išgauti gylio informaciją analizė. Egzistuoja įvairaus veikimo jutikliai matuojantys vaizdo gylį:

- Struktūrizuotos šviesos kameros
- TOF kameros
- Stereo kameros
- Sferinio vaizdo kameros
- Lazerinės 3D kameros

2.1 lentelėje pateiktos struktūrizuotos šviesos bei TOF kamerų charakteristikos.

2.1 lentelė. Struktūrizuotos šviesos bei TOF kamerų palyginimas

Jutiklio tipas ir pavadinimas	Vaizdo rezoliucija	Vertikalus matymo kampas	Horizontalus matymo kapas	Kadrai per sekundę	Papildoma informacija	Kaina
Struktūrizuotos šviesos kameros						
Kinect for Windows	640x480	43	57	30	4 mikrofonai, 3 ašių akselerommikrofonetras su 1 lapsnio paklaida	220USD
Asus Xtion Pro Live	640x480	45	58	30	2 mikrofonai, naudoja OpenNI biblioteka	170USD
PrimeSense Camine	640x480	45	57,5	30	2 mikrofonai, naudoja OpenNI biblioteka	200USD
PrimeSense Capri	640x480	45	57,5		Mažas, naudoja OpenNI biblioteka	Nenurodyta
Structure sensor	640x480	45	58	30	Kol kas neišleista, naudoja OpenNI biblioteka	Nenurodyta
Fotonic P-Series	640x480	45	57,5	30		1800EUR-725EUR
Time of Flight (TOF) kamera						
Bluetechnix GmbH, Argos3D - P100	160x120	90	90	160	Turi SDK, OpenNI palaikymas	850EUR
PMD[vision] CamBoard nano	160x120	68	90	90	Turi SDK paketą	690USD
Fotonic E-Series	160x120	53	70	57		3800EUR-1000EUR
MESA SR4000	176x144	55	69	30		4279USD
SoftKinetic DS311	160x120	42	57,3	60		299USD
SoftKinetic DS325	320x240	58	74	60		249USD
Odos real.iZ-1K	1280x1024			50	Ethernet	10000EUR

					connector	

2.2 lentelėje pateiktos stereo bei sferinio vaizdo kamerų charakteristikos.

2.2 lentelė. Stereo, sferinio vaizdo bei lazerinių kamerų charakteristikos

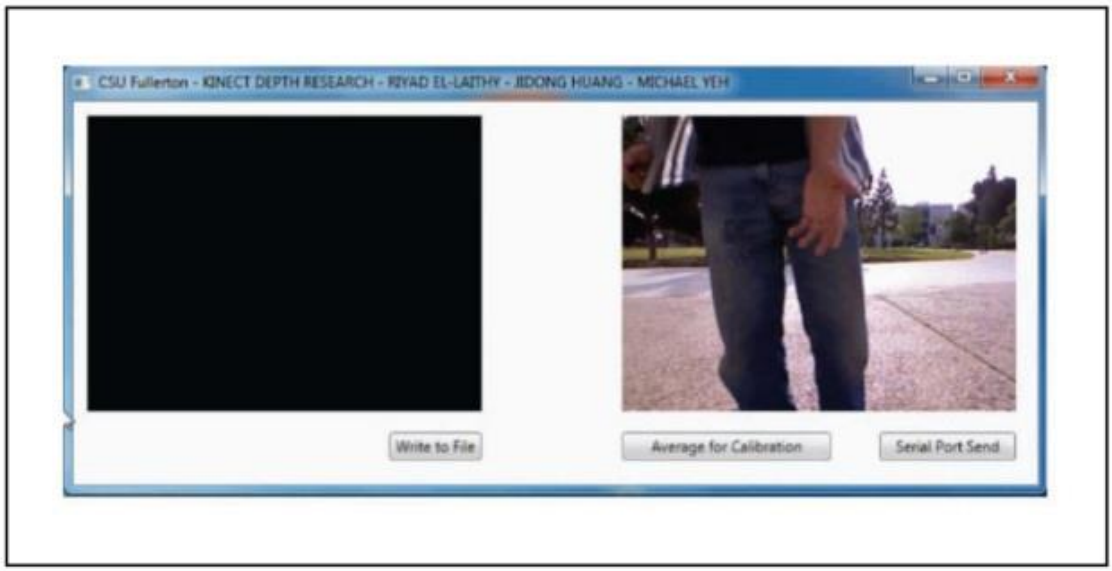
Jutiklio tipas ir pavadinimas	**Vaizdo rezoliucija**	**Kadrai per sekundę**	**Papildoma informacija**	**Kaina**
Stereo kameros				
Vmod Stereo Camera Module	1600x1200	15	Nėra PĮ kūrimo įrankių (SDK)	83USD
MobileRanger C3D Stereo Camera	752x480	30		Nenurodyta
PTGrey Bumblebee2	1024x768	20	Yra SDK skirtas vaizdo apdorojimui	1895-2395USD
PTGrey Bumblebee XB3	1280x960	15	Yra SDK skirtas vaizdo apdorojimui	Nenurodyta
iDS N10 Stereo 3D Camera	752x480	Nenurodyta	Yra SDK skirtas vaizdo apdorojimui	5900EUR
Sferinės vaizdo kameros				
PTGrey Ladybug2	1024x768	30	Matymo kampas: 360 laipsnių	9995USD
PTGrey Ladybug3	1600x1200	15	Matymo kampas: 360 laipsnių	14995USD
PTGrey Ladybug5	2048x2448	10	Matymo kampas: 360 laipsnių	19995USD
Lazerinė 3D kamera				
Sick RANGER-D	512x512	2	Matymo kampas priklauso nuo pasirinkto lęšio	3669EUR

Išanalizavus aukščiau pateiktose lentelėse esančių kamerų charakteristikas buvo nuspręsta panaudoti įmonėje turimą Kinect kamera (struktūrizuotos šviesos kamera), o galutiniam prototipui – Argos3D - P100 (TOF kamera).

Dauguma variantų buvo atmesti dėl pernelyg didelės kainos. Net ir pirkimui išsirinktos Argos3D - P100 kaina yra per didelė, jeigu yra kalbama apie masinę gamybą.

Galutiniam produktui reikėtų atlikti papildomą analizę skirtą išsiaiškinti ar apsimoka pirkti išbaigtus sprendimus.

Kinect kameros charakteristikos pakankamos kokybiškai atlikti kliūčių atpažinimo ir vengimo tyrimą. Palyginus su kitais panašaus tipo prietaisais ši kamera yra labai pigi. Tačiau verta paminėti kelis Kinect įrenginio trūkumus. Kadangi Kinect prietaiso gaunamas vaizdas yra projektuojamas pasitelkiant infraraudonuosius spindulius, iškyla problemų prie tam tikrų aplinkos sąlygų. Pavyzdžiui, Kinect infraraudonieji spinduliai pereina stiklą, kuris tokiu atveju yra neužfiksuojamas gylio vaizde. Kiti veiksniai įtakojantys Kinect gaunamo vaizdo teisingumą yra temperatūra bei tiesioginiai saulės spinduliai. CSUF studentai atliko eksperimentą su Kinect įranga [2]. Buvo paimti du gylio kameros užfiksuoti vaizdai. Vienas jų 3-čią valandą po vidurdienio (žr. 2.1 pav.).



2.1 pav. Gylio kameros vaizdas gautas dienos metu

Sekantis vaizdas gautas vakare (žr. 2.2 pav.).

2.2 pav. Gylio kameros vaizdas užfiksuotas vakare

Iš atlikto eksperimento bei gautų vaizdo rezultatų lauko sąlygomis, įsitikinta, kad dienos šviesoje, kai šviečia saulė, robotui su Kinect įranga nebūtų įmanoma gauti duomenų apie žmogų arba aplinkos objektus. Todėl sekimą ir kliūčių vengimą tektų spręsti pasikliaunant kitais jutikliais, kurie gali veikti minėtomis sąlygomis. Magistro darbe, tyrimai nebuvo atliekami lauke arba esant tiesioginiams saulės spinduliams, tačiau galutiniam prototipui buvo pasirinktas kitas variantas – TOF kamera. Jos veikimo principas panašus į lazerinių 3D kamerų. Žinant šviesos greitį kameros jutikliai matuoja laiką per kurį šviesa pasiekia objektus bei grįžta atgal. Priklausomai kaip ilgai užtruko šviesos impulsas galima nustatyti apytikslį objekto atstumą iki TOF kameros. Taip pat verta paminėti, kad tokio tipo kameros veikia net ir esant tiesioginiams saulės spinduliams su tam tikrais nukrypimais. Triukšmus atsiradusius tokiais atvejais galima koreguoti pasitelkiant korekcinius algoritmus [3].

Kinect kameros vaizdo apdorojimui galima naudoti nemokama Microsoft Kinect SDK, tačiau buvo nuspręsta panaudoti OpenNI biblioteka. Tai yra grindžiama tuo, kad galutiniam prototipui pasirinkta TOF kamera Argos3D - P100. Pakeitus Kinect kamerą, naudojami vaizdo apdorojimo algoritmai išliks tie patys.

2) Aplinkos problemų sprendimai

Analizuojant projekto problemas buvo nustatyta, kad roboto platforma judės ganėtinai užgrūstoje, dinaminėje aplinkoje: žmonės, kitos roboto platformos ar prekių vežimėliai. Taip pat viena iš svarbiausių problemų yra rampų atpažinimas.

Iškyla poreikis atpažinti objektus erdvėje. Tam tikslui reikia gauti objektų taškų pasiskirstymą trimatėje erdvėje. Buvo nuspręsta panaudoti PCL atvirojo kodo biblioteką, kuri skirta darbui su taškais trimatėje erdvėje. Sugeneruotos objektų taškų koordinatės vadinamos taškų debesimi (angl. Point cloud).

Buvo atlikta taškų debesies apdorojimo algoritmų analizė:

- RANSAC (Plane fitting) metodas

Tai metodas skirtas surasti plokštumą pasitelkiant taškų debesies duomenis. Pradžioje yra atsitiktinai parenkami trys taškai erdvėje ir tada paskaičiuojami atitinkamos plokštumos parametrai. Pagal plokštumos parametrus galima fiksuoti taškus, kurie jai priklauso. Tai apsprendžia pasirinktinai nustatytas slenksčio parametras. Taškai, kurie viršija slenksčio parametrą traktuojami kaip pašaliniai. Po to, ši procedūra yra atliekama N kartų. Kiekvieną kartą gaunamas rezultatas yra lyginimas su prieš tai išsaugotu. Jeigu naujas rezultatas yra geresnis, senąjį rezultatą keičiame naujuoju [4].

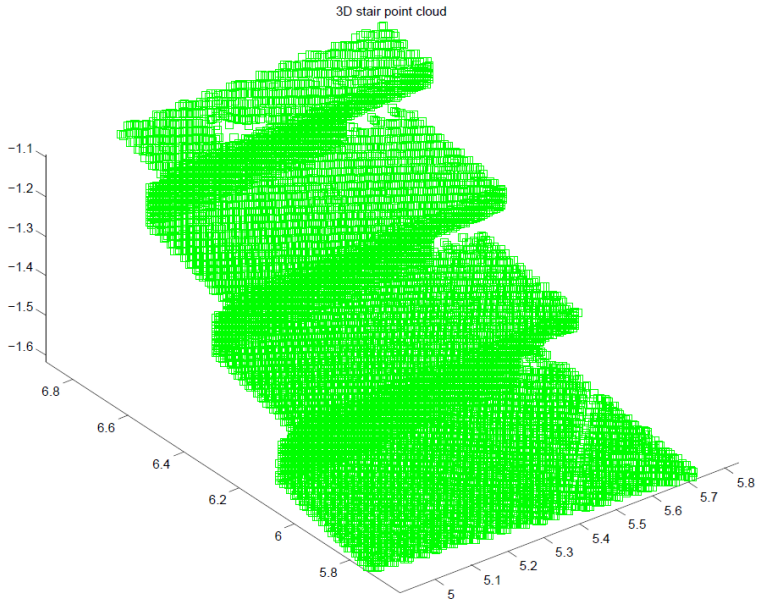
Šiam algoritmui reikia keturių skirtingų įėjimo duomenų:

1) 3D taškų debesies (X, Y, Z koordinačių matrica).
2) Slenksčio parametras *t*. Leistinas taškų nuokrypis nuo pasirinktos plokštumos.

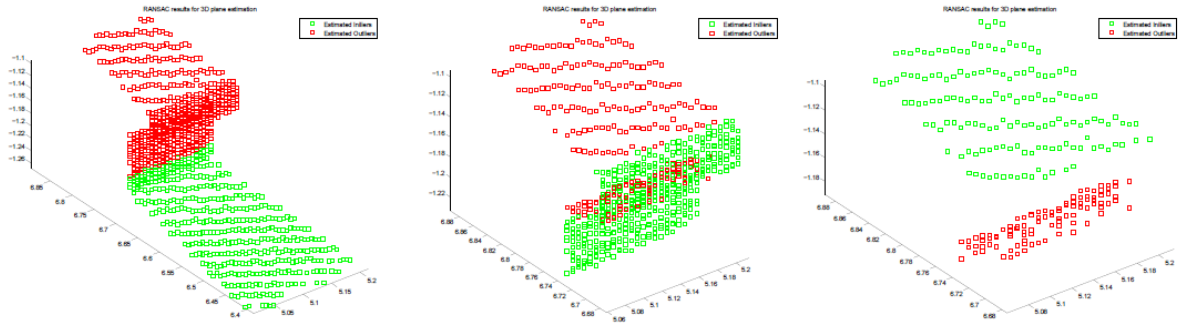
3) Numatomas didžiausias taškų skaičius, kuris gali priklausyti pasirinktai plokštumai. Išvestas iš bendro taškų tankumo bei numatomos plokštumos paviršiaus ploto.

4) Tikimybė α tai minimali tikimybė rasti bent vieną gerą rinkinį iš atliktos paieškos per N kartų. Dažniausiai α reikšmė svyruoja nuo 0,90 ir 0,99.

Pavyzdys su laiptų plokštumų atlikta paieška pateiktas 2.3 bei 2.4 paveiksluose [5].

