**Kauno technologijos universitetas**

Informatikos fakultetas

# Savavaldžio roboto algoritmų tyrimas supaprastintoje aplinkoje

Baigiamasis magistro projektas

**Jurgis Kišūnas**

Projekto autorius

**Doc. Mantas Lukoševičius**

Vadovas

**Kaunas, 2021**


**Kauno technologijos universitetas**

Informatikos fakultetas

# Savavaldžio roboto algoritmų tyrimas supaprastintoje aplinkoje

Baigiamasis magistro projektas

Programų sistemų inžinerija (6211BX011)

**Jurgis Kišūnas**

Projekto autorius

**Doc. Mantas Lukoševičius**

Vadovas

**Prof. Evaldas Vaičiukynas**

Recenzentas

**Kaunas, 2021**

**Kauno technologijos universitetas**

Informatikos fakultetas

Jurgis Kišūnas

# Savavaldžio roboto algoritmų tyrimas supaprastintoje aplinkoje

Akademinio sąžiningumo deklaracija

Patvirtinu, kad:

1. baigiamąjį projektą parengiau savarankiškai ir sąžiningai, nepažeisdama(s) kitų asmenų autoriaus ar kitų teisių, laikydamasi(s) Lietuvos Respublikos autorių teisių ir gretutinių teisių įstatymo nuostatų, Kauno technologijos universiteto (toliau – Universitetas) intelektinės nuosavybės valdymo ir perdavimo nuostatų bei Universiteto akademinės etikos kodekse nustatytų etikos reikalavimų;

2. baigiamajame projekte visi pateikti duomenys ir tyrimų rezultatai yra teisingi ir gauti teisėtai, nei viena šio projekto dalis nėra plagijuota nuo jokių spausdintinių ar elektroninių šaltinių, visos baigiamojo projekto tekste pateiktos citatos ir nuorodos yra nurodytos literatūros sąraše;

3. įstatymų nenumatytų piniginių sumų už baigiamąjį projektą ar jo dalis niekam nesu mokėjęs (-usi);

4. suprantu, kad išaiškėjus nesąžiningumo ar kitų asmenų teisių pažeidimo faktui, man bus taikomos akademinės nuobaudos pagal Universitete galiojančią tvarką ir būsiu pašalinta(s) iš Universiteto, o baigiamasis projektas gali būti pateiktas Akademinės etikos ir procedūrų kontrolieriaus tarnybai nagrinėjant galimą akademinės etikos pažeidimą.

Jurgis Kišūnas

*Patvirtinta elektroniniu būdu*

Kišūnas, Jurgis. Savavaldžio roboto algoritmų tyrimas supaprastintoje aplinkoje. Magistro baigiamasis projektas vadovas Doc. Manas Lukoševičius; Kauno technologijos universitetas, informatikos fakultetas.

Studijų kryptis ir sritis (studijų krypčių grupė): informatika.

Reikšminiai žodžiai: dirbtinis intelektas, mašininis mokymas, savavaldės transporto priemonės, DuckieTown.

Kaunas, 2021. 56 p.

## Santrauka

Tradicines transporto priemones keitimo savavaldėmis linkme siekiama, kad savavaldis robotas ar transporto priemonė supaprastintoje aplinkoje, be žmogaus pagalbos, galėtų judėti kelyje, pats atlikdamas reikalingus sprendimus ir nepažeisdamas taisyklių. Autonominė transporto priemonė, keliaudama pagal nustatytas taisykles, turėtų važiuoti kuo ilgiau nesuklydusi, nepadariusi nei vienos kritinės klaidos. Savavaldėms transporto priemonėms dar nepaplitus keliuose, šio darbo tikslas nėra realizuoti realios transporto priemonės judėjimą, o tik sukurti prototipą, kaip tai galėtų vykti, jei aplinka būtų tokia paprasta ir nuspėjama kaip atliekant modeliavimą. Turint tokią informaciją, ją galima pritaikyti ir prie realių sąlygų.

Darbo pagrindinis uždavinys yra supaprastintoje aplinkoje valdomam robotui, sukurti pozicijos išskyrimo ir roboto valdymo algoritmus, pasitelkiant dirbtinį intelekto ir matematinius modelius. Planuojama nagrinėti savavaldžių transporto priemonių judėjimo problemas, jų aplinką ir kitus su valdymu susijusius veiksnius. Sukurta sistema gebės važiuoti eismo juostoje, neišvažiuojant iš kelio, minimizavus važiavimo klaidą. Tokios sistemos kūrimo eigoje atliekami tyrimai, atskleidžiantys koreliacijas tarp skirtingų valdymo modelių ir pozicijos aptikimo algoritmų.

Kišūnas, Jurgis. The research of an autonomously controlled robot’s algorithm in a simplified environment. Master's Final Degree Project supervisor Doc. Mantas Lukoševičius; Faculty of Informatics, Kaunas University of Technology.

Study field and area (study field group): Computer science

Keywords: artificial intelligence, machine learning, autonomous vehicles, DuckieTown, autonomy, self-driving vehicle, autonomous movement of a robot, image processing.

Kaunas, 2021. 56 pages.

## Summary

Towards the replacement of traditional vehicles by autonomous systems, the aim is to enable an autonomous robot or vehicle to move on the road in a simplified environment, without human assistance, by making the necessary decisions itself and without violating the rules. An autonomous vehicle should travel in accordance with the established rules for as long as possible without making a critical mistake. Before self-driving vehicles are yet widespread, the aim of this work is not to realize the movement of a real vehicle, but only to create a prototype of how this could happen if the environment were as simple and predictable as in simplified simulation. With such information, it can be adapted to real conditions.

The main task of the work is for a robot controlled in a simplified environment to develop algorithms for position detection and robot control using artificial intelligence and mathematical models. It is planned to study the problems of the movement of autonomous vehicles, their environment and other factors related to controlling the vehicle. The developed system will be able to drive in the traffic lane without leaving the road by minimizing the driving error. While developing such a system, research is conducted that reveals the correlations between different control models and position detection algorithms.

**Turinys**

**Lentelių sąrašas** ... **8**
**Paveikslų sąrašas** ... **9**
**Santrumpų ir terminų sąrašas** ... **10**
**Įžanga** ... **11**
**1. Autonominių transporto priemonių analizė** ... **12**
1.1. Tikslas ... 12
1.2. Poreikis ... 12
1.3. Savavaldės sistemos ... 13
1.4. Rinkos tyrimas ... 14
1.5. Produktas ... 14
1.5.1. Sistemos komponentai ... 14
1.5.2. Vaizdų atpažinimas, segmentacija ... 15
1.5.3. Aplinka „DuckieTown“ ... 17
1.6. Produktas ... 19
1.6.1. Programinė įranga ... 21
1.6.2. Techninė įranga ... 22
1.7. Įgyvendinimo problemos ... 22
1.8. Konkurencija ir alternatyvos ... 22
**2. Autonominio roboto valdomo supaprastintoje aplinkoje projektas** ... **24**
2.1. Sistemos tikslai (paskirtis) ... 24
2.2. Apribojimai ... 24
2.3. Diegimo aplinka ... 24
2.4. Veiklos kontekstas (pateikiama konteksto diagrama) ... 25
2.5. Sistemos ribos ... 26
2.5.1. Panaudojimo atvejų sąrašas ... 26
2.6. Funkciniai reikalavimai ir reikalavimai duomenims ... 28
2.6.1. Funkciniai reikalavimai ... 28
2.6.2. Nefunkciniai reikalavimai ... 34
2.7. Paketų diagrama ... 38
2.8. Sistemos būsenos ... 39
2.9. Sistemos duomenys ... 39
**3. Autonominių transporto priemonių valdymo tyrimas** ... **41**
3.1. Tyrimo tikslas ... 41
3.2. Pozicijos aptikimas ... 41
3.2.1. Tyrimo aprašymas ... 41
3.2.2. Duomenų rinkimas ... 42
3.2.3. Modelių apmokymas ... 42
3.2.4. Modelių palyginimas ... 45
3.3. Roboto valdymas ... 46
3.3.1. Tyrimo aprašymas ... 46
3.3.2. Parametrų optimizavimas ... 47
3.3.3. Metodų palyginimas ... 48
**4. Roboto valdymo aptinkant jo poziciją kelyje eksperimentas** ... **49**
4.1. Eksperimento tikslas ... 49

4.2. Eksperimento aprašymas ..... 49
4.3. Eksperimentas ir jo rezultatai ..... 49
**5. Tobulinimas, galimi tolimi darbai ..... 52**
5.1. Progresas simuliatoriuje ..... 52
5.1.1. Pozicijos aptikimo tobulinimas ..... 52
5.1.2. Roboto valdymo tobulinimas ..... 52
5.1.3. Papildomos taisyklės ..... 52
5.1.4. Navigacija ..... 52
5.2. Reali aplinka ..... 53
**Išvados ..... 54**
**Literatūros sąrašas ..... 55**

