VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
KOMPIUTERIJOS KATEDRA

Magistro baigiamasis darbas

Judėjimo trajektorijos modeliavimas naudojant atstumo jutiklius

Atliko: Paulius Jonikas (parašas)

Darbo vadovas:

dr. L. Bukauskas (parašas)

Recenzentas:

(parašas)

Vilnius
2012

Turinys

Iliustracijų sąrašas

Sutartinių terminų sąrašas

- Automatinė parkavimo sistema (APS) – technologija įgalinanti automobilį automatiškai, be vairuotojo įsikišimo, atlikti parkavimosi manevrus.
- CCR (angl. concurrency and coordination runtime) – konkuravimo ir koordinavimo aplinka; asinchroninio programavimo biblioteka paremta .NET platforma, naudojama Microsoft Robotic Developer Studio projektams kurti.
- DSS (angl.) – aplinka naudojama Microsoft Robotics Develpoer Studio projektams kurti, kuri suteikia galimybę valdyti sudėtingus modelius sudarytus iš daugelio servisų (paskirstytas aplikacijas).
- Fuzzy logika – „netikslios“ logikos taisyklės.
- Kamera – vaizdo kamera skirta nuotraukai iš automobilio galinės dalies gauti, vėliau naudojama susieti automobilį su koordinačių sistema.
- „Kišenė“ – automobilio parkavimosi būdas lygiagrečiai judėjimo trajektorijos tarp kitų transporto priemonių.
- MRDS/MRDS 4 – Microsoft Robotics Developer Studio 4 – Windows operacinei sistemai sukurta platforma robotų valdymui ir simuliacijai. Pagrindinė platformos auditorija akdeminė bendruomenė, robotikos fanai ir komercinei naudotojai, kuriantys robotikos aplikacijas fiziniams robotams.
- Sonaras – ultragarso pagalba atstumą iki kliūties nustatantis jutiklis.
- VPL (angl. visual programming language) – Microsoft Robotics Developer Studio pakete esantis įrankis, kurio dėka galima vizualiai kurti robotikos modelius pasitelkiant esamus (jau sukurtus) komponentus. Darbo principas paremtas vilkimo (angl. drag and drop) principu.
- VSE (angl. visual simulation environment) – Microsoft Robotics Developer Studio pakete esanti grafinė fizikinį simuliatorių turinti aplinka, kuri suteikia galimybę kurti robotikos modelius ir eksperimentuoti 3D aplinkoje.

Anotacija

Darbe nagrinėjami algoritmai skirti automatiniam automobilio parkavimui. Pasirinkti du algoritmai: pirmasis remiasi ultragarso sonarų teikiama informacija, antrasis ultragarso sonarų ir vaizdo kameros teikiamais duomenimis. Abu algoritmai remiasi fuzzy logika. Nustatyta, kad realiomis sąlygomis atlikti automatinį automobilio parkavimą remiantis tik ultragarso sonarais yra nepatikima – patikimumas ir sistemos nauda ženkliai pagerėja pasitelkus vaizdo kamerą.

Įsisavinus ir perpratus robotikos platformą Microsoft Robotics Developer Studio 4 sukurtas automobilio modelis grafinėje miesto aplinkoje. Modelis aprūpintas sonarais ir vaizdo kameromis, todėl gali būti universaliai naudojamas parkavimosi algoritmų bandymams ir analizei Microsoft Robotics Developer Studio 4 vizualios simuliacijos aplinkoje (VSE).

Summary

The thesis „Motion Trajectory Planning Using Distance Sensors“ deals with algorithms for automatic car parking. There were chosen two algorithms: the first is based on information provided by ultrasonic sonars, and the second is based on ultrasonic sonars and video camera. It was found that under realistic conditions algorithm based only on sonars is not such reliable as algorithm based on ultrasonic sonars and video camera, which significantly improves the results.

There was created graphical model of the car in urban environment in Microsoft Robotics Developer Studio 4. The model is equipped with sonars and video cameras, so it can be universally used for parking algorithms analysis and testing in Microsoft Robotics Developer Studio 4 visual simulation environment (VSE).

Įvadas

Automatinės parkavimo sistemos (APS) gali padėti parkuoti automobilį tiek patyrusiems, tiek ilgalaikių vairavimo įgūdžių neturintiems vairuotojams. Daugelis automobilių gamintojų jau sukūrė pagalbines parkavimo sistemas ar automatinio parkavimo sprendimus, tačiau šių sistemų patikimumas ir kaina kol kas lieka aktualūs. APS sistemų darbo principas susideda iš kelių dalių: automobilio parkavimo vietos suradimo, automobilio lokalizavimo erdvėje ir automobilio pastatymo į numatytą vietą. Minėti manevrai yra euristinis uždavinys, nes reikia atsižvelgti į jutiklių paklaidas, bei galimybę, kad vykdant parkavimo manevrą pasikeis nustatytas aplinkos modelis. Labai svarbu suprasti, kad profesionalaus vairuotojo įgūdžių perkėlimas APS sistemoms gali ženkliai sumažinti vairavimo nepatogumus bei padidinti saugumą dabartiniuose ir netolimos ateities transporto priemonėse.

Automatinių parkavimo sistemų techninei realizacijai dažniausiai naudojami ultragarso sonarai ir vaizdo kameros. Jų dėka suformuojamas aplinkos vaizdas kompiuteryje bei randamas geriausias parkavimosi manevro pradžios taškas. Įgyvendinant automobilio parkavimosi trajektorijos skaičiavimus svarbu atsižvelgti į aplinkos modelį bei fizinius automobilio parametrus ir apribojimus (pvz., galimą automobilio vairo pasukimo kampą, automobilio ilgį, atstumo jutiklių paklaidą ir kt.).

Pirmojij APS sistemas Jungtinėse Amerikos Valstijose panaudojo kompanija Lexus 2004 metais. Tuo tarpu pirmieji automobiliai su APS sistemomis, kurios vėliau buvo panaudotos ir Lexus, buvo Toyota Prius modeliai sukurti Japonijos rinkai.

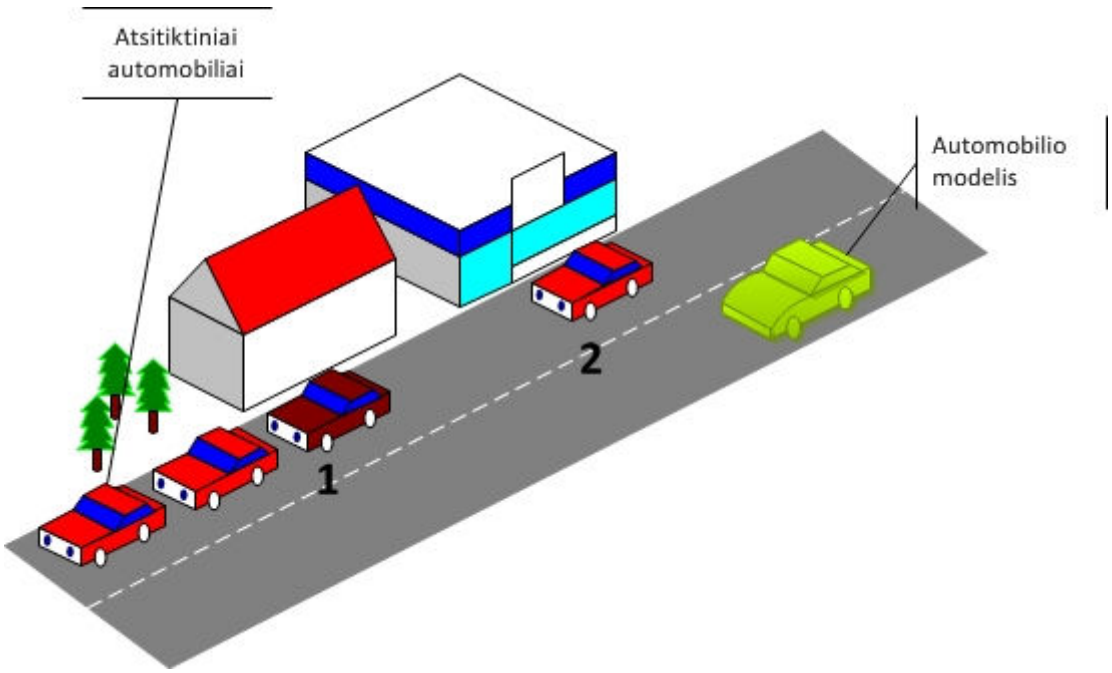
Darbo metu buvo palyginti du fuzzy logiką naudojantys parkavimosi algoritmai. Pagal nagrinėtą medžiagą nustatyta, kad ultragarso sonarų naudojimas yra nepakankamas siekiant geros APS sistemų kokybės (tikslumo). Todėl APS modeliai praplečiami papildomais sensoriais, kuriais įprastai tampa vaizdo kameros. Tolimesnio darbo metu įsisavinant šiuo metu naujausią Microsoft Robotics Developer Studio 4 robotikos platformos versiją sukurtas automobilio modelis, kurį galima pakartotinai naudoti įvairių parkavimosi algoritmų bandymams ir analizei vizualioje 3D Microsoft Robotics Developer Studio 4 aplinkoje (VSE). Esant poreikiui modelis lengvai modifikuojamas – galima greitai ir paprastai pridėti papildomus sensorius, kiekvienam jų nustatyti specifinius parametrus. Sukurtas automobilio modelis be papildomo modifikavimo tinkamas vieno iš nagrinėtų algoritmų bandymams atlikti.

1. Fuzzy logikos taikymas automobilio parkavimui naudojantis sonarais

Iliustracijoje 1 pateiktas parkavimo scenarijus, kai automobilis pravažiavęs lygiagrečiai gatvėje pastatytų transporto priemonių randa laisvą vietą ir į ją įsiparkuoja [HP96]. Eksperimentas atliktas dešimt kart sumažinus realius daiktų matmenis, siekiant, kad visą modelį būtų galima sukurti ir išbandyti darbo stalo dydžio aplinkoje. Automobilio modelis 1 iliustracijoje yra pradinėje padėtyje. Eksperimento tikslas – remiantis fuzzy logika patalpinti automobilį į erdvę tarp 1 ir 2 atsitiktinių automobilių. Užduotis susideda iš keletos veiksmų:

- atpažinti laisvą erdvę automobilio šone, pravažiuojant pro ją;
- nustatyti automatinio parkavimosi pradžios tašką (automobilio parkavimosi starto poziciją);
- atlikti automatinio parkavimosi veiksmus;
- išvengti bet kokių susidūrimų su kliūtimis.