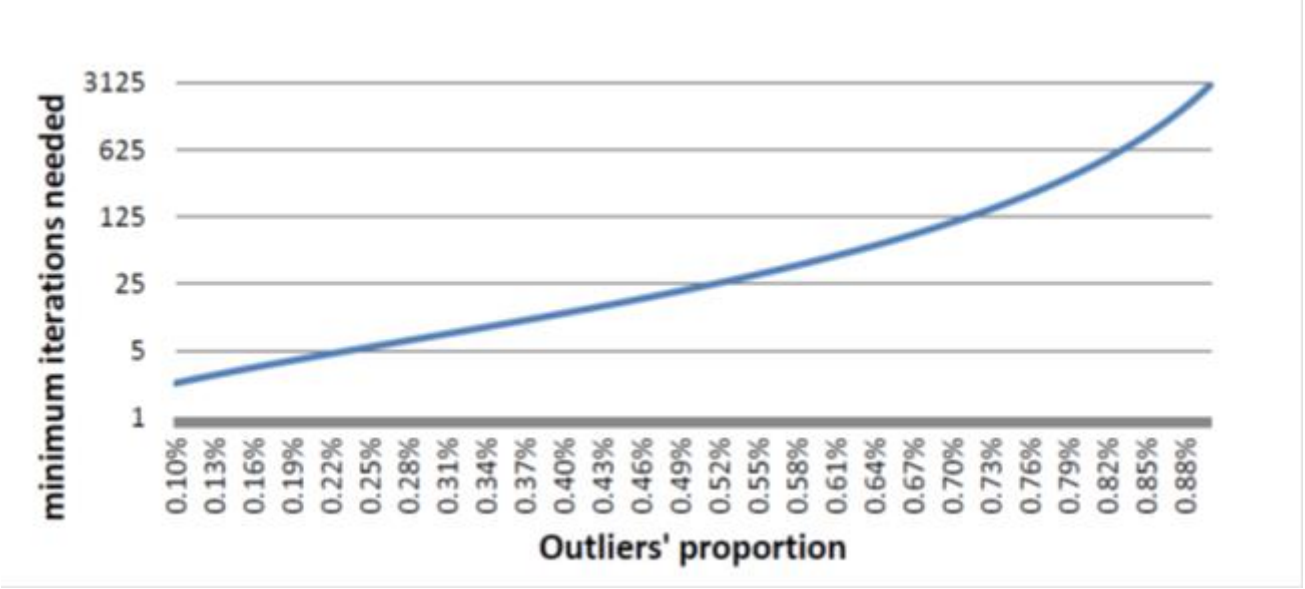


2.3 pav. Laiptų taškų debesies įėjimo duomenys



2.4 pav. Aptiktos skirtingos plokštumos

Kaip matyti, pirmiausiai buvo rasta pirmoji plokštuma (2.4 pav. kairė pusė). Toliau, einant į viršų buvo rastos ir kitos plokštumos. Algoritmas sugeba gan tiksliai surasti plokštumas, tačiau jis yra ganėtinai lėtas bei negali veikti realiu laiku. Taip yra todėl, kad kiekvieno taško skaičiavimas iki plokštumos pareikalauja nemažai kompiuterio resursų. Viename darbe buvo atliktas tyrimas kaip pašalinių taškų dydis įtakojo minimalų iteracijų skaičių užtikrinantį plokštumos radimą. Rezultatas atvaizduotas 2.5 paveiksle [6]. Ordinatėje pateikti minimalių iteracijų dydžiai, abscisėje – pašalinių taškų proporcijos.



2.5 pav. Minimalių iteracijų priklausomybė nuo pašalinių taškų proporcijos

Galima daryti išvada, kad esant dideliam skaičiui plokštumų arba kai tos plokštumos yra nedidelės, RANSAC algoritmas gali būti per lėtas. Taip yra todėl, kad daugelis taškų bus traktuojami kaip pašaliniai, priklausantys kitoms plokštumoms, o tai sąlygoja iteracijų eksponentinį augimą.

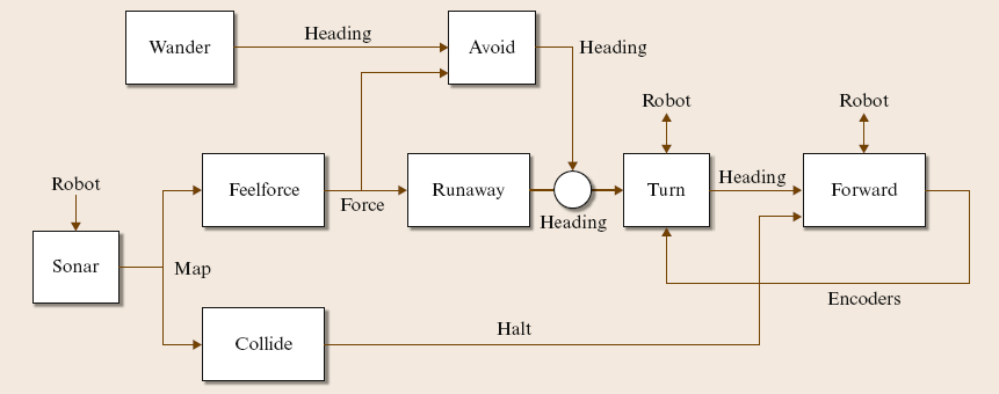
- Region growing metodas

Dalinai išsprendžia RANSAC algoritmo problemą, kai turime daug plokštumų. Metodo veikimo principas: pradedame nuo pradžios regiono arba kitaip taško erdvėje. Tada ieškome artimiausių kaimyninių taškų ir tikriname ar jie atitinka tam tikrus parametrus. Jeigu atitinka, galima priskirti rastus taškus esamai plokštumai. Tačiau šis metodas taip pat remiasi taškų distancijų skaičiavimų į ieškomą plokštumą, todėl nėra gaunami labai tikslūs rezultatai [7].

3) Roboto architektūrų apžvalga

Autonominiam roboto darbui yra reikalingi įvairūs jutikliai. Robotas negaunantis informacijos apie jį supančią aplinką gali atlikti tik tam tikrus veiksmus, kurie yra iš anksto užprogramuoti. Mobiliai autonominei platformai yra privaloma turėti bent kelis skirtingus jutiklius. Vieni jutikliai turėtų aptikti objektus (potencialias kliūtis), kiti komunikuoti su žmogumi ir jį sekti. Vienu metu atsiranda kelios skirtingos roboto elgsenos. Robotas gali judėti paskui objektą, šiam pakeitus kryptį turi būti fiksuojamas pasikeitimas, kas atitinkamai sąlygoja roboto greitį bei kryptį. Jeigu staigiai, priešais robotą užfiksuojama kliūtis, jis privalo į tai sureaguoti, kol neįvyko galimas susidūrimas [8].

Aprašytam roboto valdymui galima panaudoti roboto elgsena grįstą architektūrą. Viena iš elgsena pagrįstų architektūrų pavaizduota 2.6 paveiksle.



2.6 pav. Brooks architektūra

Šiame pavyzdyje veikia dvi programos: atsitiktinis žvalgymasis bei kliūčių vengimas. Robotas atsitiktinai juda po aplinką, kurioje nėra kliūčių, nes programa atsitiktinis žvalgymasis nebus pertraukiama kliūčių vengimo programos. Tačiau ultragarso jutikliui fiksuojant kelyje esančią kliūtį, programa kliūčių vengimas, kurios veikimas yra aukštesnio lygmens, nutrauks atsitiktinį roboto judėjimą.

Brooks architektūros pranašumai:

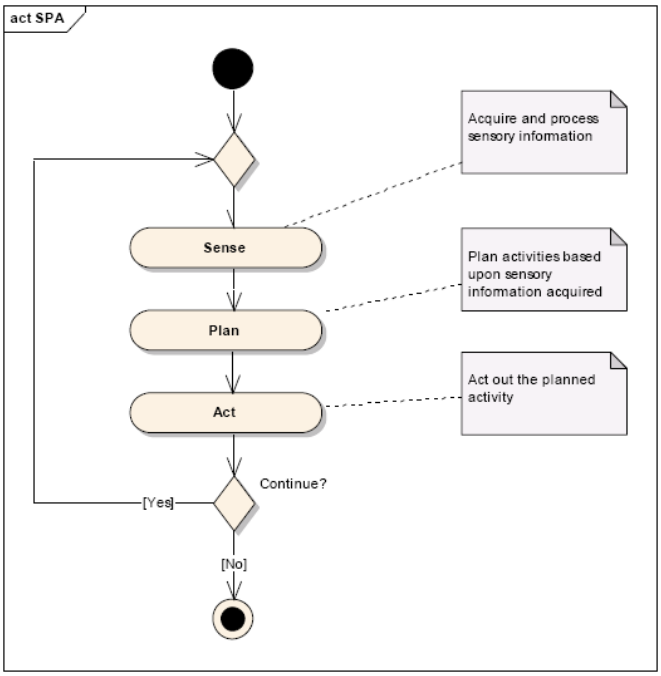
- Greitis ir reaktyvumas
- Tinkamumas dinaminiam pasauliui

Trūkumai:

- Sudėtinga apjungti elgseną siekiant ilgalaikių tikslų;
- Beveik neįmanoma optimizuoti roboto elgseną

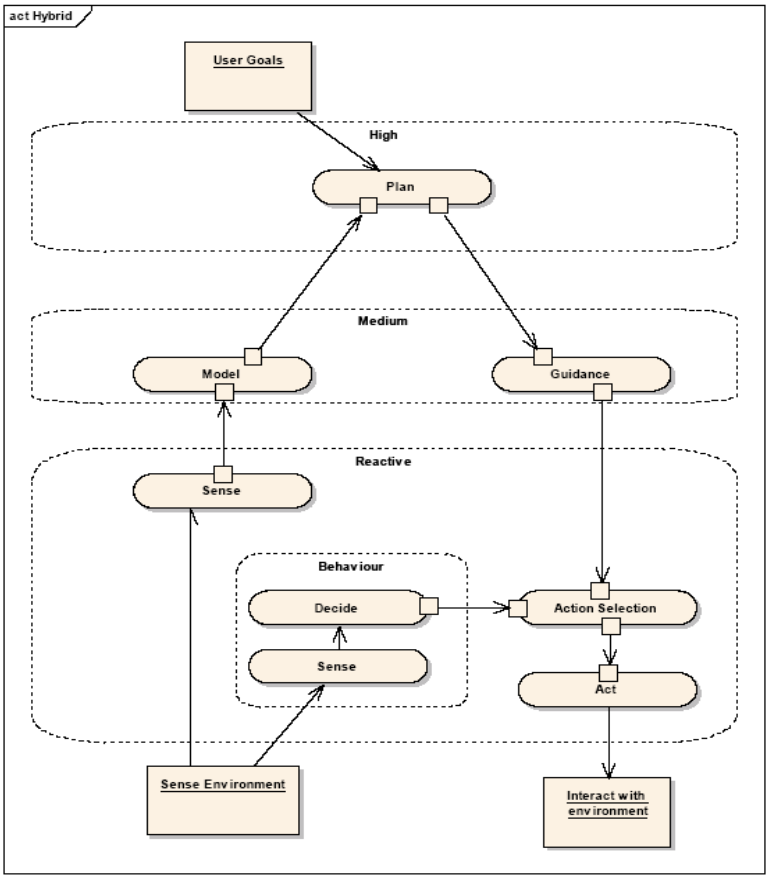
Galima nuspėti, kad robotas tokiu atveju gana greitai sureaguoja į aplinkos pokyčius bei pasirenka programą, kuri atitinka tam tikrus jos iššaukimo kriterijus. Sunkumai iškyla, kai roboto elgsena yra sudėtinga.

Sense-Plan-Act architektūroje (žr. 2.7 pav.) remiantis iš jutiklių gauta informacija suformuojamas roboto veiksmų planas. Planas yra sudarytas iš komandų roboto vykdytuvams sekos. „Act“ seka taip pat gali gauti informaciją tiesiogiai iš vykdytojų. Egzistuoja šios architektūros modifikuota versija, kai veiksmai atliekami lygiagrečiai. Pranašumai: „Act“ veiksmas gali atlikti kelis veiksmus tol, kol „Plan“ veiksmas planuoja sekančius veiksmus.



2.7 pav. Sense, Plan, Act (SPA) architektūra

Hibridinė trijų lygmenų architektūra (žr. 2.8 pav.) organizuota panašiai kaip klasikinė trijų lygmenų architektūra, tačiau žemiausias lygmuo yra pakeistas elgsena grįsta sistema. Jutiklis gali aktyvuoti veiksma elgsena grįstame lygmenyje be viduriniojo arba viršutiniojo lygmenų įtakos. Todėl viduriniame lygmenyje gali būti reikalingas elgsenos menedžeris, kuris yra atsakingas už roboto elgsenos komponentų įjungimą / išjungimą misijos metu. Jutiklio informacija yra naudojama modeliui sukurti, kuris savo ruožtu yra naudojamas veiksmų planavimui. Elgsenos lygmenyje, jutimo veiksmas yra naudojamas jau sukurtam planui aktyvuoti. Pastarasis gali panaikinti valdymo veiksmo išėjimus, pvz., dėl roboto saugumo [9].



2.8 pav. Hibridinė trijų lygmenų architektūra

2.4. Analizės išvados

Buvo išanalizuotos vaizdo kameros bei jutikliai, kurie skirti išgauti informacija apie aplinkos objektus. Nuo pasirinktų jutiklių priklauso objektų apdorojimo algoritmai. Atlikta analizė parodė, kad Kinect kamera yra tinkamas sprendimas pirmajam prototipui atsižvelgiant į kainos ir kokybės santykį. Galutiniam prototipui buvo pasirinkta TOF kamera Argos3D - P100, kuri gali veikti ir lauko sąlygomis. Šiom kamerom galima panaudoti tas pačias atvirojo kodo vaizdų apdorojimo bibliotekas (OpenCV, OpenNI, PCL).

Grindų, nuožulnių plokštumų bei slenksčių radimui buvo ištirti aplinkos plokštumų paieškos algoritmai. Deja, visi algoritmai nėra pakankamai greiti, kad galėtų veikti realiu laiku.

Iš nagrinėtų roboto sistemos architektūrų, hibridinė architektūra pasirodė priimtiniausia. Taip yra todėl, kad naudojant šią architektūra galima žemesniame lygmenyje esantį komponentą traktuoti kaip elgseną. Sukurtos sistemos atveju – kliūčių vengimo servisas atlieka minėtąjį vaidmenį.

3. PROJEKTINĖ DALIS

3.1. Sistemos paskirtis

Autonominės mobilios platformos sistema geba surinkti informaciją iš įvairių roboto naudojamų jutiklių, analizuoja bei apdoroja surinktus duomenis, pagal poreikį vykdo roboto elgsenas pasiųsdamas reikiamą komandą į roboto aktuatorius.

Sistema turi grafinę vartotojo sąsają, kuri leidžia eksperimentuoti su atskirais roboto valdymo moduliais. Vartotojas gali stebėti roboto atliekamus veiksmus, realiu laiku peržiūrėti gaunamus duomenis iš jutiklių. Taip pat duomenys gali būti įrašomi, o vėliau ir atkartojami. Atkartojimo modulis atlieka jutiklių komponentų imitavimą skaitydamas sukauptus duomenis iš failo ir juos siųsdamas į jutiklių apdorojimo modulius. Šios sistemos funkcijos suteikia vartotojui galimybę analizuoti bei patikrinti ar robotas teisingai interpretuoja gautus duomenis. Tokiu būdu galima atrasti nenumatytas klaidas arba efektyviai suderinti roboto veikimą, keičiant jo konfigūracijos parametrų reikšmes.

3.2. Probleminė sritis

Viena iš problemų, kurią reikia spręsti yra sistemos suderinamumas su roboto platforma. Pakeitus vieną ar kelias roboto dalis tenka modifikuoti įvairias sistemos vietas.