## Lentelių sąrašas

**2.3.1 lentelė.** Autonomijos lygiai (Copyright © 2014 SAE International) ..... 13
**2.6.1 lentelė** Darbe naudojama programinė įranga ..... 21
**2.6.2 lentelė** Kompiuterio aprašymas ..... 22
**2.8.1 lentelė.** Savivaldžių ir žmogaus valdomų transporto priemonių palyginimas ..... 22
**3.5.1 lentelė** PA “Priimti sprendimą” ..... 26
**3.5.2 lentelė** PA “Gauti vaizdą” ..... 26
**3.5.3 lentelė** PA “Važiuoti tiesiai” ..... 27
**3.5.4 lentelė** PA “Važiuoti į kairę” ..... 27
**3.5.5 lentelė** PA “Važiuoti į dešinę” ..... 27
**3.5.6 lentelė** PA “Važiuoti atgal” ..... 27
**3.5.7 lentelė** PA “Keisti sistemos parametrus” ..... 27
**3.5.8 lentelė** PA “Matyti sistemos apdorotą vaizdą” ..... 27
**3.5.9 lentelė** PA “Matyti sistemos sprendimų išvestį” ..... 28
**3.5.10 lentelė** PA “Paleisti sistemą” ..... 28
**3.5.11 lentelė** PA “Stabdyti sistemą” ..... 28
**3.6.1 lentelė** Funkcinis reikalavimas nr. 1 ..... 28
**3.6.2 lentelė** Funkcinis reikalavimas nr. 2 ..... 29
**3.6.3 lentelė** Funkcinis reikalavimas nr. 3 ..... 29
**3.6.4 lentelė** Funkcinis reikalavimas nr. 4 ..... 30
**3.6.5 lentelė** Funkcinis reikalavimas nr. 5 ..... 30
**3.6.6 lentelė** Funkcinis reikalavimas nr. 6 ..... 31
**3.6.7 lentelė** Funkcinis reikalavimas nr. 7 ..... 31
**3.6.8 lentelė** Funkcinis reikalavimas nr. 8 ..... 32
**3.6.9 lentelė** Funkcinis reikalavimas nr. 9 ..... 32
**3.6.10 lentelė** Funkcinis reikalavimas nr. 10 ..... 33
**3.6.11 lentelė** Nefunkcinis reikalavimas nr. 12 ..... 34
**3.6.12 lentelė** Nefunkcinis reikalavimas nr. 13 ..... 34
**3.6.13 lentelė** Nefunkcinis reikalavimas nr. 14 ..... 35
**3.6.14 lentelė** Nefunkcinis reikalavimas nr. 15 ..... 35
**3.6.15 lentelė** Nefunkcinis reikalavimas nr. 16 ..... 36
**3.6.16 lentelė** Nefunkcinis reikalavimas nr. 17 ..... 36
**3.6.17 lentelė** Nefunkcinis reikalavimas nr. 18 ..... 37
**3.6.18 lentelė** Nefunkcinis reikalavimas nr. 19 ..... 37
**4.2.1 lentelė** Mašininio mokymo modelio (Nr. 1) savybės ..... 43
**4.2.2 lentelė** Mašininio mokymo modelio (Nr. 2) savybės ..... 45
**4.3.1 lentelė** Mažiausios modelio klaidos ..... 48
**5.2.1 lentelė** Pozicijos aptikimo ir roboto valdymo metodų grupės ..... 49
**5.3.1 lentelė** Grupių 1, 2 ir 3 bandymai ..... 49
**5.3.2 lentelė** Grupių 3, 4 ir 5 bandymai ..... 50
**5.3.3 lentelė** Tyrimo ir eksperimento duomenų palyginimas ..... 51

## Paveikslų sąrašas

**2.5.1 pav.** OpenCv kalibravimo šablonas su skaičiavimo dalių žymėjimais [18] ..... 16
**2.5.2 pav.** DuckieTown fizinė aplinka ir DuckietBot [21] ..... 17
**2.5.3 pav.** Savavaldžio robotuko komponentai [22] ..... 18
**2.5.4 pav.** Grindų sluoksnis [22] ..... 18
**2.5.5 pav.** Ženklų sluoksnis [22] ..... 19
**2.6.1 pav.** Galutinis produktas ..... 20
**2.6.2 pav.** Konteksto diagrama ..... 21
**3.3.1 pav.** Virtuali diegimo aplinka ..... 25
**3.4.1 pav.** Veiklos kontekstas ..... 25
**3.5.1 pav.** Panaudojimo atvejų diagrama ..... 26
**3.7.1 pav.** Paketų diagrama ..... 39
**3.8.1 pav.** Sistemos būsenos diagrama ..... 39
**3.9.1 pav.** Sistemos esybių ryšių modelis ..... 40
**4.2.1 pav.** Roboto pozicija kelyje [23] ..... 41
**4.2.2 pav.** Mašininio mokymo modelio (Nr. 1) architektūra ..... 43
**4.2.3 pav.** Mašininio mokymo modelio (Nr. 1) treniravimo rezultatų diagrama ..... 43
**4.2.4 pav.** Mašininio mokymo modelio (Nr. 2) architektūra ..... 44
**4.2.5 pav.** Mašininio mokymo modelio (Nr. 2) treniravimo rezultatų diagrama ..... 45
**4.3.1 pav.** PID reguliatorius ..... 46
**4.3.2 pav.** P, PI ir PID modelių klaidos diagrama ..... 47
**5.3.1 pav.** Pozicijos aptikimo ir valdymo paklaidos bandymų metu ..... 50

## Santrumpų ir terminų sąrašas

**Santrumpos:**

**TP** – Transporto Priemonė.

**Terminai:**

**Autonomija** – savivalda, savarankiškumas, teisė pačiam priimti sprendimus ir pagal juos save valdyti.
**Savavaldis** – savarankiškas, nepriklausomas nuo kitų.
**Padėties koduotuvas** – (angl. encoder) prietaisas konvertuojantis informaciją iš vieno formato į kitą. Padėties koduotuvas verčia kokio nors objekto judėjimą į tam tikrą kiekį impulsų, pagal tai galima nuspręsti kiek tas objektas pajudėjo.
**Lokacija** – objekto aptikimas ir vietos koordinatėmis nustatymas.
**DuckieTown** – supaprastina aplinka savavaldžio roboto tyrimui.
**Klasifikavimas** – įvairiarūšių objektų suskirstymas pagal požymius.
**Simuliatorius** – virtuali supaprastinta aplinka su robotu gebančiu joje važinėti.
**Repozitorija** – kodo talpykla.
**PID reguliatorius** - reguliatorius su grįžtamuoju ryšiu. Parametriškai optimizuojamas per tris grandis: proporcinę (P), integralinė (I) ir diferencijuojančią (D).

# Įžanga

Šis dokumentas skirtas supažindinimui su magistrinio darbo tema ir atliktu tyrimu, bei eksperimentu. Tai akademinis baigiamojo magistrantūros studijų projektas aprašantis savavaldžių transporto priemonių valdymo problemų bei jų sprendžiantis jas pasiteikiant dirbtinį intelektą.

Nenumaldomas noras tobulėti ir galimai tingumas žmones veda link technologinio tobulėjimo. Kasdienių žmonių problemų ar galimybių trūkumo sprendimas – autonominės transporto priemonės. Turint neįgalumą, nemokant vairuoti, esant per jaunam ar per senam vis vien reikia nusigauti iš taško A į tašką B. Tradicinės transporto priemonės tokiais atvejais netinka. Šio baigiamojo projekto darbo metu siekiama, kad, kad savavaldis robotas ar transporto priemonė supaprastintoje aplinkoje, be žmogaus pagalbos, galėtų judėti kelyje, pats atlikdamas reikalingus sprendimus ir nepažeisdamas taisyklių.

Darbe apžvelgiami skirtingi metodai kurie pritaikomi įvairiausių autonominio valdymo problemų sprendimui. Naudojami konvoliuciniai neuroniniai tinklai siekiant išskirti roboto poziciją kelyje. Morfologinis vaizdų apdorojimas reikalingas pirminiam nuotraukų apdorojimui. Sukaupiama galybė duomenų, kurie vėliau apdorojami ir patiekiami mašininio mokymo modeliui. Valdymui naudojamas matematinis proporcinis, integralinis, diferencialinis modelis, kuris padeda minimizuoti roboto nuokrypį nuo norimos trajektorijos, bei mažina vairavimo klaidą. Darbo gale eksperimento metu tiriama koreliacija tarp minėtų metodų.

# 1. Autonominių transporto priemonių analizė

## 1.1. Tikslas

Savavaldžiai robotai bei automatizacija sparčiai augo pastarajame dešimtmetyje, taip padėdama visuomenei. Kiekviena autonomiškai besivaldanti transporto priemonė gali taupyti kelionės laiką, žinodama eismą ir efektyviau bendraudama su kitomis transporto priemonėmis. Taip pat gali smarkiai sumažinti spūsčių kiekį, taip tausojant gamtą ir mažinant kelionių kaštus. Kelionės tokiu automobiliu taptų pasiekiamos visiems, nepriklausomai nuo jų amžiaus, sveikatos, nuovargio bei kitų sutrikimų ar būsenos [1, 2]. Tai kol kas gali atrodyti neįtikinamai ir nepasiekiamai, tačiau yra priešingai. Universitetuose jau dabar mokomi kursai apie tokių sistemų kūrimą, kas veda prie spartaus tobulėjimo šioje srityje.

Pakeisti tradicines transporto priemones savavaldėmis būtų didžiausias, tačiau sunkiai pasiekiamas darbo tikslas, bet tokia yra darbo kryptis. Taigi judant ta linkme siekiama, kad savavaldis robotas ar transporto priemonė supaprastintoje aplinkoje, be žmogaus pagalbos, galėtų judėti kelyje, pats atlikdamas reikalingus sprendimus. Autonominė transporto priemonė, keliaudama pagal nustatytas taisykles, turėtų važiuoti kuo ilgiau nesuklydus.

Sklandžiam transporto priemonės judėjimui įgyvendinti, numatoma sukaupus didelį kiekį duomenų apmokyti ir optimizuoti kelis mašininio mokymo modelius. Tinklai numatomi kurti naudojant konvoliucinius sluoksnius, taip tiesiogiai apdorojant TP matomus vaizdus. Šis modelis privalo su kuo mažesne paklaida aptikti roboto poziciją kelyje. Aptikus poziciją, roboto valdymui numatoma sukurti atskirą kontrolerį ir minimizuoti jo vairavimo klaidą. Visi šie tikslai bus įgyvendinti simuliatoriuje, virtualioje aplinkoje, taip sumažinant sistemos kūrimo kaštus. Sistemai sukūrus paprastą ir nesudėtingą vartotojo sąsają, ja galės pasinaudoti ir kiti norintys.

Kadangi autonominės transporto priemonės dar nėra paplitusios keliuose, šio darbo tikslas nėra realizuoti realios transporto priemonės judėjimą eisme, o tik sukurti prototipą, kaip tai galėtų vykti, jei aplinka būtų tokia paprasta ir nuspėjama kaip simuliacijoje. Tyrimo metu surinkta informacija ir sukurti metodai, gali padėti autonominių transporto priemonių vystymuisi.