1 iliustracija. Automobilio parkavimasis tarp dviejų automobilių kelio pakraštyje

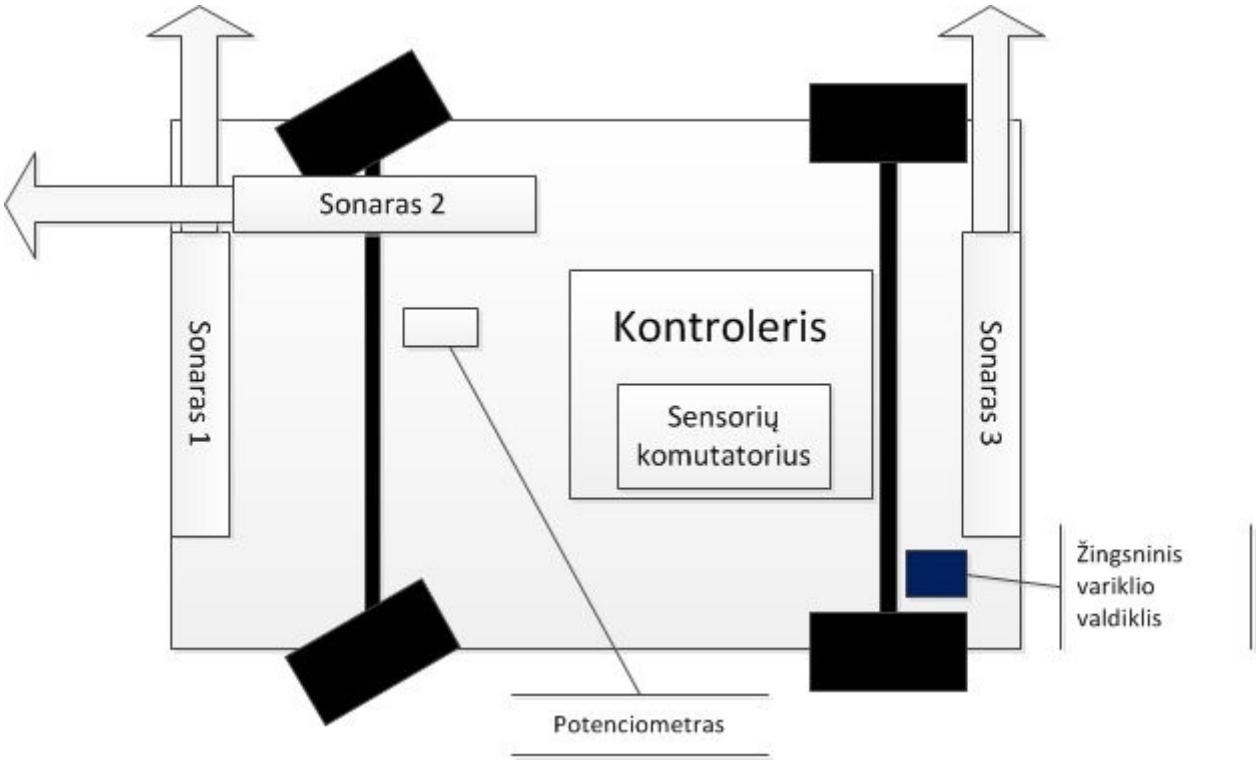


Eksperimento tikslas – išspręsti parkavimosi problemą taip, kad sprendimą be didesnių modifikacijų būtų galima pritaikyti kasdienybėje.

1.1. Techninis sprendimas

Techninis autonominio automobilio modelio skirto prisiparkuoti tarp dviejų automobilių gatvės pakraštyje („į kišenę“) sprendimas pateikiamas 2 iliustracijoje. Schemoje nepateikiami vairo mechanizmo ir greičių dėžės aprašai.

2 iliustracija. Techninis automobilio modelio sprendimas

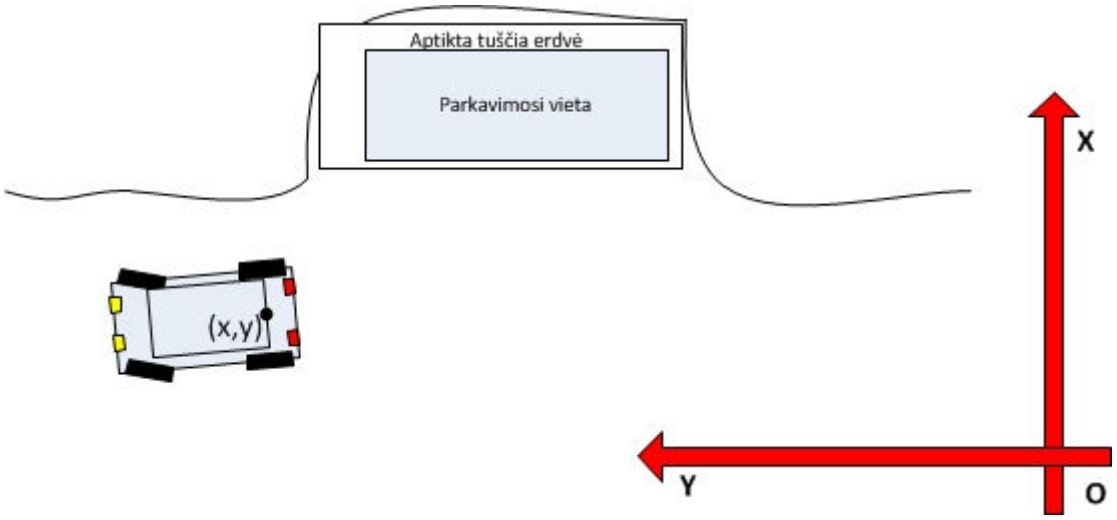


Automobilio modelyje esantis kontroleris yra sujungtas su kompiuteriu per SERIAL jungtį. PASCAL ir asemblerio programavimo kalbomis realiu laiku veikianti kontrolerio programinė įranga apdoroja visų sonarų pateikiamą informaciją ir SERIAL jungties dėka perduoda signalus į personalinį kompiuterį, kur vykdomas pagrindinis automatinio parkavimo „į kišenę“ algoritmas. Kompiuteris perduoda komandas vairo mechanizmui ir varikliams. Modelyje naudoti trys sonarai matuoja atstumą 0.05 – 0.5 metro atstumu su 5 laipsnių spindulio pločiu. Visi sonarai yra valdomi sensorių komunikatoriaus, kuris užtikrina, kad skirtingų sensorių signalai nepersidengs. Ant modelio sumontuotas žingsninis variklio valdiklis leidžia apibrėžti automobilio modelio judėjimo greitį bei pagal žingsnius apskaičiuoti nuvažiuotą atstumą.Potenciometras bendrame modelyje leidžia matuoti ir 3 laipsnių tikslumu žinoti vairo pasukimo kampą. Eksperimente pateikto automobilio modelio vairo posūkio kampas tik +/- 28 laipsniais, tuo tarpu, realaus automobilio vairas gali suktis +/- 45 laipsnių kampu.

1.2. Sukurto „pasaulio“ modeliavimas

Nors aprašomas modelis turi ribotą sonarų skaičių (tik tris), tačiau galime stebėti visą dešinį automobilio šoną (vengti susidūrimo su kliūtimis). Iš kitos pusės, pasirinktas sprendimas su sonarais neleidžia užtikrinti, kad parkavimosi vieta bus nustatyta teisingai, ką galima greitai atsakyti naudojant lazerį. Tačiau šio eksperimento metu nėra siekiama perkrauti modelio pertekline informacija, todėl pasirinktas paprastas sprendimas su trim sonarais. „Pasaulio“ modeliavimas vyksta automobilio modeliui lėtai važiuojant pro parkavimosi vietą (3 iliustracija). Dėl sonarų paklaidų aptikta tuščia erdvė nėra taisyklingos formos, kaip parodyta iliustracijoje.

3 iliustracija. Sonarais nustatyta parkavimosi vieta, koordinačių sistema ir supančių kliūčių modelis



1.3. Parkavimosi algoritmas

Parkavimosi problematika ir pats algoritmas mūsų atveju skyla į žemiau pateiktas dalis.

1.3.1. Preliminarios parkavimosi vietos paieška

Automobilio modelis važiuodamas pirmyn lygiagrečiai gatvės krašto dešiniajame šone ieško laisvos parkavimosi vietos (kuria aplinkos/pasaulio modelį). Remiantis šią sąlyga automobilis važiuoja gatve laikydamasis saugaus atstumo nuo kitų automobilių ir sienos (kuri realiame pasaulyje greičiausiai būtų šaligatvio kraštas) tol, kol randa tinkamą parkavimosi vietą pagal fuzzy logikos kontrolerio taisykles:

jei **ATSTUMAS** iki dešiniajame šone esančios kliūties yra (**DIDELIS** arba **LABAI LABAI MAŽAS**) tada **SUKAM VAIRĄ į KAIRĘ;**

jei **ATSTUMAS** iki dešiniajame šone esančios kliūties yra (**LABAI LABAI DIDELIS** arba **MAŽAS**) tada **SUKAM VAIRĄ į DEŠINĘ;**

jei **ATSTUMAS** iki dešiniajame šone esančios kliūties yra (**LABAI DIDELIS** arba **LABAI MAŽAS**) tada **JUDAME TIESIAI.**

Remiantis aukščiau pateiktomis taisyklėmis automobilis manevruodamas niekada neišklysta iš tuščios parkavimosi vietos paieškos trajektorijos.

1.3.2. Automobilio lokalizavimas

Algoritmo atveju (su konkrečiu modelio dydžiu) geriausia startavimo padėtis parkavimui „į kišenę“ pradėti yra apie 0.17 m priekyje tuščios vietos. Automobilio modelio padėtis parkavimosi pradžioje (radus vietą) pateikta 3 iliustracijoje.

1.3.3. Automobilio parkavimas atbuline eiga

Manevras labai panašus į tai kas mokoma vairavimo mokyklose ir reikalaujama per praktinį vairavimo egzaminą:

- vairas maksimaliai susukamas į dešinę, važiuojama maksimaliai giliai į parkavimosi vietą;
- vairas maksimaliai susukamas į kairę, važiuojama maksimaliai į parkavimosi vietos galą.