Kita problema yra roboto elgsenų prioritetų suskirstymas. Dažniausiai kliūčių vengimo funkcija yra viena iš roboto judesius koordinuojančių modulių sudedamųjų dalių. Pavyzdžiui, kelio planavimo komponentas gauna sudarytą žemėlapį su kliūtimis bei laisva erdve. Pritaikomas optimalaus kelio radimo algoritmas, kuris paskaičiuoja kelią iki reikiamo tikslo. Tačiau žemėlapio sudarymas bei kelio radimas naudoja nemažai resursų. Planavimo metu robotas tęsia judėjimą, tad esant vėlinimui bei atsiradus staigiai kliūčiai gali būti nespėta pakeisti roboto trajektorijos.

Šių problemų sprendimai yra aprašomi sekančiuose skyreliuose pagrinde atkreipiant dėmesį į architektūros parinkimą, naudojamus kūrimo įrankius bei technologijas.

3.3. Projekto reikalavimai ir apribojimai

3.3.1. Funkciniai reikalavimai

Pagrindinės sistemos funkcijos:

- vaizdo kameros atvaizdavimas ekrane: gylio vaizdas, žmogaus sekimo vaizdas;
- roboto pozicijos bei susijusių jutiklių duomenų rodymas ekrane (realiu laiku);
- roboto pozicijos rankinis nustatymas;
- roboto platformos motorų rankinis valdymas;
- distanciją matuojančių jutiklių reikšmių atvaizdavimas;
- grafinis roboto atvaizdavimas 2-matėje erdvėje;
- tikslo taško pažymėjimas 2-matėje erdvėje (inicijuoja roboto judesių vykdymo komponento darbą);

- kelių taškų pažymėjimas 2-matėje erdvėje (inicijuoja roboto judesių planavimo komponento darbą);
- roboto aptiktų objektų atvaizdavimo ekrane funkcija, kuri leidžia pasirinkti skirtingų tipų objektus: aptikti visi objektai, apjungti objektai, objektai klasifikuojami kaip kliūtys, objektai klasifikuojami kaip potencialios kliūtys bei pasirinktame aukštyje fiksuojami objektai;
- sudaromo užimtumo žemėlapio atvaizdavimas ekrane;
- distancijos iki objektų bei roboto greičio grafikų sudarymas bei atvaizdavimas realiu laiku;
- roboto automatinis greičio reguliavimas aptikus kliūtis.

3.3.2. Nefunkciniai reikalavimai

Esminiai sistemai keliami nefunkciniai reikalavimai:

- duomenų apdorojimas bei aktuatorių valdymas turi veikti realiu laiku;
- sistema turi veikti Windows 7 arba Windows Embedded Standart 7 operacinėse sistemose;
- robotas turi pervažiuoti per 5 cm aukščio slenksčius;
- maksimalus roboto greitis ne daugiau kaip 10 m/s;
- turi funkcionuoti patalpoje bei lauke;
- pilnas manevringumas: apsisukimas vietoje, judėjimas į priekį, atgal bei į šalis.

3.4. Sistemos kūrimo įrankiai

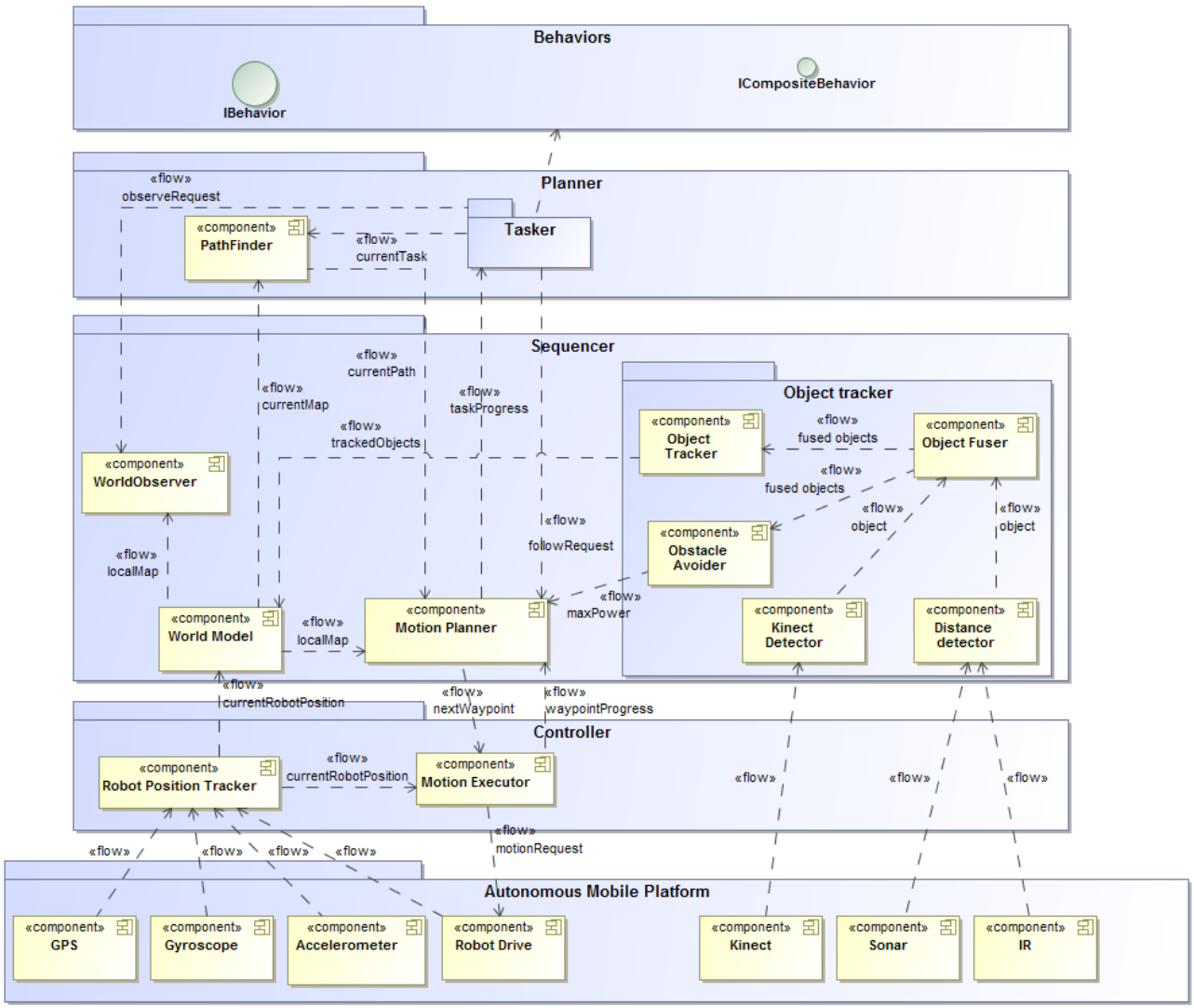
Įvertinus vieną iš nefunkcinių reikalavimų, kad sistema turi veikti Windows operacinėje sistemoje buvo atlikta PĮ kūrimo įrankių analizė. Buvo atrastas MRDS 4 (Microsoft Robotics Developer Studio) programų kūrimo rinkinys. Šį rinkinį sudaro:

- CCR (Concurrency and Coordination Runtime) – biblioteka skirta lygiagrečių procesų valdymui;
- DSS (Decentralized Software Services) – paslaugomis grįsta architektūra leidžianti kurti ir koordinuoti paskirstytas programas;
- VPL (Visual Programming Language) – grafinio programavimo aplinka leidžia greitai suprojektuoti programą ir taip patikrinti sukurtų komponentų veikimą;
- VSE (Visual Simulation Environment) – 3-matės erdvės simuliatorius, kuriame galioja visi fizikos dėsniai. Kuriamų robotų prototipus ir jų algoritmus galima išbandyti tiek patalpoje, tiek natūralioje lauko aplinkoje.

Pasirinktos programavimo kalbos – C# bei C++, kuris naudojamas tik vaizdo apdorojimui. Grafinei vartotojo sąsajai panaudota WPF (Windows Presentation Foundation) grafinė posistemė. Taip pat panaudotas .NET 4.0 karkasas.

3.5. Sistemos architektūra

Pasirinktas MRDS paketas leido apsispręsti dėl naudojamos sistemos architektūros. Sistemoje komponentai arba kitaip servisai bendrauja tarpusavyje apsikeisdami žinutėmis. Sukurtos sistemos architektūros modelis pateiktas 3.1 paveiksle. Tai sluoksninė architektūra, kurios kiekvienas sluoksnis atvaizduoja skirtingą abstrakcijos lygį.



3.1 pav. Sistemos architektūra

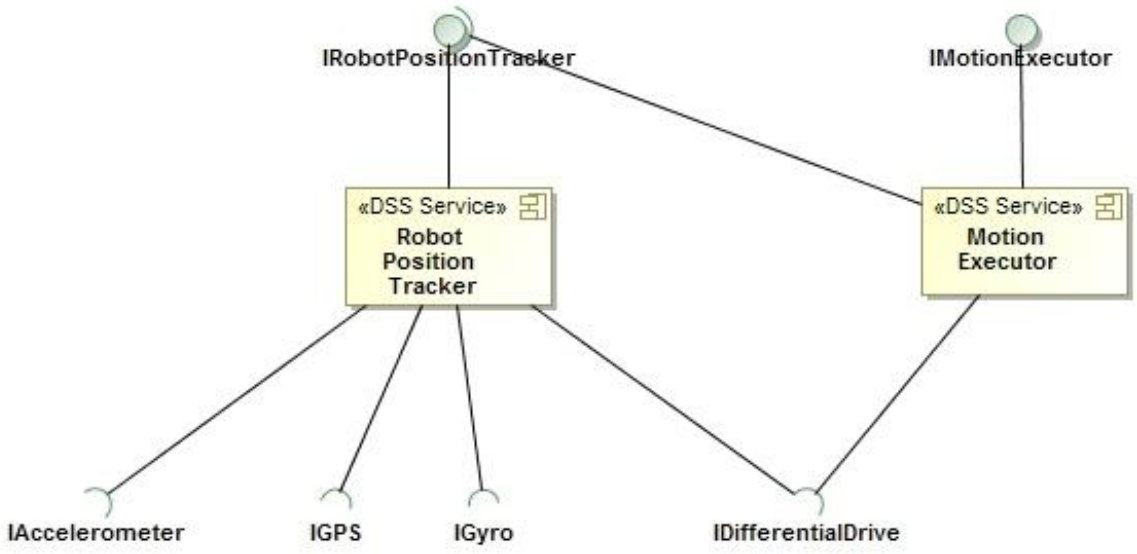
Sistemos architektūros sluoksnių aprašymas pateiktas atskiruose punktuose:

3.5.1. Roboto platformos jutiklių ir aktuatorių sluoksnis

Šiame sluoksnyje yra jutiklių bei aktuatorių servisai, kurie tiesiogiai bendrauja su roboto plokštėje esančia mikroprograma arba atskirais jutikliais. Šie komponentai yra konkretūs. Pavyzdžiui, jeigu būtų atliktas Kinect prietaiso pakeitimas kitu gamintojo įrenginiu, tektų apsirašyti atitinkamą servisą, kuris sugebėtų išgauti reikiamą informaciją iš naujojo jutiklio.

3.5.2. Valdymo sluoksnis

Tai aukštesnio abstrakcijos lygio, reaktyvių servisų sluoksnis. Jame apskaičiuojama roboto pozicija, kuri gali būti pateikta tiek 2-matėje, tiek 3-matėje erdvėje. Tai priklauso nuo naudojamų jutiklių duomenų. Kitas komponentas – judesių vykdytojas. Šis servisas prisijungia prie bendro roboto pozicijos sekimo serviso, kurį gali realizuoti skirtingų tipų roboto pozicijos skaičiavimo servisai. Iš viršesnio vykdymo sluoksnio gautų tikslo koordinačių, judesių vykdytojo servisas pagal naujausią roboto pozicijos reikšmę gali pasiskaičiuoti roboto motorų galias, kurias persiunčia į ratų valdymo servisą. Valdymo sluoksnio komponentų diagrama pateikta 3.2 paveiksle.

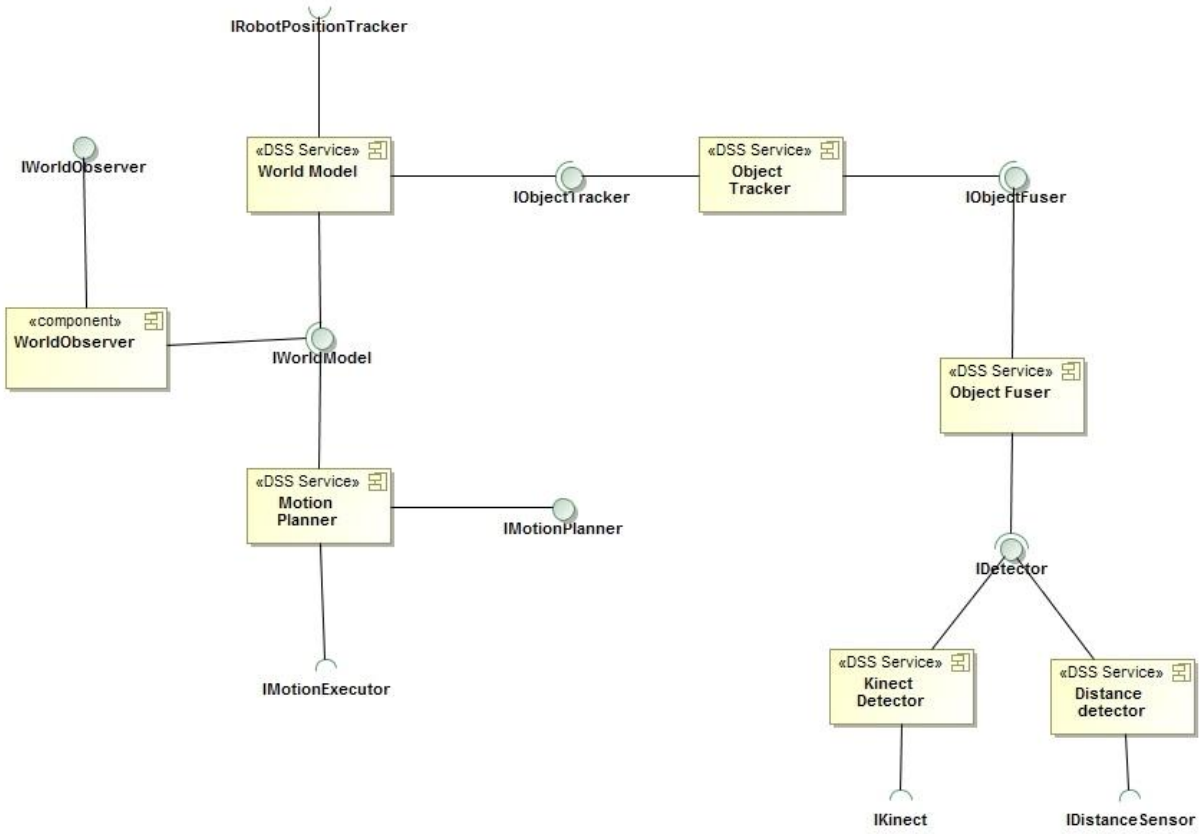


3.2 pav. Valdymo sluoksnio komponentų diagrama

3.5.3. Vykdytojo sluoksnis

Tai lygmuo, kuriame vyksta duomenų apdorojimas. Iš distanciją matuojančių jutiklių yra surenkami duomenys. Iš informacijos apie gautą distanciją yra kuriami objektų sąrašai. Objektai turi taško koordinates, kurios yra reliatyvios roboto koordinatėms.