## 1.2. Poreikis

Projekto įgyvendinimas kaip procesas, naudingas ne tik moksliniam tobulėjimui, bet sprendžia saugumo, pasiekiamumo ir patogumo problemas. Nors šis projektas skirtas tik priartėti dar vienu žingsniu arčiau prie autonominių transporto priemonių, sprendžiamos vartotojo problemos iš dalies sutampa su galutiniu tikslu. Tyrimas padeda apibendrinti iki tyrimo dienos išleistą medžiagą bei formuoti naują sprendimą. Naujas sprendimas padės suformuluoti išvadas bei rezultatus, panaudojamus ateityje. Galutinis siekiamas tikslas didins saugumą, klysdamas rečiau nei žmogus; suteiks galimybę važiuoti automobiliu įvairias negalias turintiems asmenims ar asmenims, dėl kitų priežasčių negalintiems vairuoti; suteiks galimybę kelionės metu užsiimti kitomis veiklomis.

Tai ankstyva fazė galutinio produkto, todėl didžioji dalis klientų bus kiti bendraminčiai, tokį mokslą tyrinėjantys asmenys. Žvelgiant detaliau, produkto vartotojai gali būti visi norintys pamatyti kaip dirbtinis intelektas sujungiamas su vaizdų apdorojimu ir panaudojamas savavaldžioms priemonėms kurti. Klientus tokio tipo projektas pasiektų tik tuomet, kai sukurta sistema būtų pradėta naudoti, tačiau iki to dar ją reikia ištobulinti.

Bent dalis medžiagos gali būti suprantama ir eiliniam žmogui, kadangi tema aktuali. Dažnai spaudoje galima išvysti straipsnių, teigiančių viena ar kita tiek apie autonominius automobilius, tiek apie dirbinį intelektą. Projekto tyrimai eiliniam žmogui galėtų sudaryti nuomonę apie realią dabartinę situaciją, paremtą pavyzdžiais ir šaltinais.

### 1.3. Savavaldės sistemos

Dabar, kai visi automobiliai valdomi žmonių, pasitikėjimas kompiuterio gebėjimu atlikti tokią funkciją nedidelis. Spaudoje tenka pamatyti savavaldžių automobilių avarijas, kurios verčia susimastyti, ar tokios priemonės gali būti pakankamai saugios. Augant žinių kiekiui apie tokių mašinų kūrimą ir funkcionavimą, tampa aišku, kad kompiuteris gali valdyti transporto priemonę geriau nei žmogus, bet 100% patikimumo galime ir nepasiekti. Tereikia, kad kompiuteris klystų mažiau nei žmogus ir tada transporto valdymą pilnai gali perimti kompiuteris. Dėl šių priežasčių savavaldžių transporto priemonių reikalavimai saugumui buvo pakeisti, kas paskatino jų kūrimą. Esminė užduotis, kuri negali keistis, lieka – realizuoti automobilį, kuris pats gali važiuoti suprasdamas aplinką ir išvengdamas susidūrimų [3].

Autonominiai automobiliai smarkiai patobulėjo nuo pirmojo sėkmingo bandymo 1980 metais [4]. Toks automobilis apibūdinamas kaip – transporto priemonė, gebanti suvokti aplinką ir judėti joje be žmogaus pagalbos [1]. Pavyzdžiu šiomis dienomis galėtumėme laikyti: įmonės Tesla sukurtą automobilio valdymo sistemą „autopilot“, Volvo ir Uber bendromis jėgomis sukurtą trečios kartos autonominio valdymo sistemą, bei tokių gigantų kaip Honda, Daimler ir Toyota kuriamas sistemas numatomas pasirodyti jau 2020 metais.

Tokios mašinos geba sekti savo lokaciją, matant ją žemėlapyje, bei stebint aplinką joje judėti. Iš skirtingų jutiklių kompiuteryje sudaromas detalus žemėlapis, apibūdinantis automobilio netolimą aplinką [2]. Kad automobilis galėtų judėti, sistema privalo gebėti atsakyti į tokius tris klausimus:

1. Kur aš esu?
2. Kur turiu nukeliauti?
3. Kaip ten nukeliausiu?

Į šių klausimų atsakymą gali būti įtraukiamas ir žmogus, tačiau tobulame scenarijuje, žmogus tik pateikia užklausą, kuri padeda sistemai atsakyti į antrą klausimą [5].

Savavaldės sistemos skaidomos į penkis lygius [6], apibūdintus 1.3.1 lentelėje.

**1.3.1 lentelė.** Autonomijos lygiai (Copyright © 2014 SAE International)

| **Lygis** | **Pavadinimas** | **Aprašas** |
|---|---|---|
| 0 | Neautonominė | Visus veiksmus atlieka vairuotojas |
| 1 | Pagalba vairuotojui | Pagalba vairuotojui, kai žmogaus suvokimu įjungiama sistema greitinanti ar lėtinanti automobilį, arba padedanti vairuoti |
| 2 | Dalinai autonominė | Vienos ar daugiau sistemų pagalba vairuotojui iš automobiliui suvokiamos aplinkos, greitinant ar lėtinant automobilį, padedant vairuoti tam tikrais apibrėžtais atvejais |
| 3 | Sąlyginai autonominė | Pilnas transporto priemonės valdymas su išimtimi, kad sistema bet kuriuo momentu gali perduoti valdymą vairuotojui, nebesugebėdama valdyti po perdavimo |

| | | |
|---|---|---|
| 4 | Stipriai autonominė | Pilnas transporto priemonės valdymas su išimtimi, kad sistema bet kuriuo momentu gali perduoti valdymą vairuotojui |
| 5 | Pilnai autonominė | Pilnas transporto priemonės valdymas visą laiką, bet kokiomis vairavimo sąlygomis, bet kokia danga, kur žmogus gebėtų vairuoti |

## 1.4. Rinkos tyrimas

Technologijoms tobulėjant, vyriausybėms vis priimant palankius įstatymus, žmonėms vis labiau pasitikint, rinka sparčiai auga. Taip pat padeda ir panašaus tipo projektai kaip šis. Rinkos dydis 2019 metais įvertintas 54,23 milijardo dolerių bei prognozuojama, kad iki 2026 metų išaugs iki 556,67 milijardo dolerių [7]. Jau naudojami automobiliai su *pagalba vairuotojui* (pirmojo lygio autonomija) šiuo metu sudaro 90 % autonominių automobilių rinkos, o smarkiai tobulėjant technologijoms kompanijos gauna leidimus iš vietinių valdančiųjų testuoti *stipriai autonominius* (ketvirtojo lygio autonomija) automobilius. Prognozuojama, kad jau 2022 metais bus galima įsigyti *stipriai autonominius* automobilius [8].

Rinką taip pat skatina *mobilumo kaip paslaugos* sektorius, suteikiantis pavėžėjimo paslaugas be vairuotojo. Tokia paslauga yra unikali ir labai patraukli, todėl pritraukia daug investicijų.

## 1.5. Produktas

### 1.5.1. Sistemos komponentai

Pagrindinis sistemos komponentas yra transporto priemonė. Savavaldė transporto priemonė privalo turėti bent keletą sensorių, kuriais suvokia aplinką. Tokie sensoriai sistemai suteikia aplinkos vaizdą, pagal kurį sistema atlieka sprendimus. Aplinkos jutikliai gali būti dviejų tipų: išorinės arba vidinės būsenos [2, 9].

Išorinės būsenos jutikliai:

- „LiDAR“ – jutiklis skirtas atstumo matavimui. Veikimo principas paremtas skrydžio trukmės principu (angl. time of flight), skaičiuojant laiką nuo šviesos signalo išsiuntimo iki jo grįžimo. Savaime suprantama, kad jutiklio veikimo laikas tiesiogiai susijęs su šviesos greičiu, taigi tai pakankamai greitas jutiklis [10]. Paprastai šio jutiklio pagalba sudaromas netolimos aplinkos atstumų žemėlapis, leidžiantis matyti kliūtis ir taip jų išvengti. Greitas, tačiau norint matyti 360° vaizdą gali kainuoti ir dešimtimis tūkstančių eurų.
- Radaras – tai jutiklis, veikiantis elektromagnetinės radiacijos principais, siunčiama radijo banga atsimušus nuo objektų (50m-150m atstumu) grįžta į jutiklį ir taip sužinoma objekto lokacija ir pasisukimo kampas [11]. Paprastai naudojamas judančių daiktų sekimui, be to, šis jutiklis gerokai pigesnis nei „LiDAR“, todėl jau dabar plačiai naudojamas dalinai autonominiuose automobiliuose.
- Kamera – prietaisas, veikiantis šviesos absorbcijos į skaitmeninį jutiklį principu. Pranašesnis už kitus išorinės būsenos jutiklius, nes geba išskirti tekstūrą, spalvas. Gerai sukalibruota kamerų pora galima nuspręsti ir atstumą tarp objektų [12]. Didžiausios kameros silpnybės būtų, bet kokia aplinka su prastu apšvietimu, oro sąlygos ir sunkiai suvokiamas gylis. Paprastai naudojamos vaizdo segmentavimui, kelių eismo taisyklių pagal žymėjimus suvokimui.

- Ultragarso jutiklis – veikia siųsdamas garso bangas ir laukiantis jų sugrįžtant, bei skaičiuodamas šio proceso laiką [13]. Lėčiausias, mažiausiai patikimas (nes stipriai veikiamas aplinkos veiksnių) tačiau pigiausias jutiklis. Ilgą laiką naudojamas automobiliuose kaip parkavimosi daviklis, iki šių dienų vis dar skirtas tikslumo nereikalaujantiems ir netolimiems matavimams.

Vidinės būsenos jutikliai:

- GPS – tai jutiklis, kuris, komunikuojant su keturiais ar dargiau palydovų, gali nusakyti objekto buvimo vietą. Lokacija, deja, numatoma apie trijų metrų tikslumu, o ir tuo ne visada galima pasikliauti, nes šis jutiklis sunkiai veikia tuneliuose ar vietose su daug aukštų pastatų [14]. Paprastai naudojamas lokacijos nustatymui ir maršrutų pagal lokaciją radimui. Pats jutiklis nėra brangus, o naudojimasis palydovais nemokamas.
- Padėties koduotuvas – (angl. encoder) jutiklis, skirtas matuoti objektų judėjimui, su kuriais galima turėti sąlytį. Koduotuvas mechaninių apsisukimų metu grąžina impulsų kiekį, kurį galima suvesti su realaus pasaulio matmenimis. Tokie jutikliai dažnai naudojami atlikti odometro funkcijai.
- IMU – jutiklis, sudarytas iš giroskopų, akselerometrų ir magnetometrų kiekvienai trimatės dimensijos ašiai (X, Y, Z). Daviklis geba nusakyti kūno judėjimą, tačiau ne padėtį [15]. Dabar dažnai sutinkamas visuose telefonuose.