Jei parkavimosi vieta yra pakankamai ilga, tai šiais dviem veiksmais užtikrinamas automobilio priprakavimas pasirinktoje vietoje, tačiau jei to nepavyksta atlikti šiais dviem veiksmais vykdomas fuzzy logikos kontroleris, kuris atlieka veiksmus panašius ką daro vairuotojas, kai nepavyksta prisiparkuoti iš pirmo karto: važiuojama atgal ir priekin sukant vairą kairėn ir dešinėn pamažu pastatant automobilį į numatytą vietą.

2. Fuzzy logikos taikymas automobilio parkavimui naudojant sonarus ir vaizdo kamerą

Aprašomas algoritmas naudoja ultragarso sonarus bei vaizdo kamerą ir kitaip nei „Fuzzy logikos taikymas automobilio parkavimui naudojantis sonarais“ skyriuje pateiktas algoritmas gali be didesnių patobulinimų būti taikomas automobilių parkavimui [YSS08]. Pagrindinis skirtumas nuo ankščiau pateikto algoritmo yra automobilio judėjimo trajektorijos koordinavimas ir tinkamos parkavimosi vietos nustatymas naudojant tiek ultragarso sonarus, tiek galinę vaizdo kamerą. Naudojant abiejų tipų sensorius kartu galima išvengti situacijų, kai vienas jų „klystų“ dėl turimų apribojimų ar paklaidų. Antra vertus pateikiamame algoritme nėra naudojamas nuvažiuoto kelio ar greičio įvertis, taip ženkliai supaprastinant algoritmo taikymą tikrose transporto priemonėse, kur tikslus minėtų įverčių matavimas gali būti per daug sudėtingas ir finansiškai brangus. Taip pat šis algoritmas pritaikytas tiek parkavimui tarp dviejų automobilių gatvės šone („į kišenę“), tiek parkavimui garaže.

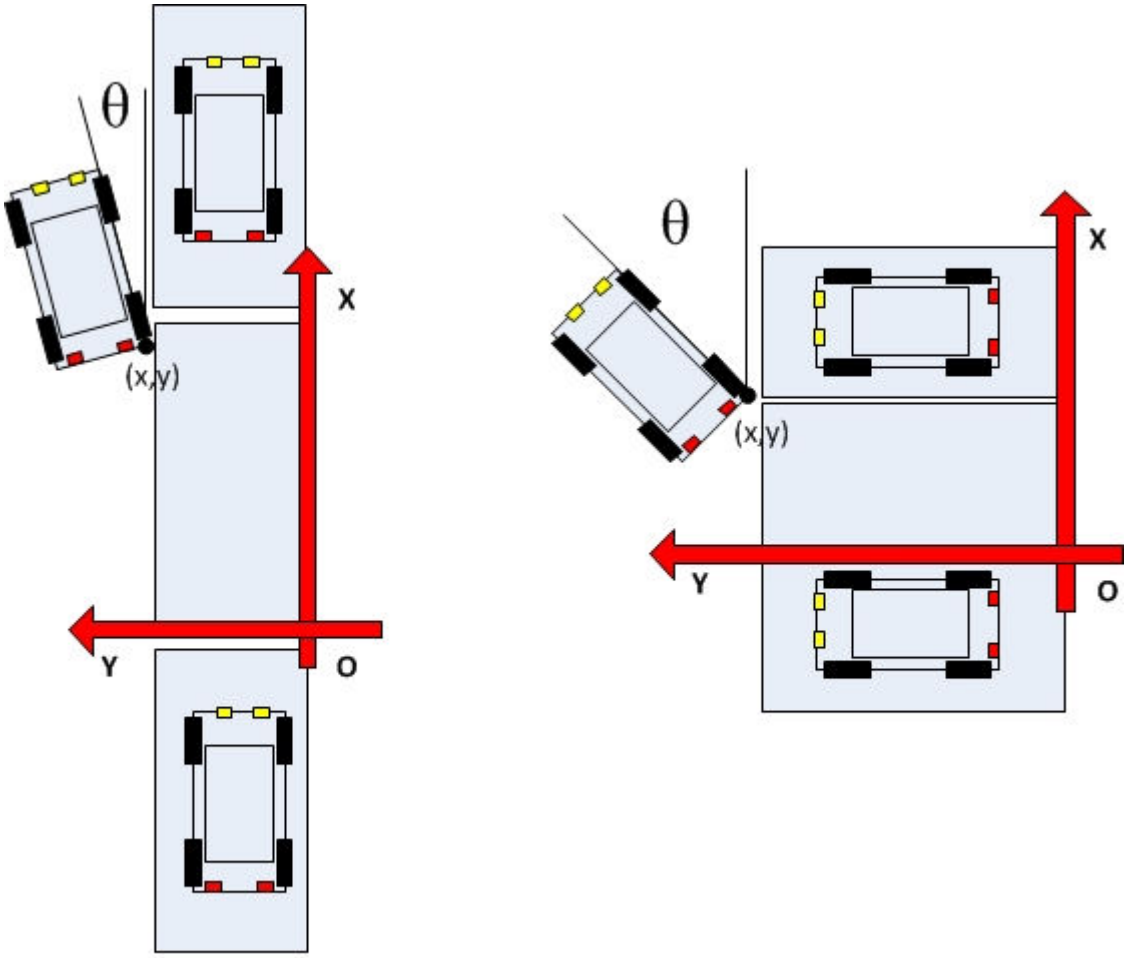
2.1. Automobilio vietos lokalizavimas ir fuzzy logika naudojant euristinių taisyklių rinkinį

Automobilio parkavimas kaip ir „Fuzzy logikos taikymas automobilio parkavimui naudojantis sonarais“ algoritme susideda iš kelių etapų. Pirmiausia sistema nustato, kuria kryptimi galima parkuotis (automobilį nukreipia lygiagrečiai parkavimosi vietoms). Nustačius reikiamą parkavimosi vietų paieškos kryptį automobilis juda lygiagrečiai galimų tuščių vietų, o radęs tinkamą įvertina kurį parkavimos būdą pasirinkti: „į kišenę“ ar garažą (priklausomai nuo išmatuoto parkavimosi vietos gylio ir pločio). Atlikus pastaruosius du veiksmus sistema sustoja geriausioje nustatytoje pozicijoje ir pasiruošia parkavimui atbuline eiga. Toliau, remiantis fuzzy logikos kontroleriu, vykdomas automobilio manevravimas į pasirinktą vietą.

2.1.1. Automobilio lokalizavimas

Siekdami, kad automobilis atliktų automatinį parkavimosi manevrą pats, turime žinoti tikslią automobilio poziciją pasirinktos parkavimosi vietos atžvilgiu – automobilio pozicijos vektorių $X = [x, y, \theta]$. Siekiant išvengti galimų parkavimosi vietos modelio klaidų naudojami tiek ultragarso sonarai, tiek vaizdo kamera. Iliustracijoje 4 pateikiama algoritme naudojama koordinačių sistema parkavimosi „į kišenę“, bei garažą atvejais.

4 iliustracija. Automobilio parkavimo būdai ir koordinačių sistemos



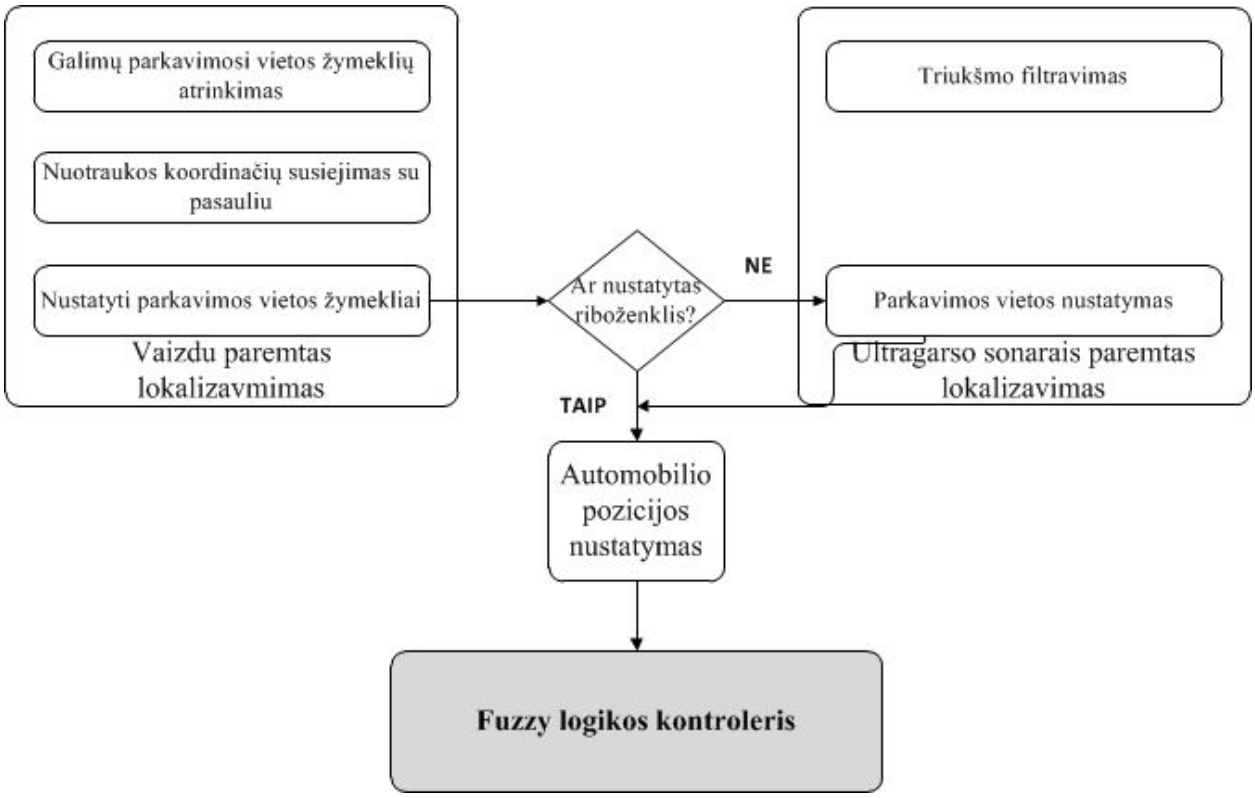
2.1.2. Automobilio lokalizavimas – ultragarso sonarų ir kameros duomenų apjungimas

Jei aplink numatytą parkavimos vietą yra kitų automobilių ar aiškių sienų, ar kitų aiškiai erdvę ribojančių objektų – automobilio padėtį galima paprastai apskaičiuoti naudojant ultragarso sonarus. Tačiau, sonarai gali turėti didelę krypties paklaidą bei teikti rezultatus su dideliais nuokrypiais priklausomai nuo ultragarsą atspindinčių paviršių. Šiai problemai spręsti naudojama vaizdo kamera. Detalus ultragarso sonarų ir vaizdo kameros rezultatų apjungimas pateikiamas 5 iliustracijoje pateiktame pseudo kode. Algoritmo rezultas – automobilio pozicija erdvėje apibrėžta vektoriumi $X = [x, y, \theta]$, kuris yra fuzzy logikos kontrolerio naudojamo automatiniam parkavimui atlikti įvestis. Papildomai vaizdo kameros teikiama nauda: nustatyti automobilio poziciją 4 iliustracijoje pateiktose koordinačių sistemose, kai automobilis yra „pasiruošęs

parkavimui atbuline eiga“ – pasiruošęs judėti į parkavimosi vietą ir su x koordinačių ašimi sudaro kampą θ.