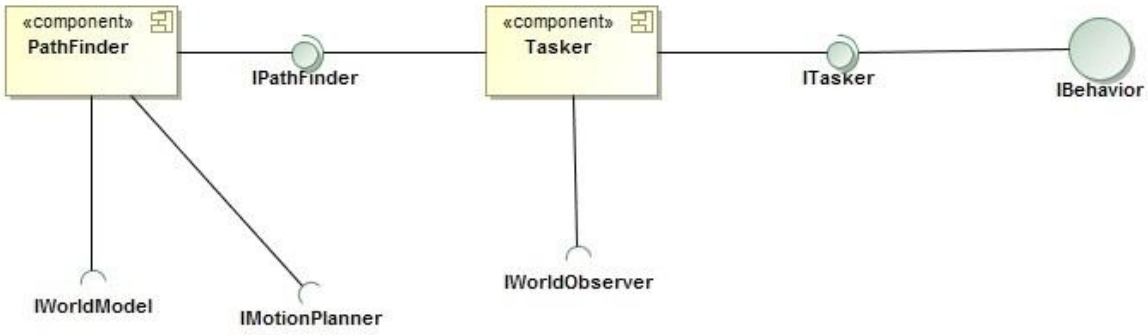
Šių atskirų objektų grupes į bendrą objektų masyvą sujungia objektų „apjungėjo“ servisas. Sekantis servisas stebi gaunamus objektus. Jeigu aptiktų objektų pozicija tam tikrame laiko periode išlieka panaši, galima traktuoti, kad aptiktas objektas yra statinis (jį reikia įrašyti į statinių objektų žemėlapį). Sudarytas žemėlapis yra naudojamas aukšto lygio servisų, kurie atlieka tam tikras užduotis, pavyzdžiui, architektūroje pateiktas kelio radimo (PathFinder) servisas. Gautas kelio taškų masyvas yra siunčiamas į judesių planavimo komponentą, kuris savo ruožtu vykdo kiekvieno taško siuntimą į roboto judesių vykdytojo servisą. Vykdytojo sluoksnio komponentų diagrama pateikta 3.3 paveiksle.



3.3 pav. Vykdymo sluoksnio komponentų diagrama

3.5.4. Planavimo sluoksnis

Aukščiausio lygio servisai, kurie reprezentuoja roboto elgsenas yra planavimo lygmens dalis. Šių servisų veikimą koordinuoja užduočių vykdytojas (Tasker). Planavimo bei elgsenų sluoksnio komponentų diagrama pateikta 3.4 paveiksle.



3.4 pav. Planavimo sluoksnio komponentų diagrama

Užduočių vykdytojas koordinuoja roboto elgsenų veiksmus. Pavyzdžiui, užtikrina, kad skirtingos elgsenos atliekančios tą patį veiksmą negalėtų veikti tuo pačiu metu.

Panaudota architektūra išsprendžia sistemos suderinamumą su skirtingomis platformomis. Aukšto lygio servisai bei jų naudojami algoritmai nepriklauso nuo techninės įrangos. Atliekant roboto platformos techninių komponentų keitimą tereikia apsirašyti servisą, kuris atlieka tvarkyklės (angl. Driver) vaidmenį.

Skyriaus pradžioje buvo minėta, kad elgsenos tam tikrais atvejais gali užtrukti su skaičiavimais. Atsirado būtinybė kliūčių vengimą traktuoti kaip servisą, kuris nepriklauso elgsenų lygmeniui. Kliūčių vengimą nuspręsta panaudoti vykdytojo lygmenyje, kuris veiktų nepriklausomai nuo vykdomų elgsenų. Tokiu atveju, kliūčių vengimo servisas gautų duomenis iš objektų apjungimo komponento.

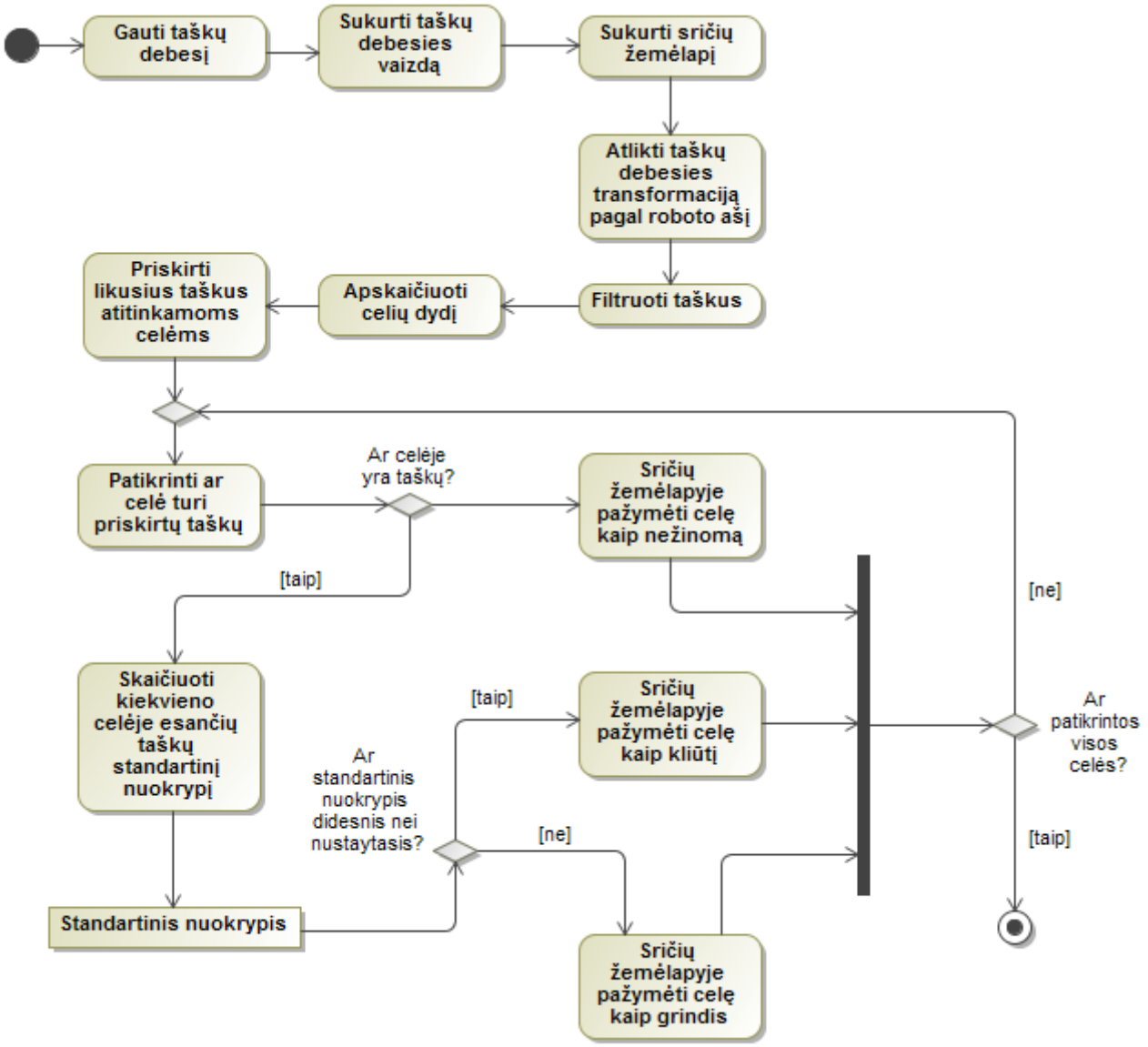
4. TYRIMO DALIS

Šiame skyriuje aprašomi realizuoti objektų apdorojimo bei kliūčių vengimo algoritmai.

4.1. Grindų atpažinimo algoritmas

Analitinėje dalyje nagrinėti plokštumų atpažinimo algoritmai yra per lėti, kad būtų galima efektyviai panaudoti potencialių kliūčių išskyrimui. Buvo sugalvotas problemos sprendimas nenaudojant plokštumų atpažinimo algoritmų. Kol kas nėra būtinybės atpažinti rampų bei slenksčių kaip atskirų objektų. Pastarieji objektai gali būti traktuojami kaip grindys. Sričių žemėlapio sudarymo algoritmo veiklos diagrama pateikta 4.1 paveiksle.

Algoritmas yra skirtas sričių žemėlapio sudarymui. Pagal nustatytus konfigūracijos parametrus yra sukuriamas tinklelio tipo žemėlapis. Paskaičiuojamas vienos žemėlapio celės dydis. Būtina atlikti gauto taškų debesies transformaciją. Pagal kameros poziciją konfigūracijos faile įrašoma transformacijos matrica. Toliau atliekamas nereikalingų taškų filtravimas pagal nustatytąsias maksimalias X, Y bei Z reikšmes. Taškai, kurie yra už nustatytų maksimalių reikšmių – pašalinami. Likę taškai pagal realaus pasaulio koordinates yra priskiriami atitinkamoms celėms. Toliau vykdoma sričių žemėlapio pildymo dalis. Patikrinama kiekviena celė. Jeigu celėje nėra taškų, sričių žemėlapyje pažymima reikšmė kaip nežinoma. Celėse, kuriose yra taškų reikia paskaičiuoti jų standartinį nuokrypį. Paskaičiuotas vienos celės standartinis nuokrypis viršijęs konfigūracijos faile nustatytą maksimalią reikšmę identifikuoja rastą kliūtį. Kitu atveju laikoma, kad celėje esantys taškai priklauso grindų plokštumai. Galiausiai reikia išskirti artimiausių aptiktų objektų masyvą. Tai atliekama skaitant sukurto sričių žemėlapio kiekvienos eilutės reikšmes judant iš žemiausio stulpelio aukštyn (didėja distancijos reikšmė). Aptikus sritį, kuri pažymėta kaip kliūtis, į kliūčių masyvą įdėdama nauja rasta reikšmė bei pereinama į kitą sričių žemėlapio eilutę.



4.1 pav. Sričių žemėlapio sudarymo veiklos diagrama

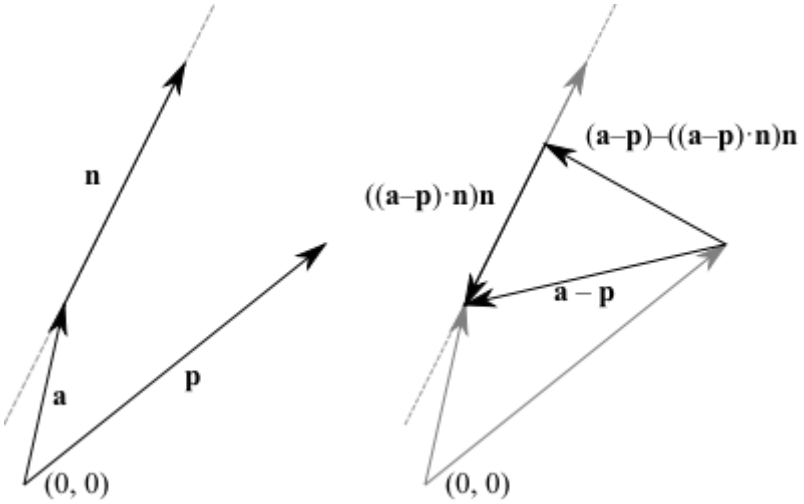
4.2. Objektų filtravimas pagal roboto judėjimo trajektoriją

Objektų filtravimui pagal roboto judėjimo trajektoriją buvo pasirinkta formulė (1), kuri skirta apskaičiuoti 2-matėje erdvėje esančio taško atstumą iki linijos.

$$\text{atstumas}(\mathbf{x} = \mathbf{a} + t\mathbf{n}, \mathbf{p}) = \|(\mathbf{a} - \mathbf{p}) - ((\mathbf{a} - \mathbf{p}) \cdot \mathbf{n})\mathbf{n}\| \quad ; \qquad (1)$$

čia *a* - taškas ant linijos, *t* - skaliaras, *n* – vienetinis vektorius, kurio kryptis sutampa su linijos kryptimi (roboto judėjimo trajektorija), *p* – taškas erdvėje (kliūtis).

Vektoriaus formuluotės iliustracija pateikta 4.2 paveiksle.

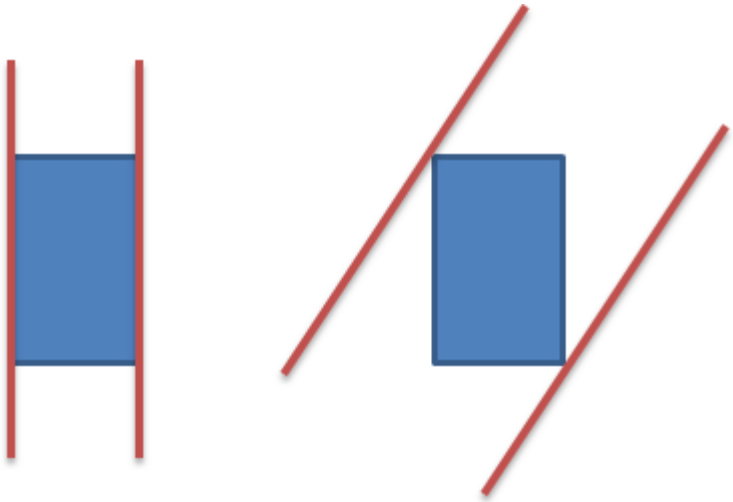


4.2 pav. Vektoriaus formuluotės iliustracija

Iš visų kitų formulių, kurios leidžia gauti taško atstumą iki linijos, vektorinė formuluotė tiko dėl to, kad robotas gali judėti visomis kryptimis. Pavyzdžiui, skaičiuojant Dekarto koordinačių sistemoje pagal pateiktą formulę (2) atsiranda neapibrėžtumas – dalyba iš nulio, kai roboto judėjimo kryptis yra horizontali arba vertikali.

$$\text{atstumas}(ax + by + c = 0, (x_0, y_0)) = \frac{|ax_0 + by_0 + c|}{\sqrt{a^2 + b^2}} \qquad (2)$$