Apžvelgus pagrindinius aplinkos suvokimo prietaisus, buvo pasirinkta tyrimą atlikti simuliacijoje. Tai daug griežtesnė ir stipriau kontroliuojama aplinka, kurioje galima koncentruotis į mažesnes problemas. Visas problemas sprendžiant palaipsniui ir keliant aplinkos sudėtingumą, gaunami aiškesni tyrimo rezultatai.

### 1.5.2. Vaizdų atpažinimas, segmentacija

Pagrindinis sistemos veikimas pagrįstas vaizdų apdorojimu. Savavaldis robotas, turėdamas prie savęs pritaisytą kamerą, iš jos gautus vaizdus apdoroja ir pagal tai pateikia roboto valdymo sprendimus.

Objektų išskyrimui reikalinga tų objektų specifika, tačiau prieš atliekant bet kokius išskyrimus realiomis sąlygomis ir tikintis tikslumo, reikia atlikti pradinius veiksmus. Tokių veiksmų paskirtis, sutvarkyti iš kameros gaunamą vaizdą, kad jis būtų kuo mažiau užterštas ir kiekvieną kart gavus naują kadrą, matytume pasikeitusią aplinkos informaciją, o ne objektų metamus šešėlius ar kitus trikdžius.

Pradiniai žingsniai: Apšvietimo kompensavimas – šis žingsnis reikalingas, norint normalizuoti gaunamą vaizdą. Kadangi kiekvienam gaunamam vaizdui bus taikoma ši funkcija, tai turėtų būti paprastas ir greit atliekamas skaičiavimas. Vienas iš paprastesnių metodų OpenCV bibliotekoje yra histrogramos išlyginimas. Šio metodo veikimo principas yra, kontrasto lyginimas visame paveikslėlyje, išskleidžiant jo histogramą, per visą ilgį [16]. Šio proceso metu taip pat normalizuojamos ir paveikslėlio vertės, kai aukščiausia vertė pasiekia 255, o žemiausia 0. Histogramos lyginimas skaičiuojamas formule [17]:

$$Išlyginta(x, y) = H'(pradinė(x, y)) \quad (1)$$

$$H'(i) = \sum_{0 \le j \le i} H(j) \quad (2)$$

Kai $H$ – histograma, $H'$ - Išlyginta histograma.

Kameros kalibravimas – pradedant nuo to, kad kiekviena kamera turi objektyvą. Reikia turėti omenyje, kad objektyvas gaunamą vaizdą iškraipo (kuo toliau nuo objektyvo centro, tuo vaizdas labiau iškreiptas ir tamsesnis). Problema neišsprendžiama keičiant objektyvą, nes visi objektyvai veikia panašiu principu. Vienintelis realus variantas gauti neiškreiptą vaizdą, yra naudoti telecentrinio tipo objektyvą (angl. telecentric lens), tačiau jo dydis turi būti norimo matyti vaizdo dydis, taigi pats objektyvas būtų automobilio dydžio, o tai mažų mažiausia – nepraktiška. Taigi, pasirenkamas paprastesnis kelias ir tokie iškraipymai taisomi programinės įrangos pagalba. Iškraipymo kompensavimui naudojami spinduliniai ir tangentiniai faktoriai. Spinduliniam iškraipymui kompensuoti taikome [18]:

$$x_{kalibruotas} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \quad (3)$$

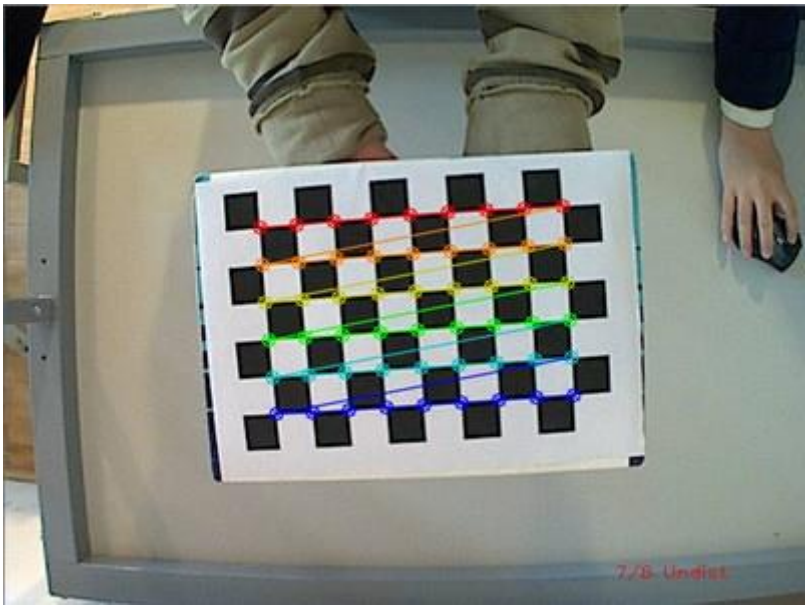
$$y_{kalibruotas} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \quad (4)$$

Tangentiniam iškraipymui kompensuoti taikome:

$$x_{kompensuotas} = x + [2p_1 xy + p_2(r^2 + 2x^2)] \quad (5)$$

$$y_{kompensuotas} = y + [p_1(r^2 + 2y^2) + 2p_2 xy] \quad (6)$$

Kai $k_1,\ k_2, p_1, p_2, k_3$ yra kompensavimo parametrai. Šie parametrai gaunami kalibravimo metu, skaičiuojant į šachmatų lentą panašios struktūros (1.5.1 pav.) iškraipymus.



**1.5.1 pav.** OpenCv kalibravimo šablonas su skaičiavimo dalių žymėjimais [18]

Atlikus tokius pradinius veiksmus, galima kibti į darbą ir ieškoti reikiamų vaizdo detalių. Pirmoji užduotis – identifikuoti kelio juostą, kad būtų galima ją sekti. Jau sutvarkytam vaizdui pritaikome „Bayesian“ filtrą, taip išskirdami kelio linijas nuo fono. Toliau linijų kraštų išskyrimui, naudojamas

„Canny“ algoritmas, o nustatyti spalvai „Hue-Saturation-Value colorspace thresholding“ [19]. Taip išgavę kelio juostos kontūrus galime atlikti sprendimą – kur važiuos robotas.

Roboto valdymas gali būti įgyvendintas dviem būdais:

- Vaizdų atpažinimu ir skaičiavimais grįstas sprendimas. Šis metodas remiasi vaizdo pavertimu į plokštumos homografiją (angl. homography). Turint tokį vaizdą, iš jo kiekvienam sąrašo (kelio juostai i) elementui nustatome poziciją, iš šių elementų sudarius histogramą „Bayes“ filtro pagalba galime apskaičiuoti, kur robotui reiks važiuoti [19].
- Mašininiu mokymu grįstas sprendimas. Dažnai taikomas sprendimas, kai susiduriama su vaizdais, yra kognityvinis neuroninis tinklas, tačiau toks sprendimas yra pakankamai imlus resursams. Žinoma, taip galima spręsti tokio tipo uždavinį, ir ,apmokius modelį, gauti išvestis, į kurią pusę važiuoti robotui [20]. Tačiau paprastesnis būdas būtų gautais duomenimis apmokyti paprastesnį modelį kaip atraminių vektorių mašinos ar k artimiausių kaimynų.

Reiktų pastebėti, kad naudojant stimuliatorių su šiais trikdžiais galime nesusidurti. Tai vienas iš supaprastintos aplinkos privalumų. Tačiau, norint algoritmą geriau paruošti realioms sąlygoms, simuliatoriuje galime įjungti tiek vaizdo iškraipymą, tiek skirtingą apšvietimą.

### 1.5.3. Aplinka „DuckieTown“

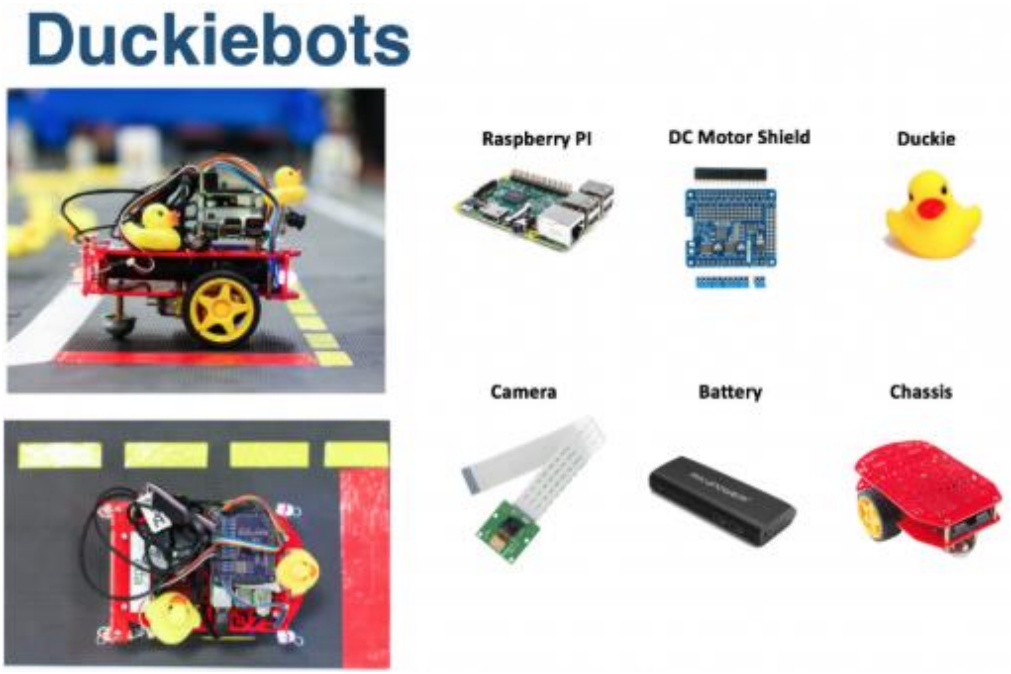
Eksperimentams atlikti buvo pasirinkta naudoti DuckieTown virtualią aplinką. Taip planuojami atlikti anksčiau aprašyti testavimai. Tokiu būdu bus išvengiama techninės įrangos gedimų ir kitų tiesiogiai su autonominiu transporto priemonės valdymu nesusijusių problemų.