Automobiliui „pasiruošus parkuotis atbuline eiga“ pasitelkus vaizdo kameros teikiamą informaciją vykdomas „Galimų parkavimosi vietos žymeklių atrinkimas“ (5 iliustracija), kurie yra susiejami su realiomis aplinkos koordinatėmis leidžiant įvertinti automobilio padėtį erdvėje. Kameros pateikti duomenys papildomi ultragarso sonaro duomenimis – galutinis rezultatas tiksli automobilio padėtis koordinačių sistemoje $X = [x, y, \theta]$ pateikiama fuzzy logikos kotroleriui, kuris atlieka automatinio automobilio parkavimo manevrus.

5 iliustracija. Automobilio pozicijos nustatymo algoritmo pseudo kodas



2.2. Automobilio parkavimas naudojant fuzzy logikos kontrolerį

Automatinio parkavimosi algoritmas paremtas euristiniu fuzzy logikos kontroleriu turi tris įvesties paremtrus (x, y, θ), kur x ir y yra koordinatės koordinačių sistemoje pateiktoje 4 iliustracijoje. Kampas θ yra kampas sudaromas tarp automobilio šono linijos ir koordinačių sistemos x ašies. Fuzzy logikos kontrolerio rezultas – kampas $\dot{\theta}$, kuriuo reikėtų pasukti automobilį sėkmingam parkavimui atlikti. Visas fuzzy logikos rinkinys pateiktam automatinio

parkavimosi algoritmui susideda iš 27 taisyklių pateiktų 6 iliustracijoje, kurios pagal pateiktus įvesties parametrus gražina rezultatą $\dot{\theta}$.

6 iliustracija. Fuzzy logikos taisyklių rinkinys automobilio parkavimui į kišenę

	x/y	Mažas	Didelis	Labai didelis
$\theta = Neigiamas$	**Mažas**	Teigiamu dideliu	Teigiamu dideliu	Neigiamu dideliu
	Didelis	Teigiamu vidutiniu	Teigiamu dideliu	Teigiamu dideliu
	Labai didelis	Teigiamu vidutiniu	Teigiamu dideliu	Teigiamu mažu
$\theta = Nulis$	**Mažas**	Nulis	Nulis	Neigiamas vidutinis
	Didelis	Nulis	Teigiamas didelis	Teigiamas didelis
	Labai didelis	Nulis	Teigiamas vidutinis	Teigiamas didelis
$\theta = Teigiamas$	**Mažas**	Neigiamas didelis	Nulis	Nulis
	Didelis	Neigiamas vidutinis	Nulis	Teigiamas vidutinis
	Labai didelis	Neigiamas vidutinis	Nulis	Teigiamas vidutinis

7 iliustracija. Fuzzy logikos taisyklių rinkinys automobilio parkavimui į garažą

	x/y	Mažas	Didelis	Labai didelis
$\theta = Neigiamas$	**Mažas**	Dideliu	Dideliu	Vidutiniu
	Didelis	Dideliu	Dideliu	Vidutiniu
	Labai didelis	Vidutiniu	Vidutiniu	Vidutiniu
$\theta = Nulis$	**Mažas**	Dideliu	Dideliu	Vidutiniu
	Didelis	Dideliu	Vidutiniu	Mažu
	Labai didelis	Vidutiniu	Mažu	Mažu
$\theta = Teigiamas$	**Mažas**	Vidutiniu	Vidutiniu	Mažu
	Didelis	Vidutiniu	Mažu	Nulis
	Labai didelis	Mažu	Nulis	Nulis

2.2.1. Fuzzy logikos kontrolerio panaudojimo pavyzdys

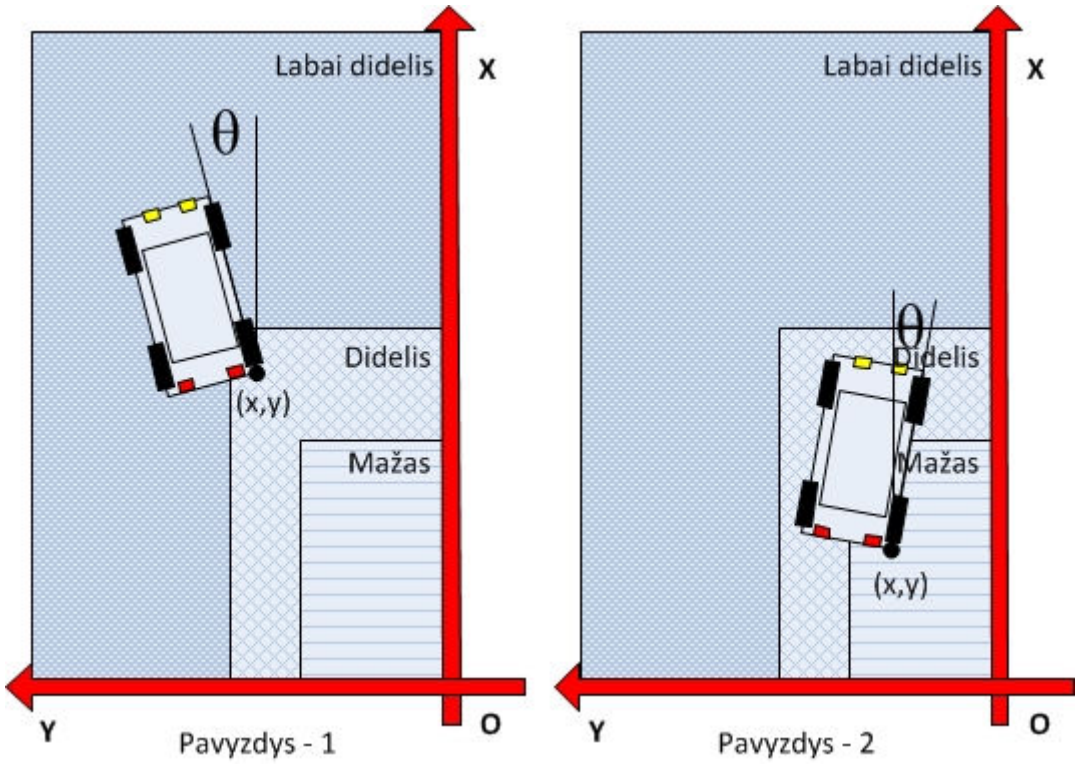
Pasitelkdami fuzzy logikos kontrolerį parkavimuisi „į kišenę“ pateiktą 6 iliustracijoje apžvelgsime du jo panaudojimo pavyzdžius:

- **Pavyzdys 1** pateiktas 8 iliustracijoje – remiantis įvesties parametrų rinkiniu x ir y koordinatės patenka į „Didelį“ parkavimosi lauką, o kampas θ sudaromas tarp x

koordinačių ašies ir automobilio šoninės linijos yra teigiamas. Todėl remiantis iliustracijoje 6 pateiktu taisyklių rinkiniu gauname rezultatą $\dot{\theta} = Nulis$ - judama toliau nekeičiant krypties į numatytą parkavimosi vietą.

- **Pavyzdys 2** pateiktas 8 iliustracijoje – remiantis įvesties parametrų rinkiniu x ir y koordinatės patenka į „Mažąjį“ parkavimosi lauką, o kampas θ sudaromas tarp x koordinačių ašies ir automobilio šoninės linijos yra neigiamas. Todėl remiantis iliustracijoje 6 pateiktu taisyklių rinkiniu gauname rezultatą $\dot{\theta} = Teigiamasdidelis$.

8 iliustracija. Automobilio automatinio parkavimosi į kišenę fuzzy logikos kontrolerio veikimo pavyzdžiai



3. Microsoft Robotics Developer Studio – platforma robototikos bandymams

Microsoft Robotics Developer Studio (MRDS) yra programinė aplinka skirta programuoti robotams Windows operacinės sistemos aplinkoje. Tai įrankis leidžiantis išvengti painiavos kuriant robotus ir jiems skirtas aplikacijas; įrankis suteikia galimybę greitai ir efektyviai atlikti bandymus virtualioje aplinkoje, o rezultatą matyti grafiniu pavidalu.

Microsoft Robotics Developer Studio pagrindiniai sudarantys komponentai: konkuravimo ir koordinavimo aplinka (angl. concurrency and coordination runtime), decentralizuotų programinės įrangos servisų aplinka (angl. decentralized software services), vizuali programavimo kalba (angl. Visual Programming Language) ir grafinė simuliavimo aplinka (angl. Visual Simulation Environment).

3.1. Konkuravimo ir koordinavimo aplinka (CCR)

CCR bibliotekos teikiamas funkcionalumas padeda vykdyti lygiagrečiai (konkurentiškai) veikiančių užduočių apdorojimą ir koordinavimą bei klaidų apdorojimo funkcijas. CCR suteikia galimybę kurti programinio kodo dalis, kurios veikia nepriklausomai, o komunikavimas vyksta naudojant žinutes (angl. messages). Vykdant komunikavimą šiuo būdu žinutės (angl. messages) kaupiamos eilėje (angl. port), kuri pagal galimybes yra apdorojama žinutės gavėjo. Kiekvienas programavinio kodo segmentas gali veikti konkurentiškai ir asinchroniškai, o dėl žinučių eilių (angl. message queues) dažniausiai nėra reikalinga atlikti papildomą sichronizavimą, kas sukelia daug problemų programuotojams ir apsunkina programinio kodo realizaciją.