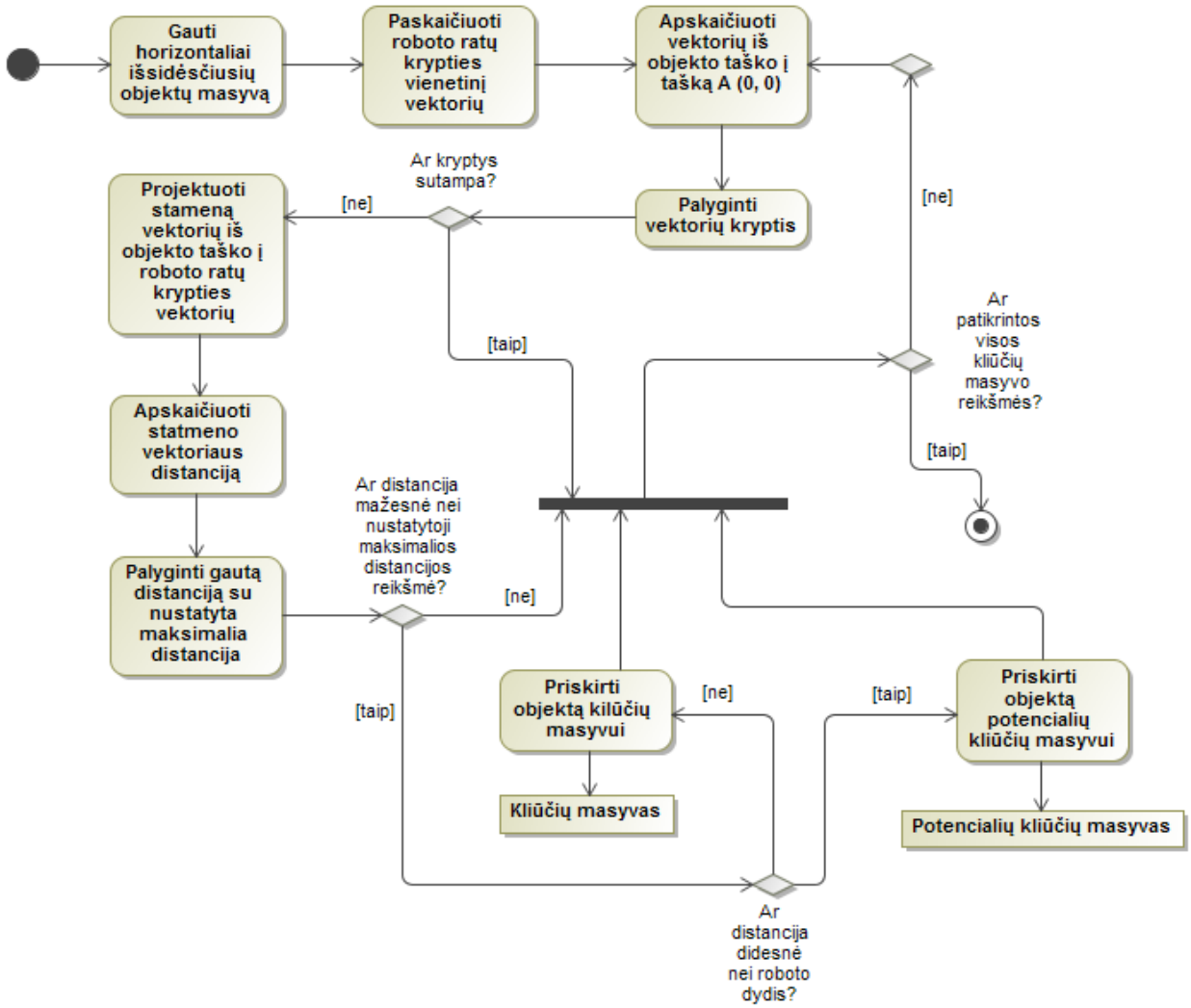
Turint roboto figūrą apibūdinančius taškus bei judėjimo kryptį galima apskaičiuoti spektro dydį, į kurį patekę objektai yra laikomi kliūtimis (žr. 4.3 pav.). Stačiakampio formos robotas, kurio ilgis yra didesnis nei plotis, judėdamas tiesiai užims mažiau erdvės už šonu judantį tokio pat dydžio bei formos robotą. Skaičiavimams pasitelkiama vektoriaus formuluotė (1).



4.3 pav. Erdvės užimtumas atsižvelgus į roboto dydį bei judėjimo kryptį

Objektų filtravimo pagal roboto trajektorija veiklos diagrama pateikta 4.4 paveiksle. Gautas horizontaliai išsidėsčiusių objektų masyvas yra apdorojamas. Kiekvienai masyvo reikšmei pritaikome tą pačią, distancijos iki linijos radimo formulę (1). Taip pat reikia patikrinti roboto krypties vektoriaus bei taško vietą. Pavyzdžiui, robotui judant atgal, priekyje esantys objektai nebus

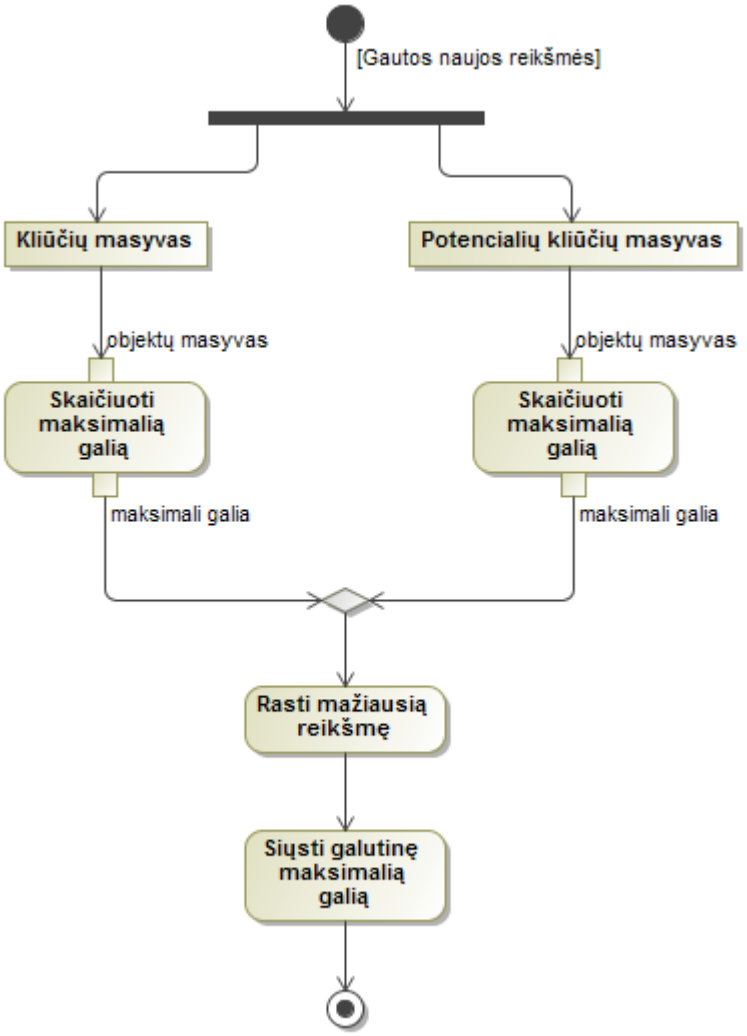
įtraukti į kliūčių sąrašą. Objektas, kuris patenka į apskaičiuotą trajektorijos erdvę yra priskiriamas kliūčių masyvui. Tačiau yra ir papildomas dydis, kuris yra pridedamas prie erdvės reikšmės. Objektai, kurie viršija pradinės erdvės dydį, bet patenka į papildomą plotą yra priskiriami potencialių kliūčių masyvui.



4.4 pav. Objektų filtravimo pagal roboto judėjimo trajektorija veiklos diagrama

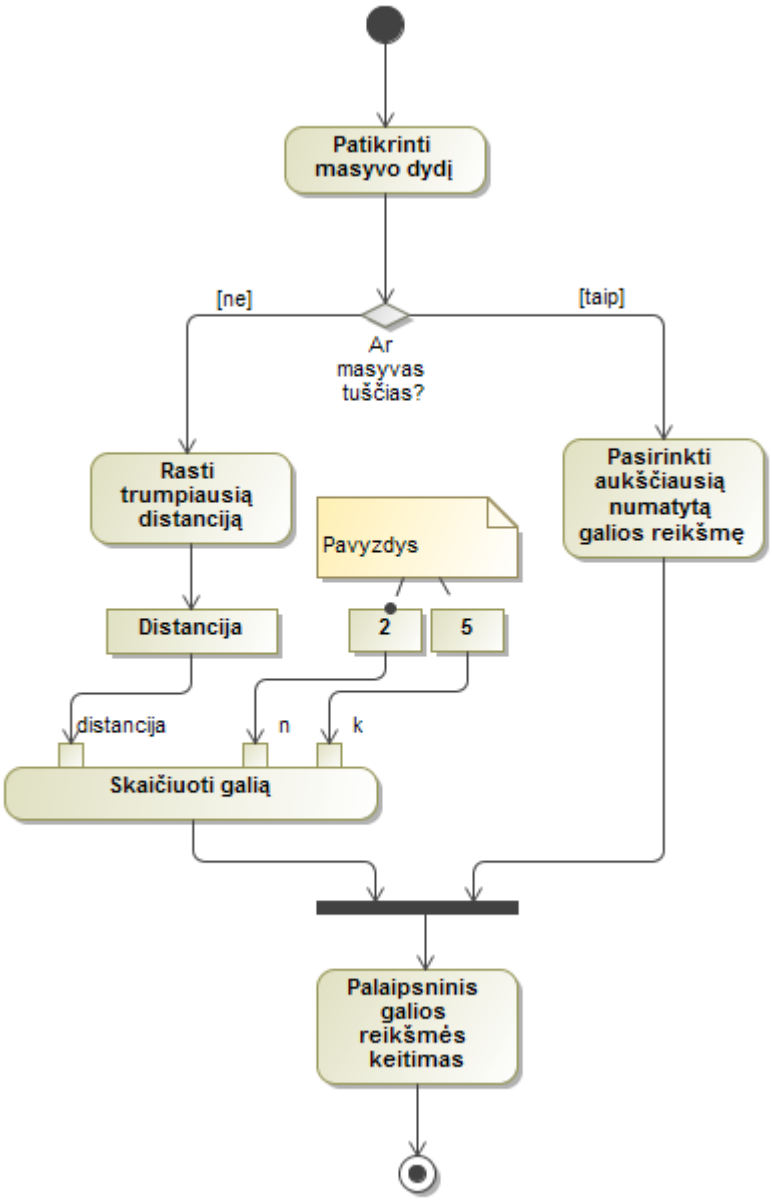
4.3. Kliūčių vengimo algoritmas

Metodo veiklos diagrama pateikta 4.5 paveiksle. Paskaičiuojamas kliūčių bei potencialių kliūčių masyvų maksimalios galios reikšmės. Iš dviejų gautų reikšmių parenkama mažesnioji, kuri siunčiama į roboto judesių planavimo komponentą.



4.5 pav. Kliūčių vengimo veiklos diagrama

Maksimalių galios reikšmių skaičiavimo veiklos diagrama pavaizduota 4.6 paveiksle. Priklausomai nuo paduoto masyvo tipo, maksimalios galios reikšmės bus skirtingos esant tam pačiam atstumui. Skaičiavimai priklauso nuo objekto distancijos iki roboto bei paduodamų kintamųjų parametrų *n* bei *k*. Kintamieji įtakoja kaip jautriai į atstumą iki objektų turėtų reaguoti robotas. Fiksuojant labai artimą kliūtį, robotas turi sustoti. Tuo tarpu esant potencialiai kliūčiai, robotas turėtų sumažinti greitį, bet nevykdyti pilno sustojimo. Potencialių kliūčių užmanymas pagrįstas tuo, kad judėdamas aplinkoje su dinaminiais objektais robotas yra pasiruošęs galimam susidūrimui.



4.6 pav. Galios skaičiavimo veiklos diagrama

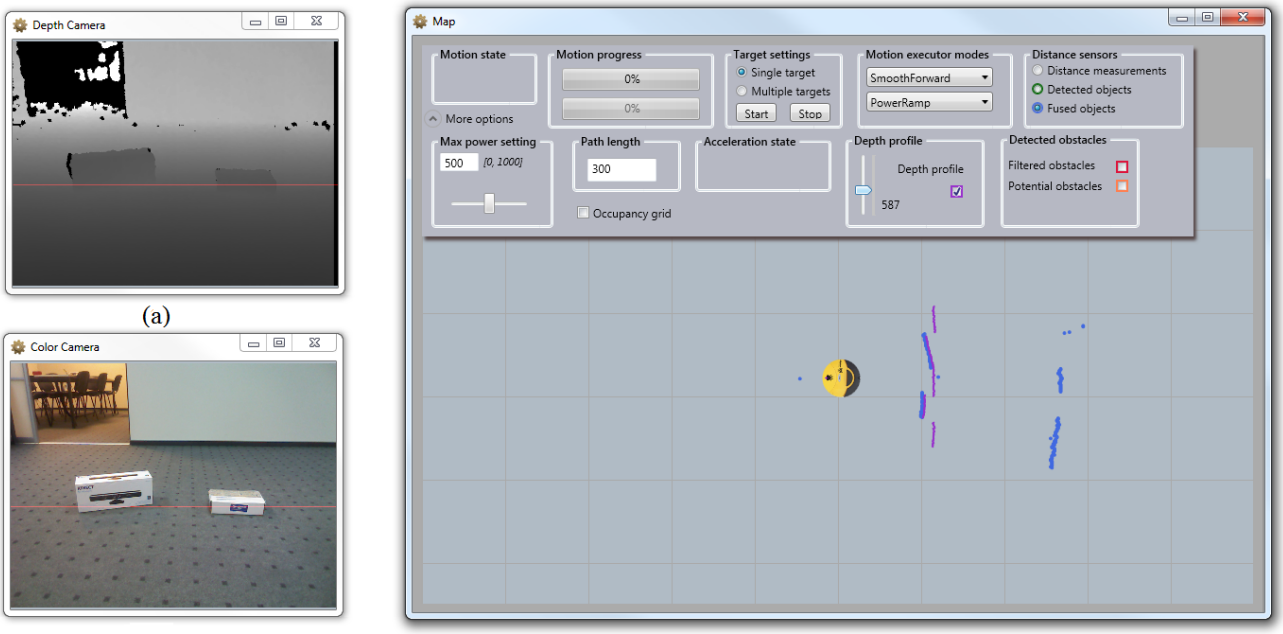
5. EKSPERIMENTINĖ DALIS

5.1. Eksperimentinių tyrimų aprašymas

Tyrimų dalyje aprašytų bei realizuotų metodų eksperimentai pateikiami sekančiuose skyriuose. Pirmasis eksperimentas parodo kliūčių bei slenksčių atskyrimo teisingumą. Eksperimentinių tyrimų tikslas – parodyti kaip metodai atlieka savo funkciją. Pirmieji du eksperimentai susiję su aplinkos klasifikavimu į grindis bei kliūtis. Sekantis eksperimentas – horizontalių objektų filtravimas. Parodomos grafiškai atvaizduotos kliūtys pagal roboto trajektoriją. Galiausiai atliekamas eksperimentas parodantis kaip atpažintos kliūtys įtakoja roboto greitį. Visi eksperimentai buvo atvaizduoti panaudojant sistemos grafinę vartotojo sąsają.