DuckieTown – 2016 metais Masačiusetso technologijų universitete sukurta supaprastinta platforma sudaryta iš parduotuvėje nuperkamų komponentų, skirta mokytis ir mokyti autonominio transporto priemonės valdymo ypatumų (1.5.2 pav.) [21].

**1.5.2 pav.** DuckieTown fizinė aplinka ir DuckietBot [21]

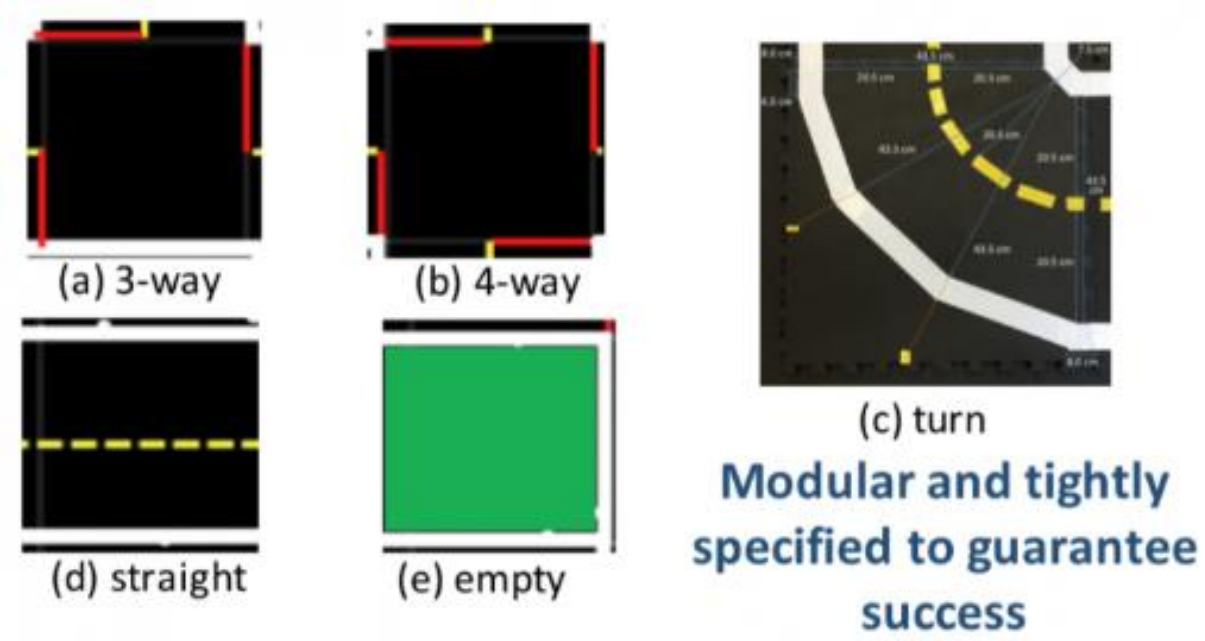
DuckieTown platforma sudaryta iš:

DuckieBots (savavaldžiai robotai) – tai nedideli robotukai, su kamera, RaspberryPi kompiuteriu ir elektriniais motorais ratų valdymui. Toks paprastas robotukas skirtas važinėti supaprastintoje aplinkoje, sekant kelių juostas, vengiant pėsčiųjų ir skaitant kelio ženklus (**1.5.3 pav**.) [22].




**1.5.3 pav.** Savavaldžio robotuko komponentai [22]

DuckieTown (supaprastinta aplinka) – tai dvejų sluoksnių aplinka iš struktūrinių elementų. Tokia aplinka buvo sukurta specialiai šiam uždaviniui spręsti, stengiantis jį maksimaliai supaprastinti (1.5.2 pav.) [22]. Supaprastintos aplinkos sluoksniai: grindų sluoksnis (1.5.2 pav.), ženklų sluoksnis (1.5.5 pav.).




**1.5.4 pav.** Grindų sluoksnis [22]

Traffic lights

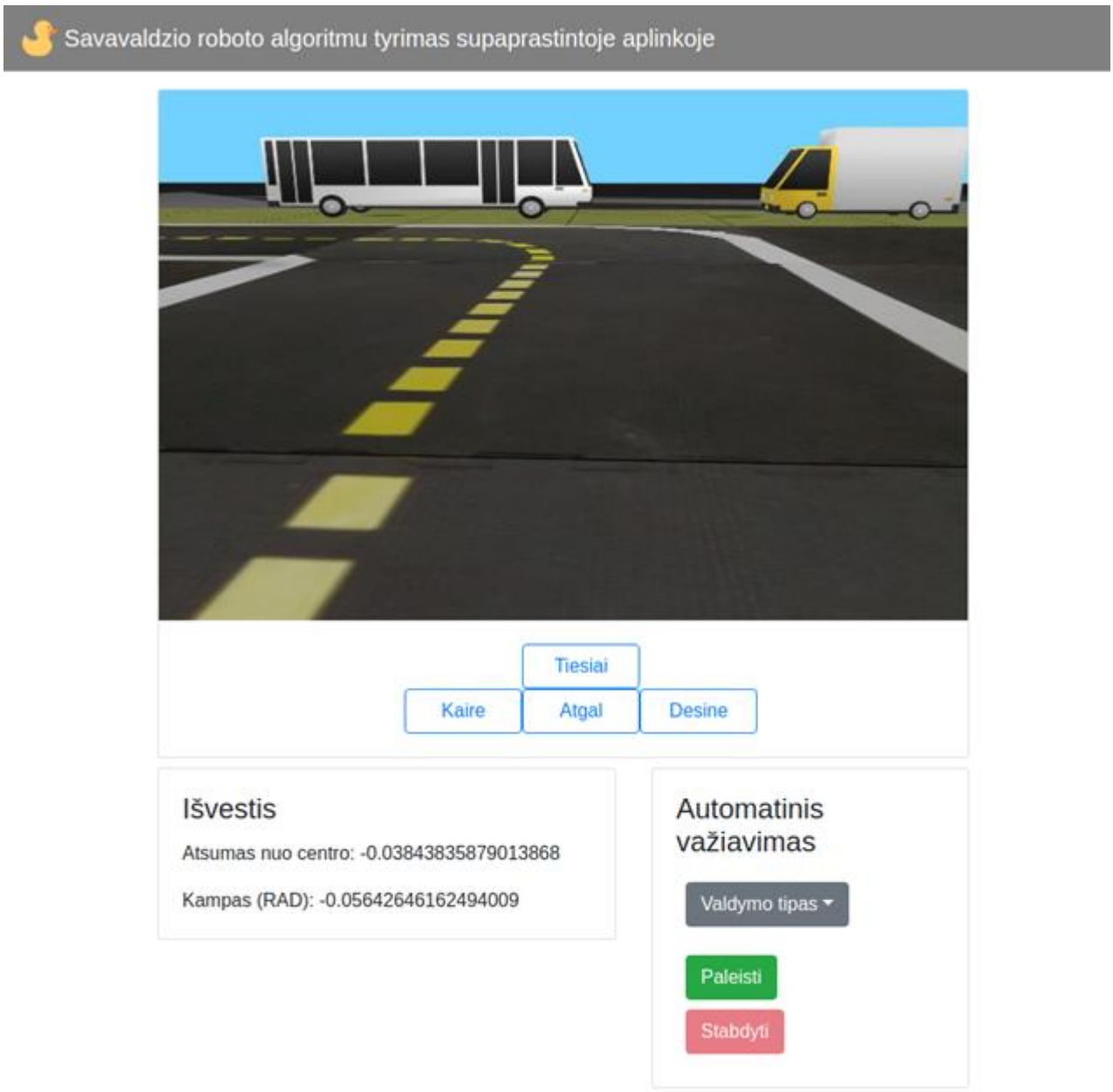

Road signs

**1.5.5 pav.** Ženklų sluoksnis [22]

## 1.6. Produktas

Produkto galutinis atlikimas įgyvendintas virtualioje aplinkoje. Taip įgyvendinta sistema, susiduria su mažiau triukšmo ir leidžia pamatyti švaresnius tyrimo rezultatus. Visi toliau atliekami euristiniai bandymai atliekami taikant vienodą, netintamą apšviestumą, su vienodu tekstūrų išsidėstymu. Pašalinti bet koki atsitiktiniai sistemos pasikeitimai, kurie galėtų turėti įtakos tyrimų netolygumams

Tyrimo galutinis produktas (1.6.1 pav.) - informacinė sistema, paleidžiama vartotojo kompiuteryje ir modeliuojanti roboto važiavimą supaprastintoje aplinkoje. Sistema skirta autonomiškai valdyti robotą apibrėžtoje aplinkoje. Patį kodą bus galima redaguoti ir atlikti pakeitimus.