Taip pat dažna asichroninio programavimo problema yra klaidų apdorojimas. Nesunku atlikti klaidų apdorojimą viename kodo fragmente, tačiau, kai duomenys sukeliantys klaidą yra perduodami žinutėmis, o klaidų apdorojimo mechanizmas nežino kokia yra klaidą sukeliančių duomenų kilmė, apdorojimas tampa sudėtingas (painu rasti klaidingus duomenis teikiantį kodo fragmentą). Ši problema CCR aplinkoje sprendžiama realizuojant priežastinį ryšį (angl. causality) tarp kodo segmentų. Yra susisiejama žinutė persiunčiama iš pirmojo kodo segmento į antrąjį, antrojo į trečiąjį ir t.t. Tokiu būdu užtikrinama, kad n-tojo lygio kodo segmente įvykusią klaidą galima atsekti iki pirminio jos šaltinio.

Apibrendinant galima teigti, kad konkuravimo ir koordinavimo aplinka supaprastina asinchroniniu būdu dirbančios sistemos įeities ir išeities duomenų apdorojimą, nes programuotojui nereikia atlikti sudėtingų gijų, blokavimo operacijų ir semaforų valdymo.

3.2. Decentralizuoti programinės įrangos servisai (DSS)

Ankščiau minėta CCR aplinka suteikia galimybę programinio kodo fragmentams keistis žinutėmis vieno proceso metu. Tuo tarpu, decentralizuoti programinės įrangos servisai (DSS) išplečia CCR funkcionalumą ir leidžia keistis žinutėmis tarp procesų ar tarp procesų veikiančių skirtingoje aparatinėje įrangoje – pavyzdžiui, skirtinguose kompiuteriuose ar kompiuteryje ir fiziniame robote/robotuose.

Aplikacija sukurta decentralizuotos programinės įrangos servisu pagrindu veikia kaip paraleliai veikiančių servisų visuma. Kiekvienas DSS aplikacijai priklausantis servisas turi apibrėžtas būsenas (angl. states), žinutes (angl. messages) ir operacijas/metodus veiksmams atlikti (angl. operations). Gavęs žinutę servisas gali keisti būseną, siųsti žinutes ar informacinius pranešimus (angl. notifications) kitiems DSS aplikacijai priklausantiems servisams.

Gali atrodyti, kad CCR ir DSS aplinkos yra analogai, tačiau jų veikimo principas yra skirtingas, nors jos ir persipina. CCR aplinka apibūdina daugelio gijų atliekančių skirtingus veiksmus koordinavimą viename Windows operacinės sistemos servise (viename kompiuteryje), tuo tarpu, DSS servisų valdymas suteikia galimybę apjungti skirtingoje aparatinėje įrangoje veikiančių servisų funkcionalumą į bendrą sistemą. Pavyzdžiui, vienas kompiuteris gali vykdyti operacijas reikalingas atstumo jutiklių teikiamos informacijos apdorojimui, tuo tarpu, kitas DSS aplikacijai priklausantis kompiuteris gali vykdyti roboto manevravimo operacijas pagal ankstesnio serviso apdorotus atstumo jutiklių duomenis. Tokiu būdu gaunama paskirstyta aplikacija, kas leidžia apkrovą išdalinti tarp skirtingos aparatinės įrangos ir atskirti skirtingo pobūdžio programinę įrangą reikalingą bendram tikslui pasiekti.

3.3. Grafinė simuliavimo aplinka (VSE)

Microsoft Robotics Developer Studio pakete yra įtraukta pilna 3D simuliavimo aplinka su fizikiniu varikliu, kuris leidžia atlikti grafinę sukurto modelio vizualizaciją su artimais pasauliui fizikiniais reiškiniais (kolizijomis su kliūtimis, atstumo matavimais apšvietimu ir t.t.).

Grafinė simuliavimo aplinka (VSE) palaiko tiek vidinius (kambario), tiek išorines (lauko) aplinkos scenarijus. Standartiniame MRDS pakete VSE aplinka pateikta su 14 unikalių panaudojimo pavyzdžių, kuriuos galima pritaikyti įvairiems robotikos bandymams atlikti.

Simuliatorius ypač pasiteisina, kai yra kuriami nauji algoritmai ar dar fiziškai neegzistuojantys robotai, nes programavimo dėka VSE aplinkoje galima sukurti norimus robotus ir jų panaudojimo scenarijus. Visa tai bandyti virtualioje aplinkoje yra žymiai greičiau, nei

sukurti fizinį modelį, tuo pačiu ir ženkliai pigiau. Vienos iš MRDS pakete pateiktų VSE aplinkų vaizdas pateiktas 9 iliustracijoje.

9 iliustracija. Grafinės simuliacijos aplinkos (VSE) naudotojo sąsaja

3.4. Grafinė programavimo kalba (VPL)

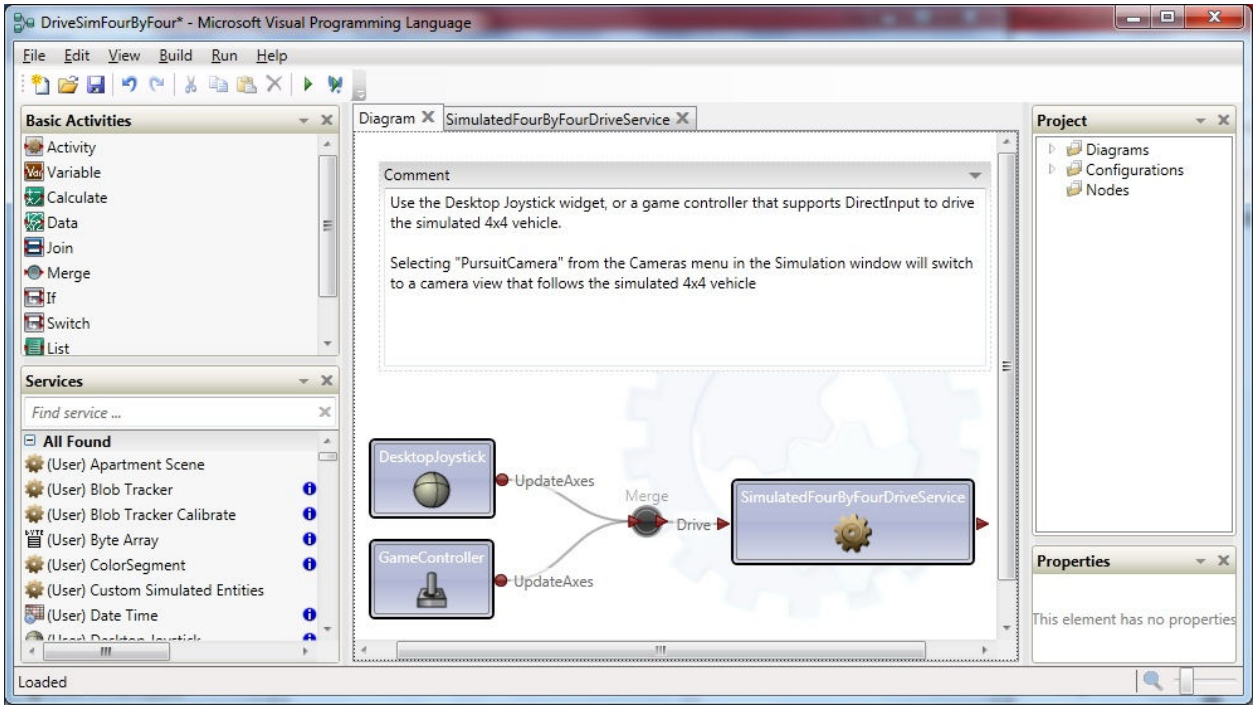
Be MRDS pakete pateikiamų CCR, DSS ir VSE aplinkų yra pateikiamas įrankis leidžiantis greitai ir efektyviai kurti robotikos aplikacijas ir simuliacijas su minimaliais programavimo veiksmais – vizuali programavimo kalba (VPL) (10 iliustracija).

Microsoft Robotics Developer Studio aplikacijos dažniausiai susideda daugiau nei iš vieno serviso. Kai kurie aplikacijas sudarančių servisų yra žemo lygio tiesiog atlieka bendravimą su aparatine įranga. Įprastinė aplikacija dažniausiai turi vieną ar daugiau servisų, kurie kontroliuoja žemo lygio servisų pateikiamą informaciją ir valdo robotą/robutus ar jų atitikmenis VSE aplinkoje. Šie aukšto lygio servisai, atliekantis „smegenų“ funkciją aplikacijoje yra vadinami instrumentavimo (angl. orchestration) servisais.

Turint standartinių ar naudotojo sukurtų servisų (objektų) rinkinį MRDS aplinkoje, bei standartinius/naudotojo sukurtus instrumentavimo servisus (pvz. prietaisų skydelius skirtus valdyti robotams) galima kurti aplikacijas nerašant nei vienos eilutės programinio kodo. Tam

yra išnaudojama Microsoft grafinė programavimo kalba (VPL). Kiekvienas blokelis 10 iliustracijoje apibūdina servisą, skaičiavimą, sąlygą ar kitą VPL diagramą. Tokiu būdu sukurtos veiksmų sekos startuojamos DSS aplinkoje, kurioje diagramoje pateikti servisai atlieka savo funkcijas, skaičiavimus ir pateikia rezultatą.