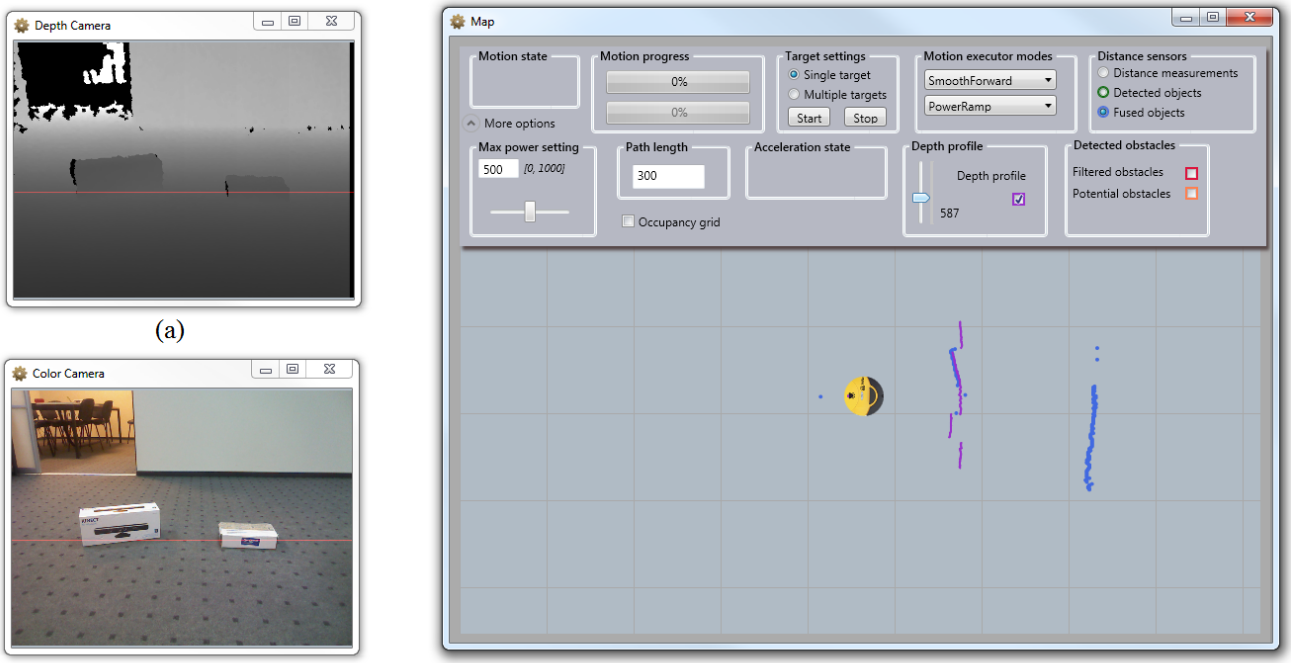
5.2. Kliūties aptikimas keičiant standartinio nuokrypio maksimalią reikšmę

Priešais robotą buvo padėtos dvi skirtingų aukščių dėžės (žr. 5.1 pav.). Didesniosios dėžės aukštis – 18 cm, mažesniosios – 7 cm. Mėlyni taškai vaizduoja aptiktas kliūtis, violetiniai – pasirinktame aukštyje gauti objektų taškai. Su pasirinkta standartinio nuokrypio reikšme – 0,008, abiejų objektų taškai klasifikuojami kaip kliūtys.



5.1 pav. (a) gylio vaizdas, (b) RGB kameros vaizdas, (c) žemėlapio vaizdas, kai maksimali standartinio nuokrypio reikšmė – 0,008

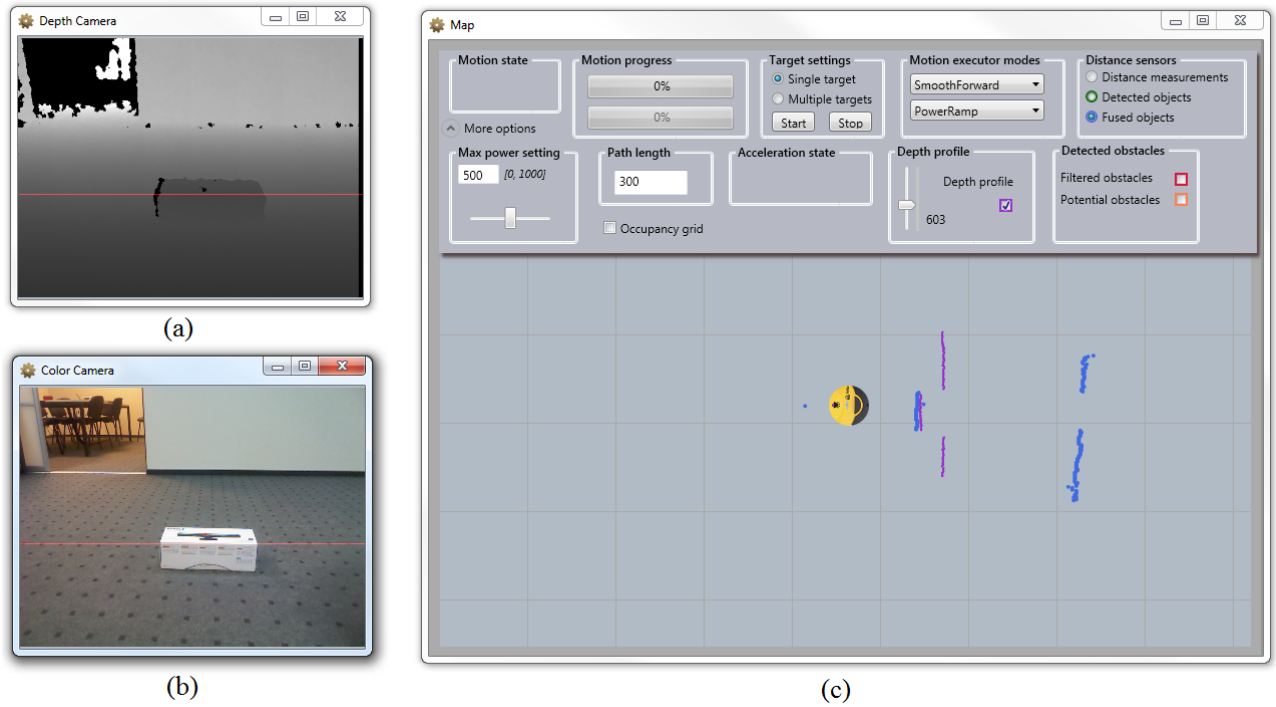
5.2 pateiktame paveiksle stebimi tie patys objektai – dviejų skirtingų dydžių dėžės. Standartinio nuokrypio reikšmė padidinta iki 0,036. Tai sąlygoja mažesnės dėžės taškų pašalinimą iš kliūčių masyvo. Objektų filtravimas pagal taškų standartinio nuokrypio dydį leidžia nekreipti dėmesio į smulkias kliūtis, kurias robotas gali pervažiuoti. Galime teigti, kad algoritmas veikia teisingai. Tai patvirtina pasirinkto aukščio gauti objektų taškai, kurie rodo, kad dėžės taškai buvo aptikti.



5.2 pav. (a) gylio vaizdas, (b) RGB kameros vaizdas, (c) žemėlapio vaizdas, kai maksimali standartinio nuokrypio reikšmė – 0,036

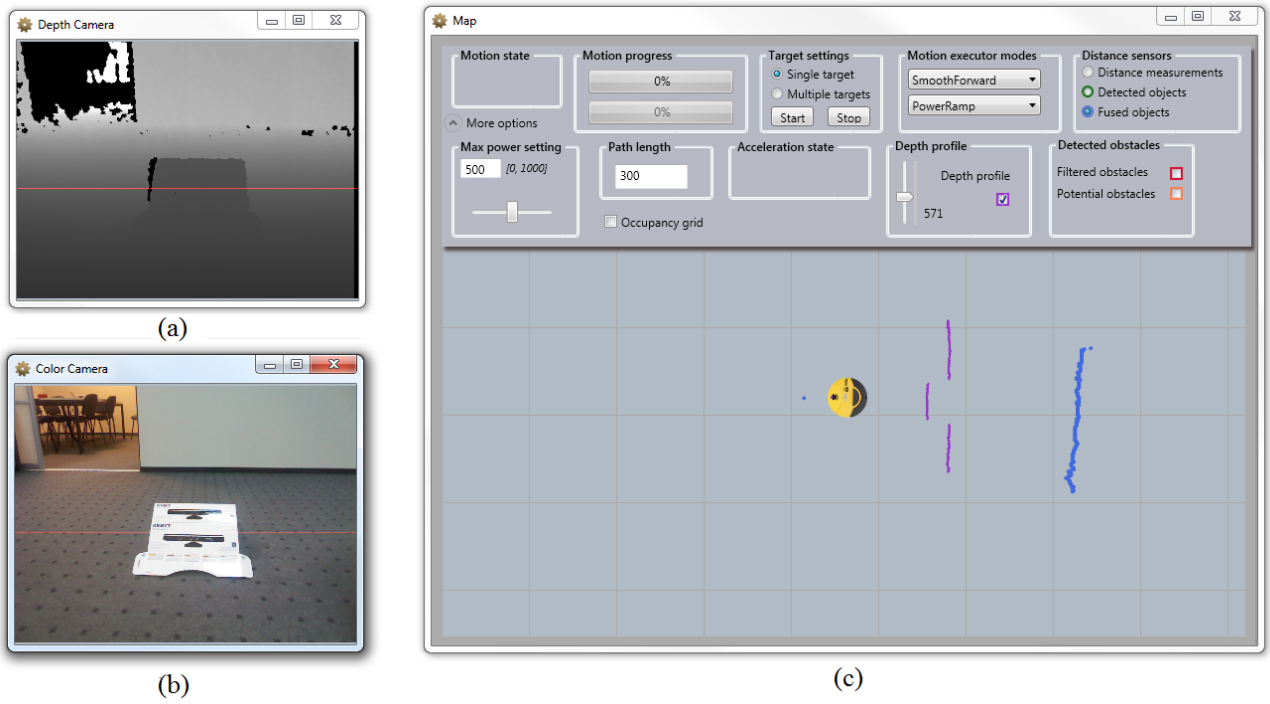
5.3. Nuožulnaus bei stataus objekto atpažinimo tyrimas

Šiame tyrime maksimali standartinio nuokrypio reikšmė nėra keičiama (viso eksperimento metu naudojama reikšmė – 0,015). Vietoj to, pakeičiama objekto struktūra. Pirmu atveju stebimas objektas yra didesnioji dėžė iš pirmojo eksperimento (žr. 5.3 pav.). Objektas yra atpažįstamas, jo taškai priskiriami kliūčių sričiai. Mėlynieji bei violetiniai taškai sutampa.



5.3 pav. (a) gylio vaizdas, (b) RGB kameros vaizdas, (c) žemėlapio vaizdas, kai maksimali standartinio nuokrypio reikšmė – 0,015 bei stebimas objektas yra su statmena plokštuma

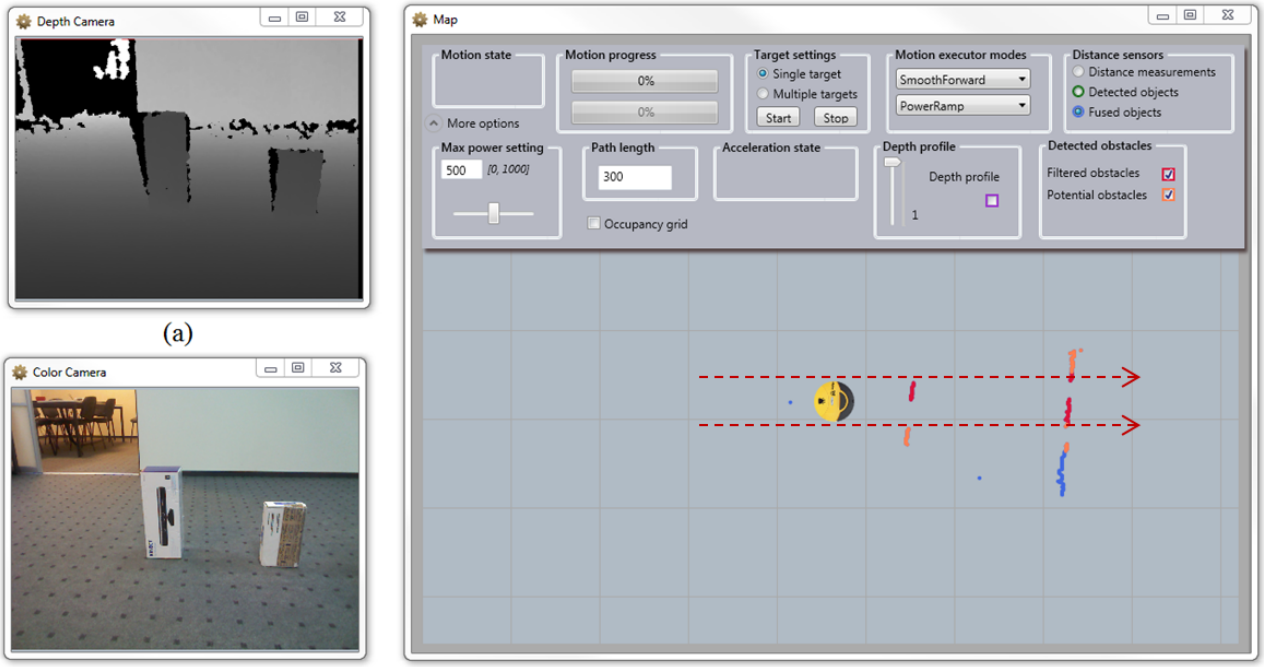
Antroje eksperimento dalyje dėžė buvo išardyta stengiantis panaudoti objektą kaip nuožulnią plokštumą (rampos imitacija). 5.4 paveiksle aiškiai matosi, kad pasirinktame aukštyje fiksuojami objekto taškai artimesni nei grindų taškai. Tačiau kliūčių masyve objekto taškų nėra. Algoritmas veikia teisingai, nes nuožulnios plokštumos taškų standartinis nuokrypis nėra toks aukštas kaip statmenos kliūties.