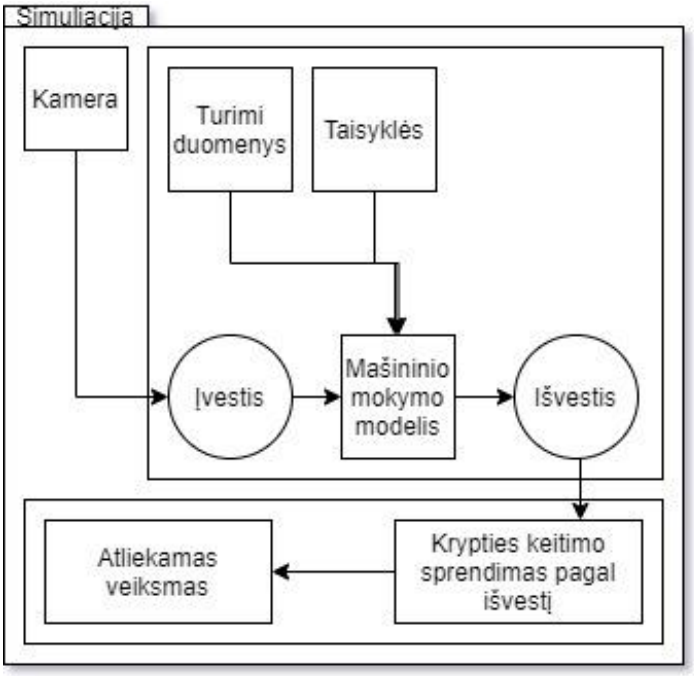

**1.6.1 pav.** Galutinis produktas

Sistema geba atlikti šiuos veiksmus:

- gauti vaizdą iš virtualaus roboto vaizdo kameros;
- identifikuoti kelių žymėjimą (judėjimo juostas);
- automatizuotai važiuoti pagal kelio žymėjimą taikant pasirinktą konfigūraciją:
    - Roboto valdymo metodai:
        - P (proporcinis) kontroleris
        - PI (proporcinis integralinis) kontroleris
        - PID (proporcinis integralinis išvestinis) kontroleris
    - Kelio atpažinimo metodai:
        - Mašininio mokymo modelis
        - Modeliavimo įtaiso duomenys
- kontroliuoti roboto judėjimą rankiniu rėžimu;

- atvaizduoti sistemos būsenos duomenis realiu laiku.

Tyrimo rezultatu gautas produktas bus paleistas kompiuteryje vykdomoje simuliacijoje. Virtualioje aplinkoje simuliuojamas triratis robotas. Virtualaus roboto pagrindą sudaro kamera ir varikliai, valdantys robotą (1.6.2 pav.).




**1.6.2 pav.** Konteksto diagrama

Tyrimo metu gauta programinė įranga yra atspari klaidoms, kad neišsijungtų neprognozuojamai. Visi skaičiavimai atliekami vietoje esančiame kompiuteryje, nes duomenų siuntimas užtrunka laiko, ateityje taikant tokią architektūrą taip apsisaugoma nuo potencialių programišių atakų per tinklą. Klaidos ir informaciniai pranešimai išvedami vartotojo sąsajoje.

### 1.6.1. Programinė įranga

Darbe numatoma naudoti programinė įranga (1.6.1 lentelė). Matoma „DuckieTown“ virtualizacija bus pirmas žingsnis savavaldžio roboto link. Visi bandymai, modifikacijos ir kūrimas bus atliekami virtualioje aplinkoje, taip sutaupant ir laiko ir lėšų.

**1.6.1 lentelė** Darbe naudojama programinė įranga

| **Naudojama technologija** | **Aprašas** |
|---|---|
| Windows OS | Simuliacijoms ir testams, atliekamiems kompiuteryje, naudojama aplinka |
| Python | Programavimo kalba. Kalba pasirinkta, nes lengvai dera su mašininiu mokymu, bei DuckieTown palaiko Python kalbą. |
| PyTorch | Dirbtinio intelekto biblioteka Python programavimo kalbai. |
| OpenCV | Vaizdų apdorojimo biblioteka, skirta vaizdams iš kameros apdoroti. |
| DuckieTown | Virtuali aplinka skirta DuckieBots testavimui. |

### 1.6.2. Techninė įranga

Bandymai simuliatoriuje atlikti staliniu kompiuteriu (1.6.2 lentelė). Tokiam darbui galingo kompiuterio nereikia. Norint atlikti apmokymą, prireiks daugiau resursų, nes tai procesas, kuriam reikia atlikti daug matematinių skaičiavimų. Tam pasirinkta naudoti „Google Colaboratory“ paslaugą. Naudojant šią paslaugą, gaunama atsitiktinė aparatūrinės įrangos kombinacija, kurioje naudojami „Intel Skylake“ linijos procesoriai ir „Nvidia“ pramoninės, dideliems skaičiavimams pritaikytos, vaizdo plokštės.

**1.6.2 lentelė** Kompiuterio aprašymas

| **Komponentas** | **Aprašas** |
|---|---|
| Procesorius | Intel Core i5-4570 CPU @ 3.20GHz |
| Operatyvioji atmintis | 8GB |
| Vaizdo plokštė | NVIDIA GeForce GT 1030 |

## 1.7. Įgyvendinimo problemos

Savivaldžių transporto priemonių kūrimas paprastai būna labai brangus ir sunkiai įgyvendamas. Apart to, taip pat visi jutikliai gauna didelį kiekį triukšmo. Dėl šios priežasties stipriai išauga užduoties sudėtingumas, nes tenka susidurti ne tik su autonominės transporto priemonės užduotimis, bet ir su minėtų triukšmų filtravimu. Tačiau šią problemą stipriai palengvina DuckieTown aplinka. Taip sumažinus užduoties sudėtingumą, galima koncentruotis ties konkrečios problemos sprendimu, o tokius sprendimus jau turint, juos galima perkelti į realias sąlygas [5].

Taip pat, tokio tipo sistemos susiduria su greitaveikos problemomis, nes kiekvienas skaičiavimas turi trukti pakankamai trumpą laiką, kad laiku būtų grąžinamas rezultatas ir transporto priemonė laiku atliktų sprendimą. O vėlavimo pasekmės gali būti labai didelės.

## 1.8. Konkurencija ir alternatyvos

Pagrindinei veiklai projekte esant tyrimui, realios konkurencijos nėra. Tyrimas skirtas peržvelgti metodams, juos palyginti ir sukurti prototipą. Tačiau žinant, kad konkurentai trukdo produktui patekti į rinką, didžiausias savavaldžių transporto priemonių konkurentas yra vairuotojų valdomi automobiliai. Nors dabar gali atrodyti priešingai, tačiau vairuotojo valdomas automobilis nėra saugus, nes stipriai priklauso nuo vairuotojo įgūdžių, patirties, būsenos ir pan., taip pat manoma, kad kompiuteriu galima išgauti ir greitesnį reakcijos laiką. Žmogaus valdomos ir savavaldės transporto priemonės palyginimas (1.8.1 lentelė).

**1.8.1 lentelė.** Savivaldžių ir žmogaus valdomų transporto priemonių palyginimas

| | **Žmogaus valdoma TP** | **Savavaldė TP** |
|---|---|---|
| Komunikacija tarp TP | Nėra (pamirksėjimas šviesomis, rankų gestai, ar garsinis signalas deja ne visiems suprantami). | Galima naudotis įvairiais jutikliais ir siųstuvais. |
| Maršruto, spūsčių tikslios žinios | Nėra (nors vairuotojas gali žinoti tendencijas kelių, kuriais dažniausiai važiuoja, tai nėra faktinė informacija, kokią gali turėti žemėlapių sistemos). | Galima naudoti įvairias GPS sistemas, su spūsčių informacija ir taip efektyviai planuoti maršrutą. |

| | | |
|---|---|---|
| Reakcijos greitis | Priklauso nuo vairuotojo nuotaikos, amžiaus, nuovargio, pastabumo, patirties ir galybės kitų faktorių. | Reakcijos greitis varijuoja per sistemas, tačiau nėra veikiamas nuotaikų, nuovargio ar pan. |
| Patirtis | Kaupiama kiekvieno vairuotojo atskirai. | Kaupiama iš skirtingų transporto priemonių į bendrą sistemą, taip užtikrinant spartų tobulėjimą. |
| Nenumatytos problemos | Žmogus, dėl sveikatos, amžiaus ar kitų, su asmens būsena susijusių faktorių, gali prarasti mašinos kontrolę ir sukelti avariją. | Įvykus negrįžtamai žalai, bet kokiu atveju TP saugiai stabdoma artimiausioje vietoje. |

## 2. Autonominio roboto valdomo supaprastintoje aplinkoje projektas

Savavaldės transporto priemonės padeda palengvinti kasdienius rūpesčius ir didinti saugumą keliuose. Tokio tipo transporto priemonės gali smarkiai sumažinti spūsčių kiekį, taip tausojant gamtą ir mažinant kelionių kaštus. Kelionės tokiu automobiliu gali būti pasiekiamos visiems, nepriklausomai nuo amžiaus, sveikatos, nuovargio bei kitų sutrikimų ar būsenos.

### 2.1. Sistemos tikslai (paskirtis)

Savavaldžio roboto autonominis judėjimas virtualioje aplinkoje pagal simuliatoriaus duomenis, sekant kelią. (pasiektas, jei robotas, be žmogaus pagalbos, gali sėkmingai judėti, sekti kelią).

Modelio, skirto važiuojamosios dalies vertinimui atpažinti, sukūrimas. (pasiektas, jei modelio tikslumas siekia 80 ar daugiau procentų.)

Savavaldžio roboto autonominis judėjimas virtualioje aplinkoje duomenis gaunant iš roboto kameros, sekant kelią. (pasiektas, jei robotas, be žmogaus pagalbos, gali sėkmingai judėti, sekti kelią).

### 2.2. Apribojimai

Sistema realizuojama Python programavimo kalba – plačiai pripažinta tarp tyrėjų. Pageidaujama paskutine jos versija arba vėliausia, kurią stabiliai palaiko PyTorch ir OpenCV bibliotekos.