10 iliustracija. Grafinės programavimo kalbos (VPL) naudotojo sąsaja



4. Automobilio modelis su atstumo sensoriais ir kameromis MRDS platformoje

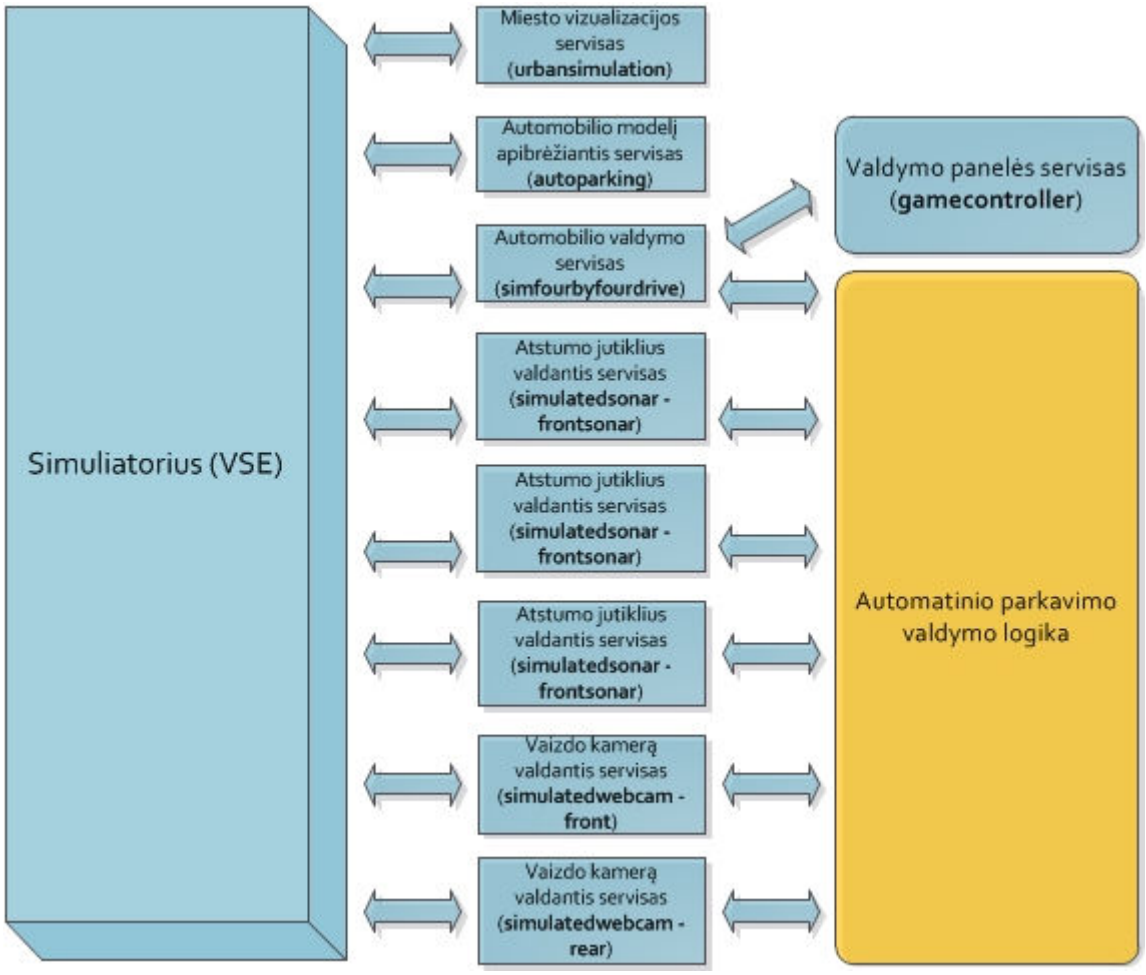
Šiame skyriuje aprašoma automobilio modelio automatinio parkavimosi algoritmams modeliuoti MRDS platformoje koncepcija. Pasirinktas automobilio modelis, tai keturiais ratais varomas automobilis turintis atstumo jutiklius (sonarus) ir vaizdo kameras. Pavyzdinis modelis kuriamas pagal skyriuje „2. Fuzzy logikos taikymas automobilio parkavimui naudojant sonarus ir vaizdo kamerą“ pateiktą algoritmą. Galutiniame rezultate MRDS platformoje sukurta klasė *JeepUrbanDemo* paveldėta nuo *VisualEntity* bazinės klasės skirtos visiems vizualiems modeliams naudojamiems VSE aplinkoje kurti. *JeepUrbanDemo* bus galima naudoti įvairiose VSE aplinkose su standartiniais valdymo kontroleriais (rankiniam valdymui) arba kurti skirtingus parkavimosi algoritmus realizuojančius valdymo/instrumentavimo (angl. orchestration) servisus. Detali automobilio modelį sudarančių komponentų visuma pateikta 11 iliustracijoje.

Schemoje pateikti melsvi komponentai yra universalūs ir jų koreguoti pritaikant modelį skirtingiems algoritmams nereikia arba pokyčiai yra minimalūs, tuo tarpu geltonai pažymėta „Automatinio parkavimosi logika“ yra unikali kiekvieno algoritmo atveju.

Plačiau apžvelgsime koncepsinį modelį sudarančių DSS servisų visumą:

- urbansimulation – miesto vizualizacijos servisas, atsakingas už grafinės miesto aplinkos su fizikiniais reiškiniais sukūrimą simuliatoriuje (VSE);
- autoparking – servisas papildantis miesto vizualizaciją automobilio modeliu *JeepUrbanDemo* ir esant poreikiui papildomais objektais;
- simfourbyfourdrive – servisas imituojantis keturiais ratais varomo automobilio valdymą VSE aplinkoje (naudojamas tiek rankiniam valdymui su programiniu/fiziniu kontroleriu, tiek automatinio parkavimosi algoritmuose);
- simulatedsonar servisai (frontsonar, rightsidefrontsonar, rightsiderearsonar) – atstumo jutiklių valdymo servisai atitinkamai: priekinis, dešinio borto priekinis ir dešinio borto galinis sonarai;
- simulatedwebcam servisai (front, rear) – automobilio kamerų centriniuose automobilio taškuose priekyje ir gale valdymo servisai;
- gamecontroller – servisas per simfourbyfourdrive valdymo servisą leidžiantis rankiniu būdu valdyti automobilį.

11 iliustracija. Automobilio modelį sudarantys komponentai



Automatinio parkavimo valdymo logika

Šis „automatinio parkavimo valdymo logikos“ servisas DSS aplinkoje apibrėžia veiksmų ir logikos sekas, kurios turi būti atliktos pasitelkus duomenis gaunamus per sensorių informaciją apdorojančius servisus. Reikia pastebėti, kad loginė dalis tiesiogiai nesąveikauja su simuliatoriumi ir vizualicijos servisais – loginė dalis atsako tik už parkavimosi algoritmo vykdymą pagal turimą sensorių informaciją. Nagrinėtų fuzzy logikos pagrindu veikiančių algoritmų atveju šis komponentas būtų fuzzy logikos kontrolerio atitikmuo.

5. Eksperimentai MRDS platformoje – automobilio modelio realizacija

Eksperimentams atlikti pasitelkta Microsoft Robotics Developer Studio 4 – naujausia šios platformos versija. Atlikti eksperimentus šioje versijoje buvo kiek keblu, nes visa turima literatūra ir joje aprašyti pavyzdžiai orientuoti į Microsoft Robotics Developer Studio 2008 R3 leidimą. Kol kas nėra išleista literatūra MRDS 4 versijai; MRDS 4 yra pakankamai naujas produktas pasaulį išvydęs šių metų kovo 8 d. (2012-03-08). Atliekant eksperimentus buvo naudojamasi literatūra „Professional Microsoft® Robotics Developer Studio“ [JT08] ir „Robot Development Using Microsoft® Robotics Developer Studio“ [SWK+11] adaptuota MRDS 2008 R3 versijai bei pavyzdžiais pateiktais įdiegtos MRDS 4 versijos kataloge *samples*.

5.1. Valdomas automobilio modelis VSE aplinkoje

Pirmasis bandymas MRDS platformoje naudojant VSE aplinką: automobilio modelio valdymas miesto gatvėmis pasitelkiant valdymo panelę.

Pasirinktam sprendimui įgyvendinti pasirinkta:

- MRDS 4 demo aplikacija *Microsoft Robotics Dev Studio 4\samples\VplExamples\DriveSimFourByFour* – automobilio modelio valdymas VSE aplinkoje;
- demonstracinė VSE aplinka *Urban Environment*.

Projektas *VPL/Examples/DriveSimFourByFour*

Pradinė projekto paskirtis naudojantis programine vairasvirte (imitatoriumi) ar žaidimams skirtu pulteliu valdyti VSE aplinkoje esantį automobilio modelį. Simuliuojama aplinka neatitinka darbo lūkesčių – automatinis parkavimas į kišenę dažniausiai atliekamas gatvėje ar aikštelėje, tuo tarpu projekte realizuota aplinka yra nelygaus paviršiaus vietovė (atviras laukas).

Demonstracinė VSE aplinka *Urban Environment*

Simuliacinė aplinka skirta vykdyti „lauko sąlygomis mieste“ sukurtus robotikos bandymus virtualioje aplinkoje. Pasirinkta dėl puikaus atitikimo parkavimosi scenarijams įgyvendinti – miesto gatvės, namai ir kt. objektai leidžia sukurti realistišką modelį.

Rezultatai

Sukurti du konfigūraciniai failai:

- *JeepUrbanDemo.Simulation.Config.xml* – miesto aplinką apibrėžiantis 3D modelio aprašas, papildytas automobilio modeliu;
- *JeepUrbanDemo.Simulation.Manifest.xml* – miesto aplinką aprašantis deklaracija/manifestas (angl. manifest) naudojamas DSS aplinkoje startuoti simuliacijos servisą.

Papildomai atlikti pakeitimai leidžiantis naudotis sukurta simuliavimo aplinka projekte VPL/Examples/DriveSimFourByFour. Išeities kodas ir demonstracija pateikta kompaktinėje plokštelėje pavadinimu - **01. Jeep Urban Demo**. Sukurtos aplinkos statinis vaizdas pateiktas 12 iliustracijoje.