5.4 pav. (a) gylio vaizdas, (b) RGB kameros vaizdas, (c) žemėlapio vaizdas, kai maksimali standartinio nuokrypio reikšmė – 0,015 bei stebimas objektas yra su nuožulnia plokštuma

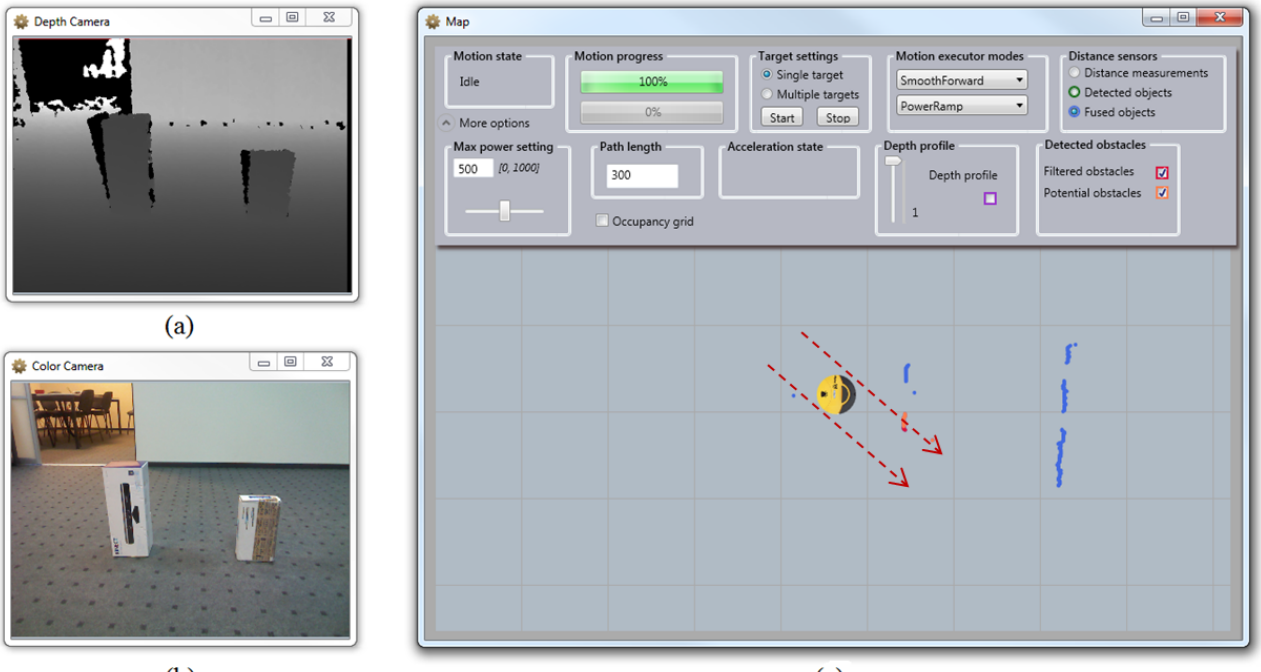
5.4. Horizontaliai išsidėsčiusių objektų filtravimo tyrimas

Šiame eksperimente buvo nagrinėjamas horizontaliai išsidėsčiusių objektų filtravimas pagal roboto judėjimo trajektorija. Pirmu atveju buvo pasirinktas nulinis ratų pasisukimo kampas (robotas imituoja judėjimą tiesiai). 5.5 paveiksle aptiktos kliūtys pažymėtos raudonai, potencialios kliūtys – oranžine spalva, likusieji objektai – mėlyna spalva. Įsitikinta, kad kliūčių bei potencialių kliūčių masyvai sudaromi korektiškai, kai robotas juda tiesiai.



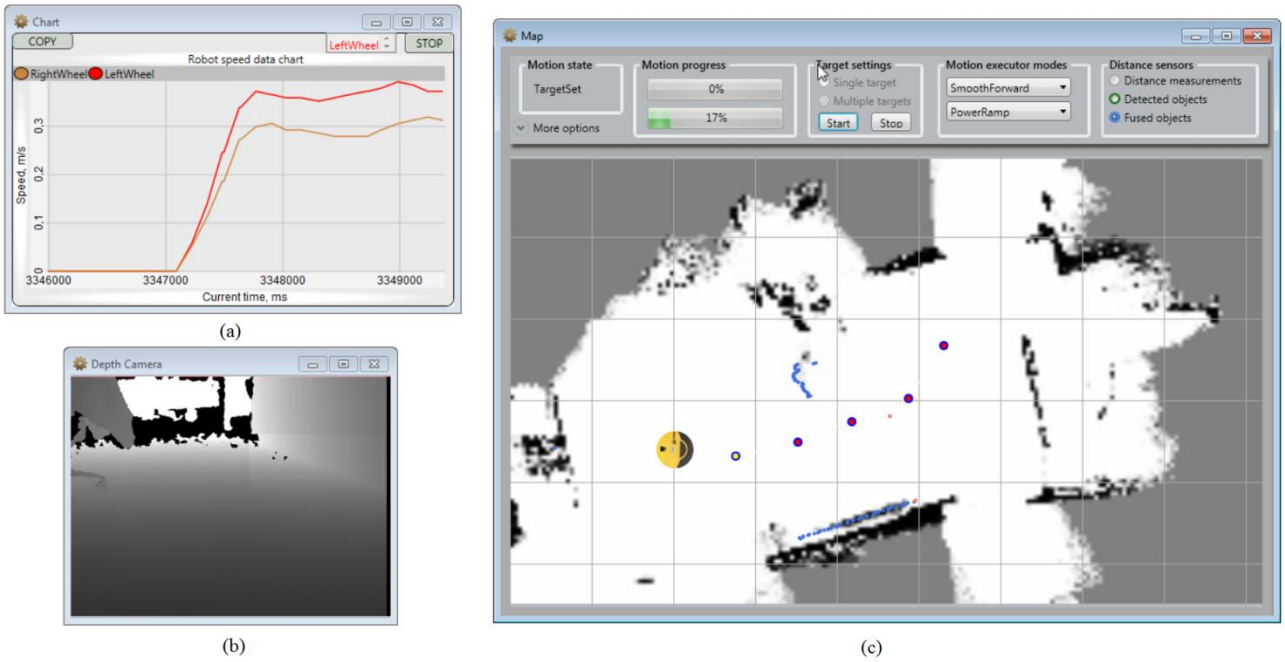
5.5 pav. (a) gylio vaizdas, (b) RGB kameros vaizdas, (c) žemėlapio vaizdas, kai roboto judėjimo kampas yra nulinis

Kitu atveju buvo atliktas roboto ratų pasukimas 45° (žr. 5.6 pav.). Kairėje pusėje esančios dėžės taškai tapo nesvarbūs (mėlyna spalva). Dešiniosios dėžės kraštiniai taškai yra fiksuojami kaip kliūtys. Mėlynas taškelis už roboto yra ultragarso daviklio fiksuojamas objektas, kuris taip pat nėra traktuojamas kaip pavojingas objektas.



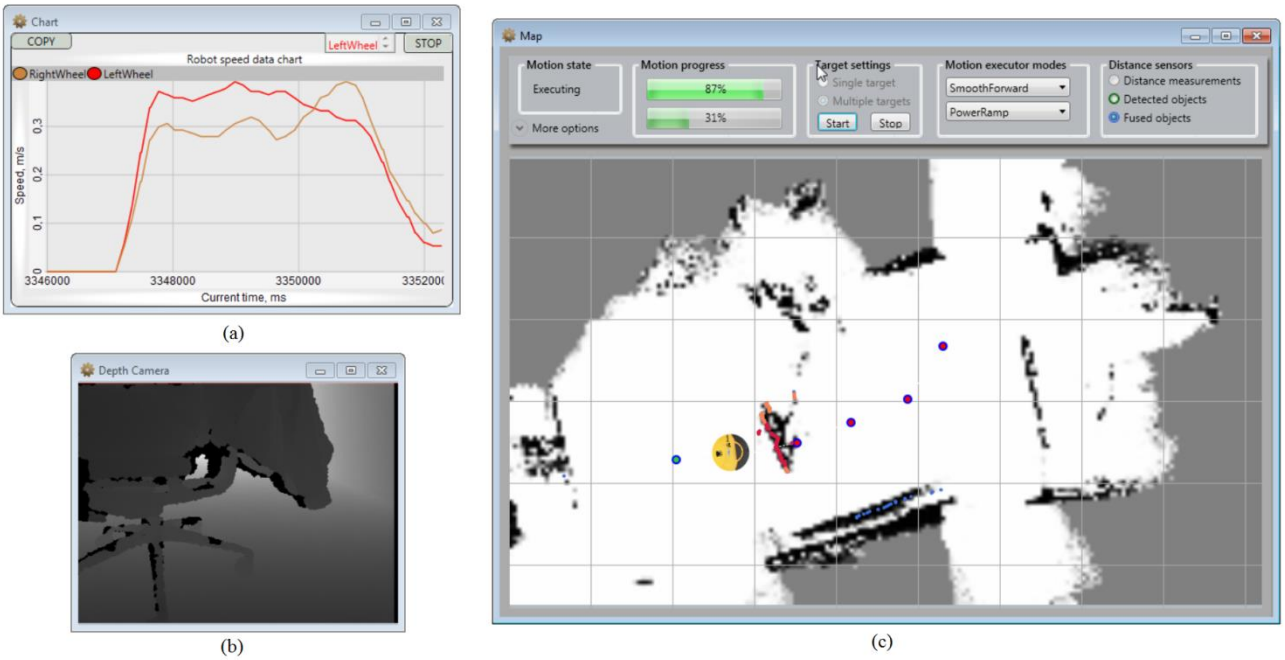
5.6 pav. (a) gylio vaizdas, (b) RGB kameros vaizdas, (c) žemėlapio vaizdas, kai roboto judėjimo kampas yra 45°

Eksperimento tikslas – patikrinti ar robotas išvengia susidūrimo su dinaminę kliūtimi. Pirmiausia atliekamas trajektorijos sudarymas – žemėlapyje sudedami taškai. Po to inicijuojamas roboto judėjimas šiais taškais. 5.7 paveiksle pateikti sistemos langų vaizdai, kai robotas pradėjo judėjimą užduota trajektorija bei pasiekia maksimalų greitį – 40 cm/s.



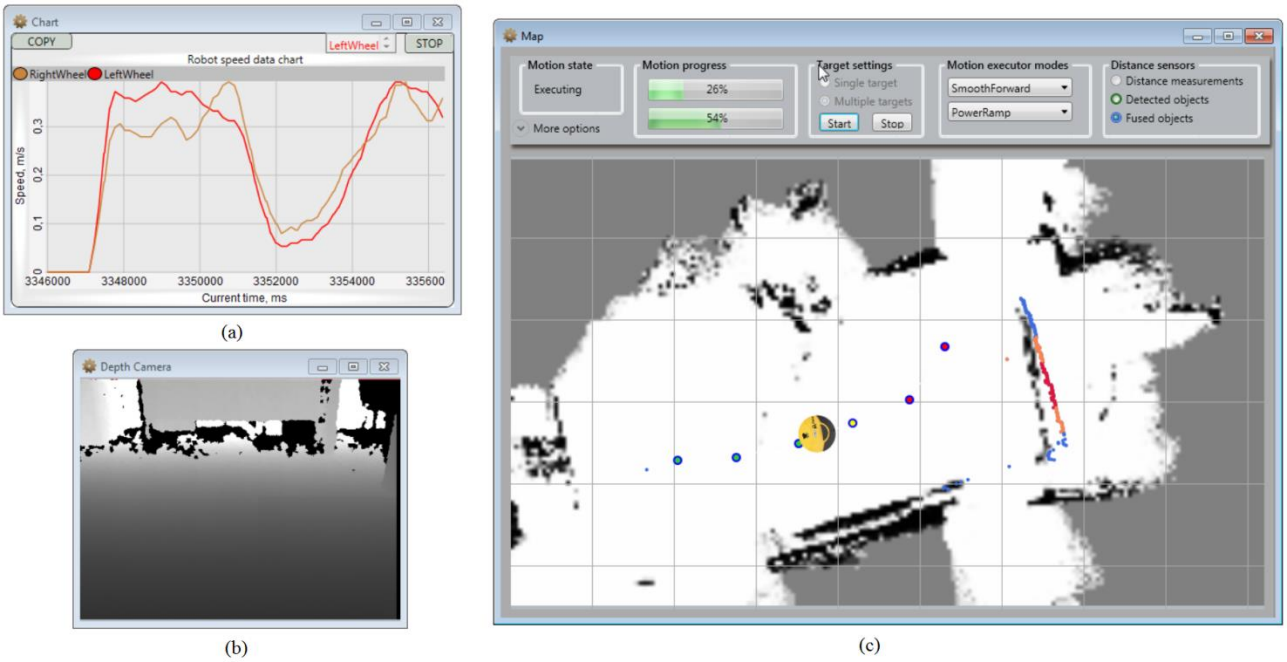
5.7 pav. (a) greičio reikšmių grafikas, (b) gylio vaizdas, (c) žemėlapio vaizdas, kai roboto judėjimo trajektorija laisva

Sekančiame vaizde (žr. 5.8 pav.) į stebimą aplinka patenka judanti kliūtis. Roboto jutikliai apdoroja aptiktus objektus – raudoni taškai. Atitinkamai roboto platformos greitis palaipsniui sumažinamas, kad būtų išvengta susidūrimo su aptikta kliūtimi.



5.8 pav. (a) greičio reikšmių grafikas, (b) gylio vaizdas, (c) žemėlapio vaizdas, kai roboto judėjimo trajektorijoje atsiranda dinaminė kliūtis

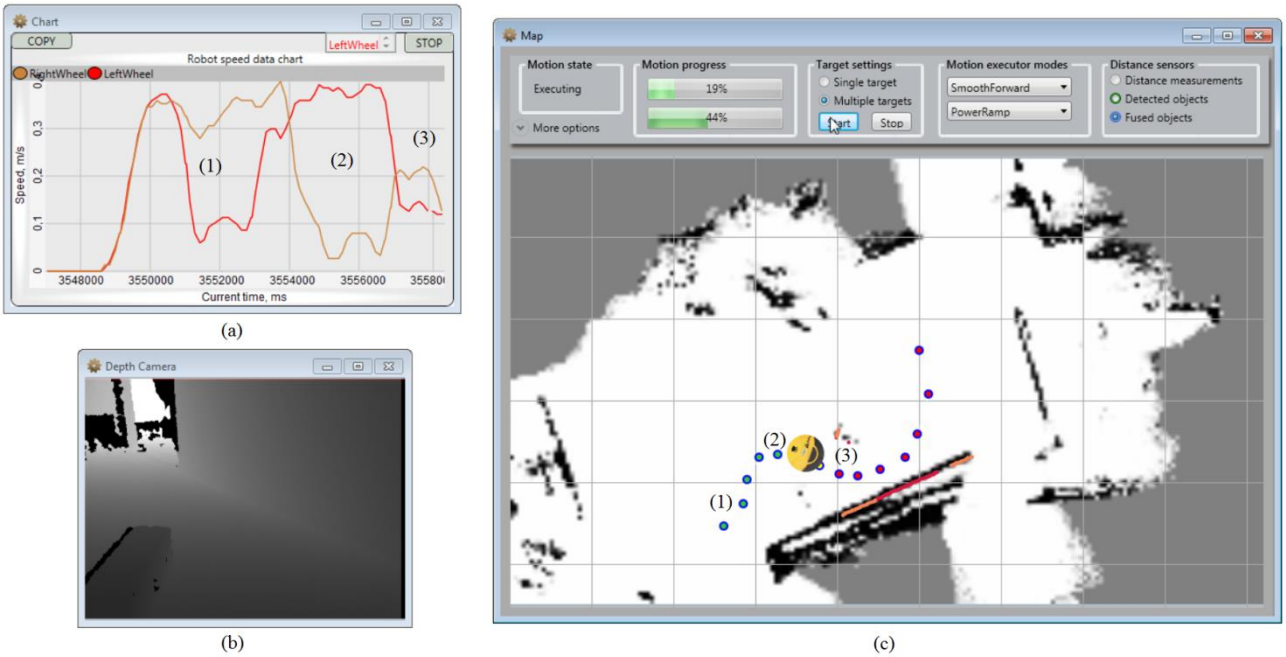
Galiausiai dinaminė kliūtis pasišalina (žr. 5.9 pav.). Robotui atsiveria laisvas kelias, todėl greitis palaipsniui didinamas iki nustatytos maksimalios greičio reikšmės. Roboto greitis yra kontroliuojamas PID (angl. Proportional-integral-derivative) valdiklio, todėl stebimas nuožulnus greičio reikšmių leidimasis bei kylimas.