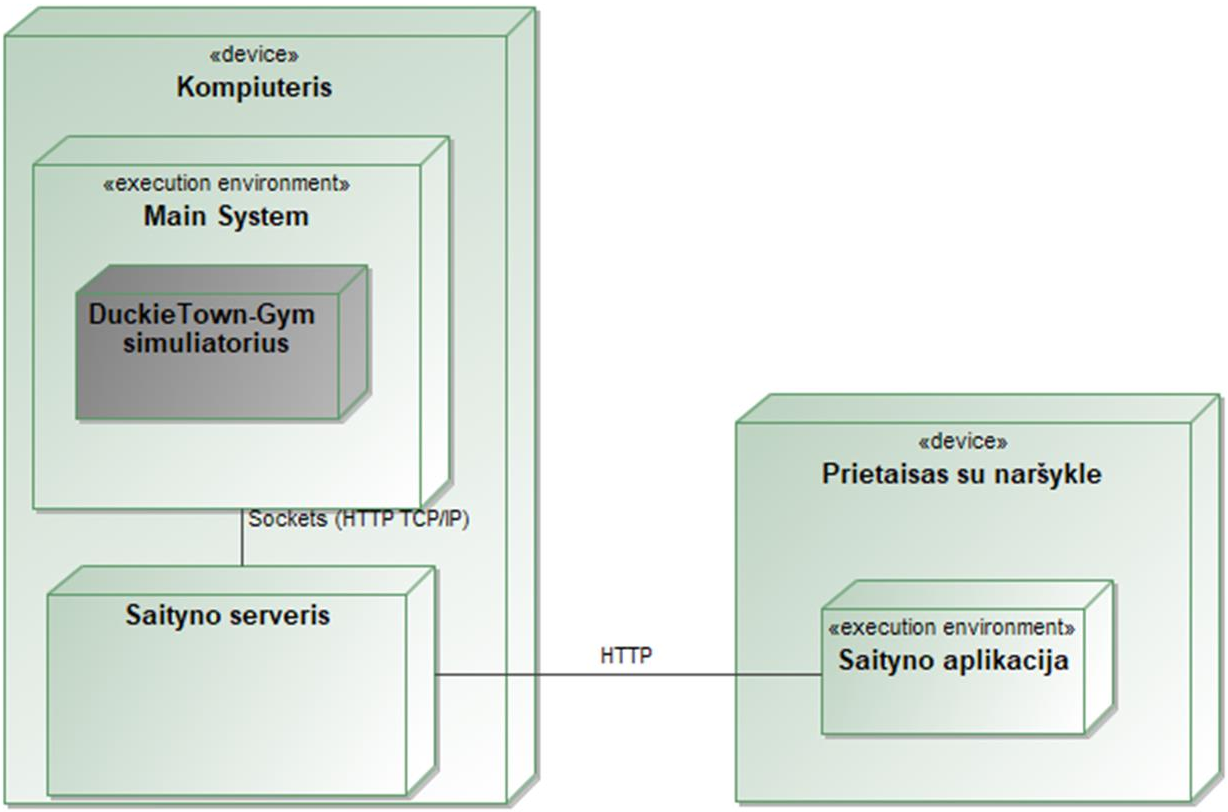
Vaizdų analizei turi būti naudojama OpenCV biblioteka kaip viena iš lyderių, atliekant bet kokius vaizdo apdorojimo sprendimus.

Mašininiam mokymui naudojama PyTorch biblioteka kaip viena iš lyderių, apmokant mašininius modelius. Taip pat, biblioteka nesunkiai siejama su virtualia aplinka Duckietown.

Virtualios sistemos bandymai bus atliekami ant duckietown virtualios aplinkos – tai pritaikytas ir stabilus įrankis, kuriamos sistemos testavimams atlikti.

### 2.3. Diegimo aplinka

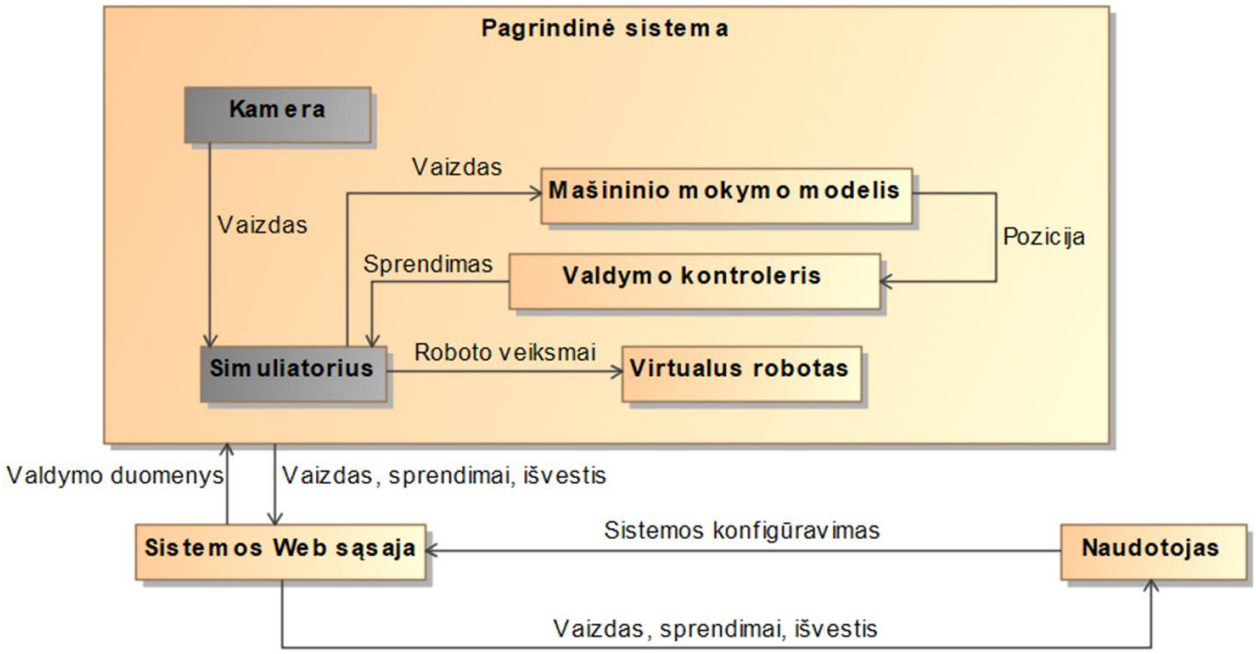
Sistemos diegimas numatomas kompiuteryje naudojant DuckieTown simuliatorių, kuriame bus paleidžiama ir automatinio vairavimo sistema (1.6.2 pav.).




**2.3.1 pav.** Virtuali diegimo aplinka

## 2.4. Veiklos kontekstas (pateikiama konteksto diagrama)

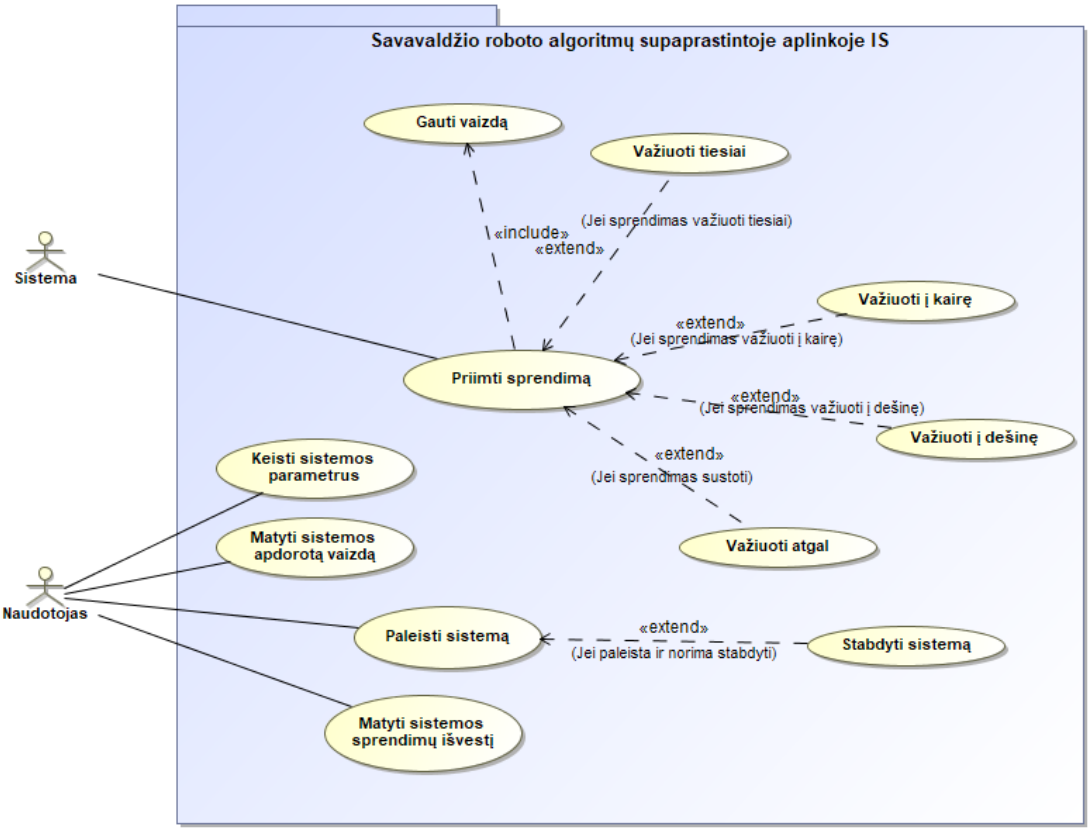
Veiklos konteksto diagrama matoma paveikslėlyje žemiau (2.4.1 pav.).




**2.4.1 pav.** Veiklos kontekstas

## 2.5. Sistemos ribos

Sistemos ribos nurodomos panaudojimo atvejų diagramos pagalba (2.5.1 pav.).




**2.5.1 pav.** Panaudojimo atvejų diagrama

### 2.5.1. Panaudojimo atvejų sąrašas

Sistemos ribose numatomų panaudojimo atvejų detalizavimas lentelėse žemiau.