12 iliustracija. Programine varasvirte valdomas automobilio modelis 3D miesto aplinkoje

5.2. Automobilio modelio praplėtimas atstumo jutikliais (sonarais)

Atliekant tolimesnius bandymus su MRDS platforma (jau turint veikiantį automobilio modelį VSE miesto aplinkoje) tapo įdomu jį papildyti atstumo jutikliais. Tai yra sukurti bazinį automobilio modelio praplėtimą, kurį galima būtų naudoti įvairiuose tolimesniuose eksperimentuose. MRDS 4 leidimas palaiko esybę *SonarEntity*, kurią panaudojame praplečiant *JeepUrbanDemo* klasę sonarų teikiamu funkcionalumu. Sonaro pridėjimas prie sukurto automobilio modelio (paveldėto nuo bazinės Microsoft Robotics klasės *VisualEntity*, atrodo taip:

```
//Add sonar: front center
SonarEntity frontSonar = new SonarEntity(new Pose(new Vector3(0,
        ChassisDimensions.Y / 2.0f + ChassisClearance - 0.01f,
        -ChassisDimensions.Z / 2.0f),
   TypeConversion.FromXNA(
       xna.Quaternion.CreateFromAxisAngle(
           new xna.Vector3(0, 1, 0), (float)Math.PI))));
frontSonar.State.Name = this.State.Name + "_FrontSonar";
InsertEntityGlobal(frontSonar);
```

5.3. Automobilio modelio praplėtimas vaizdo kameromis

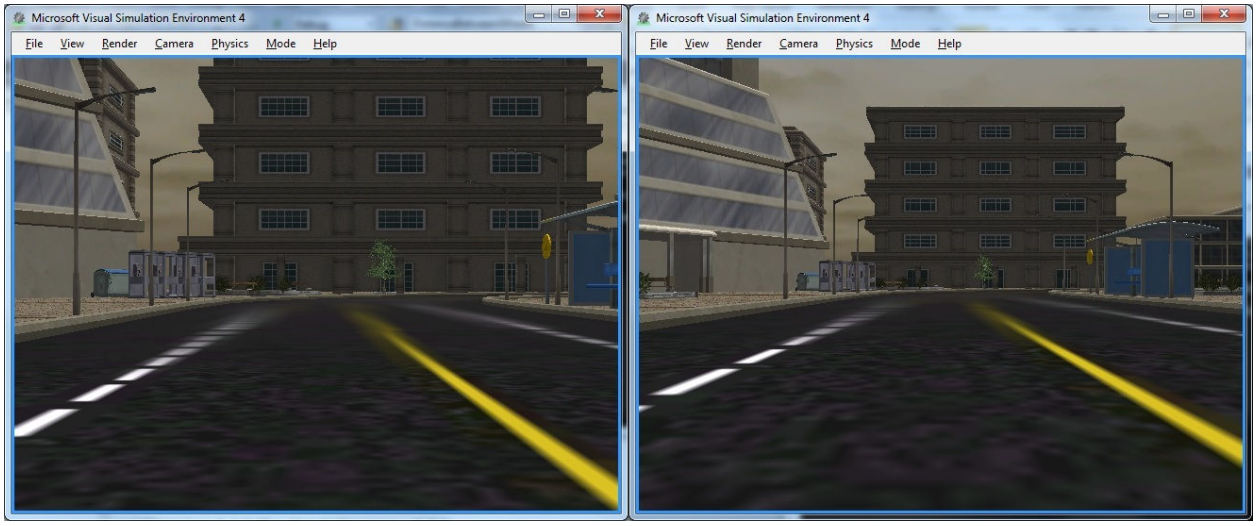
Atsižvelgiant į šiuolaikiniuose automobiliuose taikomas technologijas, automobilio modelį papildome vaizdo kameromis automobilio priekyje ir gale. MRDS platforma turi standartinį vaizdo kameros komponentą *CameraEntity*, kurį „primontuojame“ prie automobilio modelio. C# programinio kodo fragmentas galinės vaizdo kameros „primontavimui“:

```
//Add camera rear
CameraEntity rearCam = new CameraEntity(
    640,
    480,
    (float)(30.0 * Math.PI / 180.0));
rearCam.State.Name = this.State.Name + "_RearCam";
rearCam.IsRealTimeCamera = true;
rearCam.State.Pose.Position = new Vector3(
    0,
    ChassisDimensions.Y / 2.0f + ChassisClearance - 0.01f,
    ChassisDimensions.Z / 2.0f + 0.065f);
rearCam.Rotation = new xna.Vector3(0, 180f, 0);
InsertEntityGlobal(rearCam);
```

Reikia pastebėti, kad šiame mažame programinio kodo fragmente yra keletas svarbių vietų:

- konstruktoriuje nustatoma kameros rezoliucija yra 640x480 pikselių – šis dydis pasirinktas atsižvelgiant į tai, kad Microsoft Kinect sensorius naudoja minėtą rezoliuciją ir to pakanka objektams erdvėje identifikuoti;
- trečias parametras *CameraEntity* konstruktoriuje apibrėžia kameros matymo kampą, kas yra svarbu siekiant užtikrinti kuo didesnį erdvės matomumą (pavyzdžiai kampams 30 ir 180 laipsnių pateikiami iliustracijoje);
- nustatymas rearCam.Rotation leidžia kamerą modelyje apsukti norima kryptimi, konkrečiu atveju – atgal.

13 iliustracija. Objekto *CameraEntity* matymo kampo palyginimas



5.4. Modelio konfigūravimas

Skyriuje „4. Automobilio modelis su atstumo sensoriais ir kameromis MRDS platformoje“ aprašyto koncepsinio modelio konfigūravimas Microsoft Robotics Developer Studio aplinkoje atliekamas XML manifestų dėka. Vienas paprasčiausių servisų konfigūravimo pavyzdžių yra grafinė roboto valdymo konsolė, kurios konfigūravimo aprašas XML formatu atrodo taip:

```
<ServiceRecordType>
       <dssp:Contract>http://schemas.microsoft.com/robotics/2006/01/simpledashboard.html<
/dssp:Contract>
       <dssp:PartnerList />
       <Name>this:SimpleDashboard</Name>
</ServiceRecordType>
```

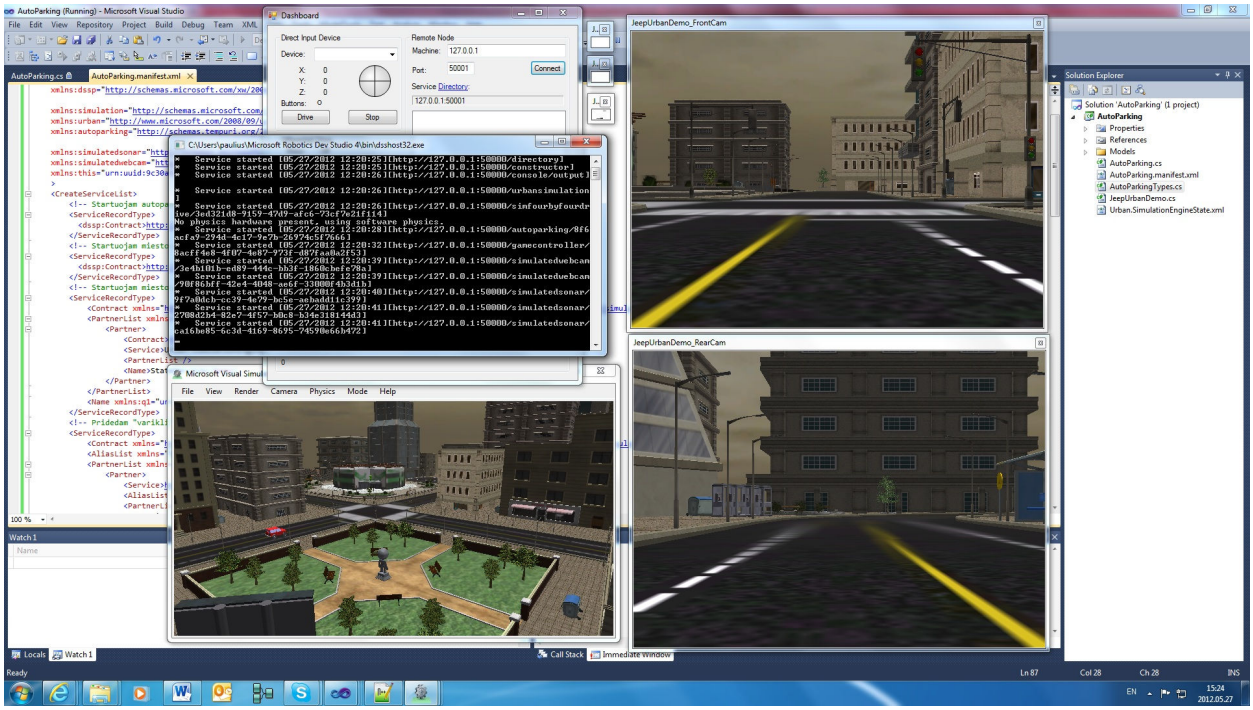
Konfigūravimo pavyzdyje nurodomas komponentą (šiuo atvejų valdymo panelę) apibrėžiantis kontraktas bei nurodomas konsolę atitinkančio serviso vardas DSS platformoje („SimpleDashboard“). Pilnas sukurto modelio konfigūravimo pavyzdys pateikiamas priede - „Priedas Nr. 2.“.

Rezultatai

Darbo metu pasiekti rezultatai:

- išanalizuoti du fuzzy logikos principu veikiantys automobilio parkavimo algoritmai;
- išmokta naudotis naujausia Microsoft Robotics Developer 4 programinės įrangos versija;
- MRDS platformoje sukurtas **01. Jeep Urban Demo** modelis, leido suprasti VPL, VSE ir DSS aplinkų darbą;
- sukurta *JeepUrbanDemo* klasė realizuoja VSE aplinkai skirtą automobilio modelį praplėstą sonarais bei vaizdo kameromis;
- sukurtas *JeepUrbanDemo* panaudojimo pavyzdys – servisų visuma sudaryta iš sonarų bei kamerų valdymo servisų, valdymo panelės ir kitų servisų veikiančių DSS aplinkoje (14 iliustracija); šis pavyzdys/modelis gali būti naudojamas automatinio parkavimo algoritmams bandyti.

14 iliustracija. Sukurto modelio DSS konsolė, valdymo panelė ir kamerų teikiami vaizdai



Išvados

Remiantis apžvelgtais eksperimentais galima teigti, kad parkavimosi problema yra daug sudėtingesnė praktikoje, nei brėžiniuose ir tiksliuose matematiniuose modeliuose. Užduotis tampa daug sudėtingesnė dėl sensorių paklaidų bei kitų automobilių savybių, kurios nėra apibrėžiamos paprastame matematiniame modelyje (pvz., algoritme pateiktame skyriuje „Fuzzy logikos taikymas automobilio parkavimui naudojantis sonarais“).

Pagal atliktą algoritmų analizę pilnai automatizuotas automobilio parkavimas reikalauja didesnio tikslumo pozicijos erdvėje nustatymo metu, nei gali užtikrinti sonarai. Šiai problemai spręsti reikia pasitelkti papildomus sensorius, pvz. kamerą.