5.9 pav. (a) greičio reikšmių grafikas, (b) gylio vaizdas, (c) žemėlapio vaizdas, kai dinaminė kliūtis pasišalina

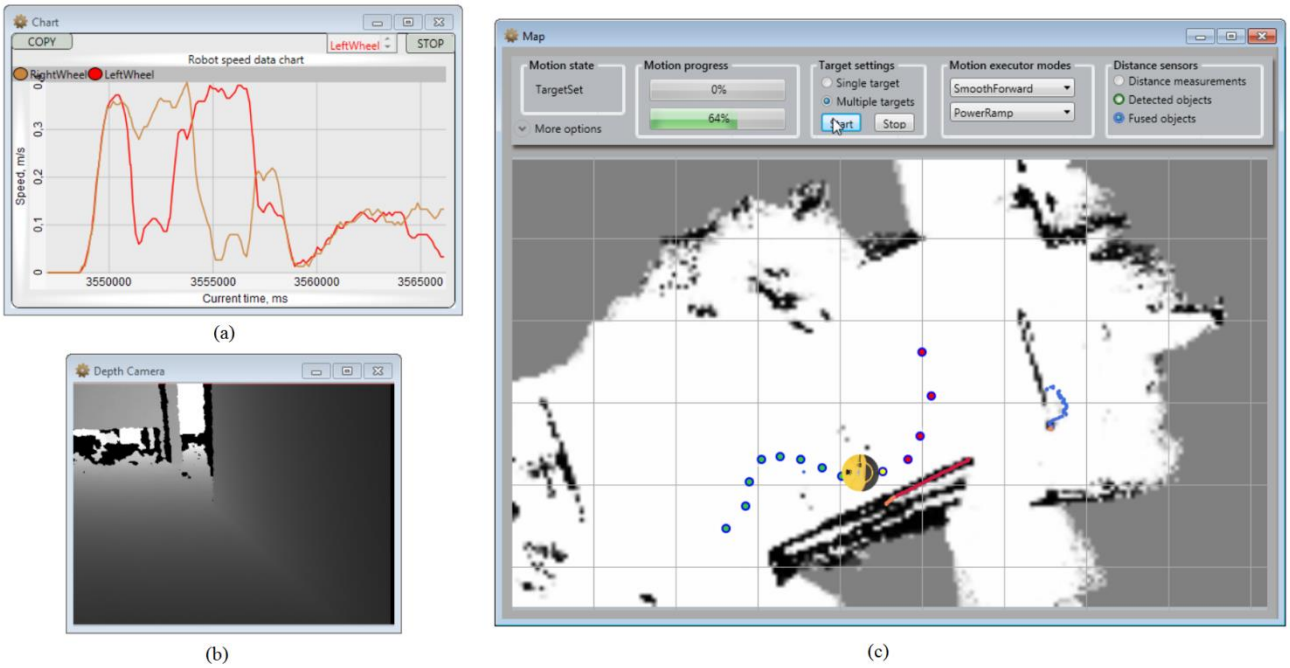
5.5. Važiavimas užduota trajektorija su statinėmis kliūtimis

Šiuo eksperimentu norima patikrinti kaip valdomas roboto platformos greitis, kai aplink judėjimo trajektorijos lauką aptinkamos statinės kliūtys. 5.10 paveiksle pateikti grafinės vartotojo sąsajos langai, kai robotas atlieka judėjimą tarp dviejų kliūčių. Roboto išskirtos trys roboto važiavimo fazės. Pirmoje bei antroje fazėje atliekami posūkiai, dėl to vieno rato greitis staiga sumažėja. Trečioje judėjimo fazėje kamera aptinka sieną bei šoninę kliūtį, kuri yra pavojingai arti roboto. Šiuo atveju greitis mažinamas iki 10 cm/s.



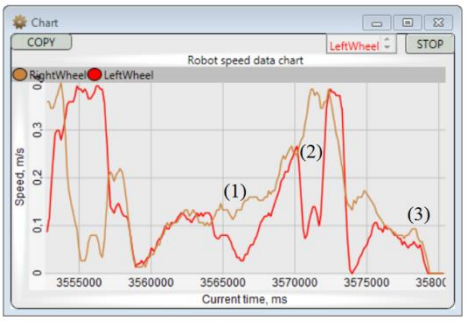
5.10 pav. (a) greičio reikšmių grafikas, (b) gylio vaizdas, (c) žemėlapio vaizdas, kai fiksuojamos artimos kliūtys

Sekančiame vaizde (žr. 5.11 pav.) robotas atlieka pavojingą judėjimą arti sienos, kas sąlygoja beveik visišką ratų sustabdymą. Kol kas nustatymuose naudojamos reikšmės, kurios priverčia robotą būti labai atsargiu. Taip daroma todėl, kad Kinect prietaisas gali fiksuoti tik toliau nei 80 cm esančius objektus. Jeigu objektai atsiduria arčiau nei gali aptikti Kinect kamera yra tikimybė, kad filtruojamų objektų nebeliks, todėl robotas nesustos. Taip pat yra vietų, kurių kameros matymo laukas nepadengia. Panaudojus daugiau aplinką stebinčių jutiklių galima būtų vykdyti sklandesnį bei greitesnį roboto judėjimą šalia esančių kliūčių.

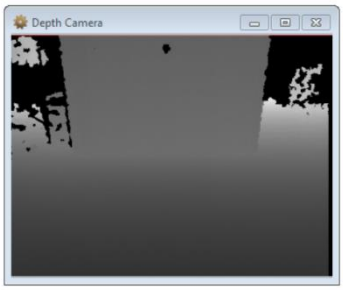


5.11 pav. (a) greičio reikšmių grafikas, (b) gylio vaizdas, (c) žemėlapio vaizdas, kai fiksuojami sienos taškai

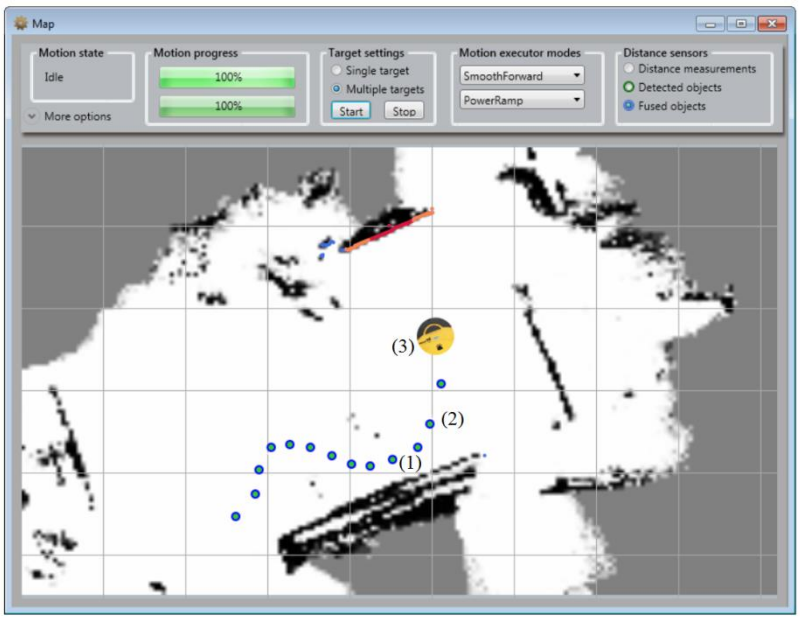
Paskutiniame vaizde (žr. 5.12 pav.) pateiktas roboto įveiktas kelias. Pirmoje fazėje robotui atsiveria tuštesnė erdvė – greitis didėja. Antroje fazėje atliekamas staigus posūkis. Galiausiai robotas pradeda stabdyti, kadangi yra vykdomas paskutinis užduotos trajektorijos taškas.



(a)



(b)



(c)

5.12 pav. (a) greičio reikšmių grafikas, (b) gylio vaizdas, (c) žemėlapio vaizdas, kai robotas pasiekia tikslą

6. IŠVADOS

1. Projektuojant sistemą buvo identifikuotos problemos bei veiksniai įtakojantys kuriamų komponentų veikimo principus. Tai priklauso nuo veikimo aplinkos, jutiklių, sistemos architektūros bei reikalavimų roboto platformai.
2. Išnagrinėjus įtakojančius veiksnius ir veikimo sąlygas, objektų atpažinimui nuspręsta panaudoti Kinect įrenginį. Taip pat panaudoti du ultragarso davikliai atstumams iki objektų matuoti. Vienas iš galutinės sistemos nefunkcinių reikalavimų – veikti lauko bei kambario sąlygomis. Tačiau magistro darbo metu atlikti eksperimentai buvo atliekami uždaroje patalpoje (motyvas – Kinect prietaiso apribojimai).
3. 3D kameros pasirinkimo priežastis – nuožulniųjų plokštumų bei mažų objektų atpažinimas. Vienas iš reikalavimų – gebėjimas įveikti mažas kliūtis kaip rampos arba slenksčiai. Šios problemos sprendimui buvo atlikta vaizdų apdorojimo metodų analizė. Išnagrinėtų algoritmų metodikos yra grįstos plokštumų generavimu. Tačiau šie metodai atlieka daug skaičiavimų kas lemia santykinai lėtą veikimą (pažeidžiamas reikalavimas – veikimas realiu laiku).
4. Suprojektuota bei sukurta roboto valdymo sistema su grafine vartotojo sąsaja. Pasirinkta hibridinė roboto valdymo architektūra, kuri skirstoma į atskirus sluoksnius pagal naudojamų komponentų abstraktumo lygį. Sistemos panaudojimo tikslai – patikrinti realizuojamus vidutinio bei aukšto lygio roboto komponentus. Šiame darbe buvo patikrinti vidutinio lygio arba vykdytojo sluoksnyje esančių komponentų veiksmai.
5. Pateikti sistemoje realizuotų metodai. Išsamiai apžvelgti bei pristatyti algoritmai: grindų atpažinimo algoritmas, horizontalių objektų filtravimas pagal roboto dydį bei judėjimo trajektoriją bei kliūčių vengimas.
6. Atlikus pirmus du eksperimentus buvo nustatyta, kad objektų klasifikavimas pagal taškų standartinio nuokrypį duoda teigiamus rezultatus. Keičiant maksimalų standartinio nuokrypio dydį yra filtruojami atitinkamo aukščio objektai. Šio algoritmo trūkumas – objektai neviršijantys standartinio nuokrypio dydžio yra priskiriami grindų plokštumai. Tokiu atveju robotas netenka dalies informacijos, kuri yra pakankamai svarbi. Pavyzdžiui, atliekant važiavimą per slenkstį robotas turėtų sumažinti greitį, kad nebūtų gadinama važiuoklė ar vežamas krovinys. Vienas iš paprasčiausių sprendimo būdų – pridėti minimalią standartinę reikšmę, kurią viršijus, aptikti taškai būtų priskiriami pasirinktam naujam objektų tipui.
7. Sekančio eksperimentinio tyrimo metu buvo analizuojamas horizontaliai išsidėsčiusių objektų filtravimas pagal roboto judėjimo trajektoriją algoritmo korektiškumas. Buvo atlikti du bandymai: roboto judėjimas tiesiai, bei roboto judėjimas 45° kampu pagal laikrodžio rodyklę. Abiem atvejais filtruojamos kliūtys atitiko judėjimo trajektoriją.
8. Galiausiai buvo tiriamas atrinktų kliūčių vengimo metodas paleidžiant robotą užduota trajektorija. Pirmu atveju buvo imituojama dinaminė kliūtis, kuri pastoja roboto platformai kelią. Kliūčių atpažinimo jutikliai greitai išskyrė pavojingų kliūties taškų masyvą, pagal kurį kliūčių vengimo algoritmas paskaičiavo reikiamą greitį. Taip pat buvo testuojamas roboto judėjimas užgrūstoje erdvėje. Abiem atvejais robotas išvengė susidūrimo su aptiktomis

kliūtimis. Greičio didinimui bei mažinimui panaudotas PID valdiklis, todėl buvo stebimas sklandus roboto važiavimas.

9. Įvertinus aukščiau aprašytų eksperimentų rezultatus galima teigti, kad kliūčių atpažinimo bei vengimo metodai veikia. Tačiau darbai turi būti tęsiami. Planuojama papildyti objektų klasifikavimo algoritmą, kuris galėtų išskirti nuožulnias plokštumas bei slenksčius papildomam greičio reguliavimui. Taip pat bus realizuojamas roboto kelio planavimas, kuriam reikalingas lokalus žemėlapis su pažymėtais objektais. Tokiu būdu robotas galėtų iš anksto apvažiuoti žinomas kliūtis.

7. LITERATŪRA

[1] J. Stonington. Becoming the Microsoft of the Robot World [Žiūrėta 2012 11 18], prieiga internete <http://www.businessweek.com/technology/content/nov2010/tc2010111_884564.htm#p1>

[2] R.A. El-laithy, J. Huang, M. Yeh. Study on the Use of Microsoft Kinect for Robotics Applications. IEEE, 2012.

[3] A.V.H. Ollikkala, A. J. Mäkynen. Tree Mapping Using a Time-Of-Flight 3D Camera. University of Oulu Technology Park 127, 87400 Kajaani, Finland, 2009.

[4] R. Schnabel, R. Wahl, R. Klein. Efficient RANSAC for Point-Cloud Shape Detection.University of Bonn, 2007.

[5] M. Y. Yang, W. Förstner. Plane Detection in Point Cloud Data. University of Bonn, 2010.

[6] N. Zhang. Plane Fitting on Airborne Laser Scanning Data Using RANSAC [Žiūrėta 2014 01 14], prieiga internete < http://www.maths.lth.se/matematiklth/personal/petter/rapporter/plane%20fitting.pdf>

[7] J.E. Deschaud, F. Goulette. A Fast Accurate Plane Detection Algorithm for Large Noisy Point Clouds Using Filtered Normals and Voxel Growing. Mines ParisTech, CAOR-Centre de Robotique, 2010.

[8] G. McComb, M. Predko. Robot Builder's Bonanza (Third Edition). McGraw-Hill, 2006.

[9] R. Damaševičius. Robotų programavimo technologijos. Kauno technologijos universitetas, 2010.