Nepriimtini sprendimai naudojantys automobilio greičio ir nuvažiuoto kelio parametrus, nes taip ženkliai „pabrangsta“ techninė algoritmo realizacija – duomenų gavimas iš automobilio.

MRDS platformai sukurtas automobilio modelis pasitelkiant sonarus bei vaizdo kameras yra tinkamas tolimesniems parkavimosi algoritmų bandymams ir analizei. Modelis lengvai plečiamas ir tobulinamas. Šiuo metu pagrindinė problema su kurią susiduriama dirbant su MRDS 4 platforma yra einamajai versijai adaptuotos literatūros trūkumas. Neseniai pasirodęs MRDS 4 leidimas neturi oficialios literatūros, todėl teko remtis MRDS 2008 R3 platformai skirta informacija, kuri, kad ir panaši, tačiau turi neatitikimų.

Literatūros sąrašas

[HP96] R. Holve, P. Protzel. Reverse Parking of a Model Car with Fuzzy Control. Proceedings of the 4th European Congress on Intellingent Techniques and Soft Computing – EUFIT `96. Aachen, Germany, Sept. 1996, p.p. 2171-2175.

[YSS08] Young-Woo Ryu Se-Young Oh and Sam-Yong Kim. Robust Automatic Parking without Odometry using Evolutionary Fuzzy Logic Controller. International Journal of Control, Automation,.and Systems, vol. 6, no. 3, pp. 434-443, June 2008.

[JT08] Kyle Johns, Trevor Taylor. Professional Microsoft® Robotics Developer Studio. Wiley Publishing Inc., 2008, Indianapolis.

[SWK+11] Shih-Chung Kang, Wei-Tze Chang, Kai-Yuan Gu, Hung-Lin Chi. Robot Development Using Microsoft® Robotics Developer Studio. CRC Press Taylor & Francis Group, 2011.

Priedas Nr. 1.

MRDS 4 aplinkos paruošimas eksperimentams

Naudota techninė/programinė įranga

Parametras	**Reikšmė**
Kompiuteris	HP Compaq dx2400 Microtower
Procesorius (CPU)	Intel® Core 2 Quad CPU Q9300 @ 2.50 GHz
Operatyvioji atmintis (RAM)	4 GB
Kietasis diskas (HDD)	250 GB
Video kontroleris	Intel® G33/G31 Express Chipset Family
Operacinė sistema (OS)	Windows 7 Enterprise 64-bit
Monitorius	HP 2159m (1920x1080 full HD)

MRDS 4 diegimas

Programinė įranga	**Komentaras**
Programavimo platforma Visual Studio 2010	MRDS 4 suderinamas su Visual Studio 2010, konkrečiu atveju naudota Microsoft Visual Studio 2010 Premium versija C# kodo modifikavimui.
Kinect sensorių valdymo platforma	Įdiegta Kinect for Windows SDK V1 (pagal diegimo rekomendacijas). Šio darbo eigoje Kinect for Windows SDK V1 funkcionalumas nenaudotas.
MRDS integracija su Silverlight	Silverlight 4.0 SDK
MRDS 4 platforma	Įdiegtas pilnas Microsoft Robotics Developer Studio 4 paketas

Aprašytą programinę įrangą (išskyrus Microsoft Visual Studio 2010 Premiumir Windows 7 OS) pridedama kompaktinėje plokštelėje kataloge šakniniame *Soft*. Diegimo eiliškumas atitinka pateiktą MRDS 4 diegimo lentelėje.

Pastaba. MRDS platforma įdiegiama šakniniame Windows 7 naudotojo kataloge. Pavyzdžiui *C:\Users\Naudotojas\Microsoft Robotics Dev Studio 4*

Priedas Nr. 2.

Automobilio modelio aprašyto skyriuje „2. Fuzzy logikos taikymas automobilio parkavimui naudojant sonarus ir vaizdo kamerą“ konfigūracijos aprašas (be instrumentavimo):

```
<?xml version="1.0" ?>
<Manifest
    xmlns="http://schemas.microsoft.com/xw/2004/10/manifest.html"
    xmlns:dssp="http://schemas.microsoft.com/xw/2004/10/dssp.html"

    xmlns:simulation="http://schemas.microsoft.com/robotics/2006/04/simulation.html"
    xmlns:urban="http://www.microsoft.com/2008/09/urban.html"
    xmlns:autoparking="http://schemas.tempuri.org/2012/05/autoparking.html"


xmlns:simulatedsonar="http://schemas.microsoft.com/robotics/simulation/services/2006/05/s
imulatedsonar.html"
    xmlns:simulatedwebcam="http://schemas.microsoft.com/2006/09/simulatedwebcam.html"
    xmlns:this="urn:uuid:9c30a5ed-b078-43a3-b99e-2773d293c707"
    >
    <CreateServiceList>
        <!-- Startuojam autoparking aplinką-->
        <ServiceRecordType>

<dssp:Contract>http://schemas.tempuri.org/2012/05/autoparking.html</dssp:Contract>
        </ServiceRecordType>
        <!-- Startuojam miesto aplinką -->
        <ServiceRecordType>
          <dssp:Contract>http://www.microsoft.com/2008/09/urban.html</dssp:Contract>
        </ServiceRecordType>
        <!-- Startuojam miesto planą-->
        <ServiceRecordType>
            <Contract
xmlns="http://schemas.microsoft.com/xw/2004/10/dssp.html">http://schemas.microsoft.com/ro
botics/2006/04/simulationengine.html</Contract>
            <PartnerList xmlns="http://schemas.microsoft.com/xw/2004/10/dssp.html">
                <Partner>

<Contract>http://schemas.microsoft.com/robotics/2006/04/simulationengine.html</Contract>
                    <Service>Urban.SimulationEngineState.xml</Service>
                    <PartnerList />
                    <Name>StateService</Name>
                </Partner>
            </PartnerList>
            <Name xmlns:q1="urn:uuid:fe27ba25-2680-4932-958c-
2aae3bde3611">q1:SimulationEngine</Name>
        </ServiceRecordType>
        <!-- Pridedam "variklį" ir "vairą"-->
        <ServiceRecordType>
            <Contract
xmlns="http://schemas.microsoft.com/xw/2004/10/dssp.html">http://schemas.microsoft.com/ro
botics/2007/10/simulatedfourbyfourdrive.html</Contract>
            <AliasList xmlns="http://schemas.microsoft.com/xw/2004/10/dssp.html" />
            <PartnerList xmlns="http://schemas.microsoft.com/xw/2004/10/dssp.html">
                <Partner>
                    <Service>http://localhost/JeepUrbanDemo</Service>
                    <AliasList />
                    <PartnerList />
                    <Name>simulation:Entity</Name>
                </Partner>
```

```
            </PartnerList>
            <Name xmlns:q3="urn:uuid:fe27ba25-2680-4932-958c-
2aae3bde3611">q3:SimulatedFourByFourDriveService</Name>
        </ServiceRecordType>
        <!-- Simulated web cam (FRONT CAM)-->
        <ServiceRecordType>

<dssp:Contract>http://schemas.microsoft.com/2006/09/simulatedwebcam.html</dssp:Contract>
            <dssp:PartnerList>
                <dssp:Partner>
                    <dssp:Service>http://localhost/JeepUrbanDemo_FrontCam</dssp:Service>
                    <dssp:PartnerList />
                    <dssp:Name>simulation:Entity</dssp:Name>
                </dssp:Partner>
            </dssp:PartnerList>
            <Name>this:simulatedwebcamfront</Name>
        </ServiceRecordType>
        <!-- Simulated web cam (REAR CAM)-->
        <ServiceRecordType>

<dssp:Contract>http://schemas.microsoft.com/2006/09/simulatedwebcam.html</dssp:Contract>
            <dssp:PartnerList>
                <dssp:Partner>
                    <dssp:Service>http://localhost/JeepUrbanDemo_RearCam</dssp:Service>
                    <dssp:PartnerList />
                    <dssp:Name>simulation:Entity</dssp:Name>
                </dssp:Partner>
            </dssp:PartnerList>
            <Name>this:simulatedwebcamrear</Name>
        </ServiceRecordType>
        <!-- Simulated sonar (FRONT SONAR) -->
        <ServiceRecordType>

<dssp:Contract>http://schemas.microsoft.com/robotics/simulation/services/2006/05/simulate
dsonar.html</dssp:Contract>
            <dssp:PartnerList>
                <dssp:Partner>
                  <dssp:Service>http://localhost/JeepUrbanDemo_FrontSonar</dssp:Service>
                  <dssp:PartnerList />
                  <dssp:Name>simulation:Entity</dssp:Name>
                </dssp:Partner>
            </dssp:PartnerList>
            <Name>this:simulatedsonar_front</Name>
        </ServiceRecordType>
        <!-- Simulated sonar (RIGHT SIDE FRONT SONAR) -->
        <ServiceRecordType>

<dssp:Contract>http://schemas.microsoft.com/robotics/simulation/services/2006/05/simulate
dsonar.html</dssp:Contract>
            <dssp:PartnerList>
                <dssp:Partner>

<dssp:Service>http://localhost/JeepUrbanDemo_RightSideFrontSonar</dssp:Service>
                    <dssp:PartnerList />
                    <dssp:Name>simulation:Entity</dssp:Name>
                </dssp:Partner>
            </dssp:PartnerList>
            <Name>this:simulatedsonar_rightsidefrontsonar</Name>
        </ServiceRecordType>
        <!-- Simulated sonar (RIGHT SIDE REAR SONAR) -->
        <ServiceRecordType>
```

```
<dssp:Contract>http://schemas.microsoft.com/robotics/simulation/services/2006/05/simulate
dsonar.html</dssp:Contract>
            <dssp:PartnerList>
                <dssp:Partner>

<dssp:Service>http://localhost/JeepUrbanDemo_RightSideRearSonar</dssp:Service>
                    <dssp:PartnerList />
                    <dssp:Name>simulation:Entity</dssp:Name>
                </dssp:Partner>
            </dssp:PartnerList>
            <Name>this:simulatedsonar_rightsiderearsonar</Name>
        </ServiceRecordType>
        <!-- Startuojam valdymo panelę -->
        <ServiceRecordType>

<dssp:Contract>http://schemas.microsoft.com/robotics/2006/01/simpledashboard.html</dssp:C
ontract>
            <dssp:PartnerList />
            <Name>this:SimpleDashboard</Name>
        </ServiceRecordType>
    </CreateServiceList>
</Manifest>
```