**2.5.1 lentelė** PA "Priimti sprendimą"

| **1 PA.** "Priimti sprendimą" | |
|---|---|
| **Vartotojas/Aktorius** | Sistema. |
| **Aprašas** | Sistema pagal turimus duomenis ir apmokytą mašininio mokymo algoritmą, priima sprendimą apie roboto veiksmus erdvėje. |
| **Prieš sąlyga** | Sistema paleista. |
| **Sužadinimo sąlyga** | Paleistos sistemos gyvybinis cikle gautas kadras ir atstumas |
| **Po sąlyga** | Priimtas sprendimas ir perduotas į roboto kontrolierių. |

**2.5.2 lentelė** PA "Gauti vaizdą"

| **2 PA.** "Gauti vaizdą" | |
|---|---|
| **Vartotojas/Aktorius** | Sistema. |
| **Aprašas** | Gaunamas aplinkos vaizdas iš kameros. Pagal šį vaizdą atliekama analizė. |
| **Prieš sąlyga** | Sistema paleista. |

| | |
|---|---|
| **Sužadinimo sąlyga** | Paleistos sistemos gyvavimo ciklas. |
| **Po sąlyga** | Sėkmingai gautas vaizdas, o informacija perduota sprendimo priėmimui. |

**2.5.3 lentelė** PA "Važiuoti tiesiai"

| **4 PA.** "Važiuoti tiesiai" | |
|---|---|
| **Vartotojas/Aktorius** | Sistema. |
| **Aprašas** | Roboto žingsnis pavažiuojantis tiesiai. |
| **Prieš sąlyga** | Paleista sistema ir priimtas sprendimas. |
| **Sužadinimo sąlyga** | Gautas sprendimas važiuoti tiesiai. |
| **Po sąlyga** | Robotas pajudėjo į priekį. |

**2.5.4 lentelė** PA "Važiuoti į kairę"

| **5 PA.** "Važiuoti į kairę" | |
|---|---|
| **Vartotojas/Aktorius** | Sistema. |
| **Aprašas** | Roboto žingsnis pavažiuojantis į kairę. |
| **Prieš sąlyga** | Paleista sistema ir priimtas sprendimas. |
| **Sužadinimo sąlyga** | Gautas sprendimas važiuoti į kairę. |
| **Po sąlyga** | Robotas pajudėjo į kairę. |

**2.5.5 lentelė** PA "Važiuoti į dešinę"

| **6 PA.** "Važiuoti į dešinę" | |
|---|---|
| **Vartotojas/Aktorius** | Sistema. |
| **Aprašas** | Roboto žingsnis pavažiuojantis į dešinę. |
| **Prieš sąlyga** | Paleista sistema ir priimtas sprendimas. |
| **Sužadinimo sąlyga** | Gautas sprendimas važiuoti į dešinę. |
| **Po sąlyga** | Robotas pajudėjo į dešinę. |

**2.5.6 lentelė** PA "Važiuoti atgal"

| **7 PA.** "Sustoti" | |
|---|---|
| **Vartotojas/Aktorius** | Sistema. |
| **Aprašas** | Roboto važiavimas atbulomis |
| **Prieš sąlyga** | Paleista sistema ir priimtas sprendimas. |
| **Sužadinimo sąlyga** | Gautas sprendimas važiuoti atbulomis. |
| **Po sąlyga** | Robotas pavažiavo atbulomis. |

**2.5.7 lentelė** PA "Keisti sistemos parametrus"

| **8 PA.** "Keisti sistemos parametrus" | |
|---|---|
| **Vartotojas/Aktorius** | Naudotojas. |
| **Aprašas** | Sistemos veikimas priklauso nuo tam tikrų sistemos parametrų. Naudotojas norėdamas juos gali pritaikyti pagal savo reikmes. |
| **Prieš sąlyga** | - |
| **Sužadinimo sąlyga** | Saugojami pakeisti parametrai. |
| **Po sąlyga** | Išsaugoti/atnaujinti pakeisti parametrai. |

**2.5.8 lentelė** PA "Matyti sistemos apdorotą vaizdą"

| **9 PA.** "Matyti sistemos apdorotą vaizdą" |
|---|

| | |
|---|---|
| **Vartotojas/Aktorius** | Naudotojas. |
| **Aprašas** | Naudotojui atvaizduojamas apdorotas sistemos vaizdas, padedantis vizualizuoti sistemos priimamų sprendimus. |
| **Prieš sąlyga** | Sistema paleista, naudotojas sistemos lange, kur atvaizduojami apdoroti vaizdai. |
| **Sužadinimo sąlyga** | Gautas apdorotas vaizdas. |
| **Po sąlyga** | Atvaizduotas apdorotas vaizdas. |

**2.5.9 lentelė** PA "Matyti sistemos sprendimų išvestį"

| **10 PA.** "Matyti sistemos sprendimų išvestį" | |
|---|---|
| **Vartotojas/Aktorius** | Naudotojas. |
| **Aprašas** | Naudotojui atvaizduojami sistemos sprendimai, padedantys perteikti daugiau informacijos apie sprendimą ir jo rezultatus. |
| **Prieš sąlyga** | Sistema paleista, naudotojas sistemos lange, kur atvaizduojama išvesties informacija. |
| **Sužadinimo sąlyga** | Gauta sprendimų išvesties informacija. |
| **Po sąlyga** | Atvaizduota sprendimų išvesties informacija. |

**2.5.10 lentelė** PA "Paleisti sistemą"

| **11 PA.** "Paleisti sistemą" | |
|---|---|
| **Vartotojas/Aktorius** | Naudotojas. |
| **Aprašas** | Sistemos paleidimas. Važiavimo sesijos pradžia. |
| **Prieš sąlyga** | Sistema nepaleista. |
| **Sužadinimo sąlyga** | Paspaustas mygtukas „Paleisti sistemą". |
| **Po sąlyga** | Sistema paleista. |

**2.5.11 lentelė** PA "Stabdyti sistemą"

| **12 PA.** "Stabdyti sistemą" | |
|---|---|
| **Vartotojas/Aktorius** | Naudotojas. |
| **Aprašas** | Sistemos stabdymas. Važiavimo sesijos pabaiga. |
| **Prieš sąlyga** | Sistema paleista. |
| **Sužadinimo sąlyga** | Paspaustas mygtukas „Stabdyti sistemą". |
| **Po sąlyga** | Sistema sustabdyta. |

## 2.6. Funkciniai reikalavimai ir reikalavimai duomenims

### 2.6.1. Funkciniai reikalavimai

**2.6.1 lentelė** Funkcinis reikalavimas nr. 1

| | |
|---|---|
| **Reikalavimas #:** **1** **Reikalavimo tipas:** **9** **Įvykis/panaudojimo atvejis #:** **1** | |
| **Aprašymas:** | Sprendimui priimti naudojamas jau apmokytas mašininio mokymo modelis. |

| | | | |
|---|---|---|---|
| **Pagrindimas:** | Sprendimą reikia priimti pagal tam tikras taisykles. Jas sistema jau turi žinoti ir pagal tai veikti. Taip pat, sistemą apmokyti kiekvieną kart prieš naudojant – nepatogu. | | |
| **Šaltinis:** | Jurgis Kišūnas | | |
| **Tikimo kriterijus:** | Sprendimas priimamas per <1s pagal nustatytas taisykles, o paleidžiant sistemą nereikia atlikti modelio apmokymo. | | |
| **Užsakovo tenkinimas:** | 5 | **Užsakovo netenkinimas:** | 4 |
| **Priklausomybės:** | Nėra | **Konfliktai:** | Nėra |
| **Papildoma medžiaga:** | Nėra | | |
| **Istorija:** | Užregistruotas 2020 kovo 10 d. | | |

**2.6.2 lentelė** Funkcinis reikalavimas nr. 2

| | | | | | |
|---|---|---|---|---|---|
| **Reikalavimas #:** | **2** | **Reikalavimo tipas:** | **9** | **Įvykis/panaudojimo atvejis #:** | **3** |
| **Aprašymas:** | Maksimali leistina modelio gaunamų verčių paklaida 0,05 | | | | |
| **Pagrindimas:** | Sklandžiam sistemos veikimui reikalingas tikslumas, pagal kurį galima pasikliauti, kurioje vietoje kelio yra robotas. | | | | |
| **Šaltinis:** | Jurgis Kišūnas | | | | |
| **Tikimo kriterijus:** | Vertinant roboto poziciją kelyje paklaida negali būti didesnė negu 0,05. | | | | |
| **Užsakovo tenkinimas:** | 2 | **Užsakovo netenkinimas:** | | | 5 |
| **Priklausomybės:** | Nėra | **Konfliktai:** | | | Nėra |
| **Papildoma medžiaga:** | Nėra | | | | |
| **Istorija:** | Užregistruotas 2020 kovo 10 d. | | | | |

**2.6.3 lentelė** Funkcinis reikalavimas nr. 3

| | | | | | |
|---|---|---|---|---|---|
| **Reikalavimas #:** | **3** | **Reikalavimo tipas:** | **9** | **Įvykis/panaudojimo atvejis #:** | **2** |

| | | | |
|---|---|---|---|
| **Aprašymas:** | Kameros gautam vaizdui reikia atstatyti objektyvo iškraipymus, jei toki iškraipymai egzistuoja. | | |
| **Pagrindimas:** | Objektyvo iškraipymai, trukdo bet kokiems tolimesniems skaičiavimams. | | |
| **Šaltinis:** | Jurgis Kišūnas | | |
| **Tikimo kriterijus:** | Vaizdo iškraipymas artimas nuliui | | |
| **Užsakovo tenkinimas:** | 1 | **Užsakovo netenkinimas:** | 1 |
| **Priklausomybės:** | Nėra | **Konfliktai:** | Nėra |
| **Papildoma medžiaga:** | Nėra | | |
| **Istorija:** | Užregistruotas 2020 kovo 10 d. | | |

**2.6.4 lentelė** Funkcinis reikalavimas nr. 4

| | | | | | |
|---|---|---|---|---|---|
| **Reikalavimas #:** | **4** | **Reikalavimo tipas:** | **9** | **Įvykis/panaudojimo atvejis #:** | **2** |
| **Aprašymas:** | Kameros gautas vaizdas turi būti 80x60x30 dydžio. | | | | |
| **Pagrindimas:** | Tokių dimensijų užtenka modeliui įvertinti aplinkybes, todėl didesnė raiška tik lėtintų sistemos veikimą. | | | | |
| **Šaltinis:** | Jurgis Kišūnas | | | | |
| **Tikimo kriterijus:** | Vaizdo dimensijos duodamos mašininio mokymo modeliui turi būti 80x60x3 | | | | |
| **Užsakovo tenkinimas:** | 1 | **Užsakovo netenkinimas:** | | | 1 |
| **Priklausomybės:** | Nėra | **Konfliktai:** | | | Nėra |
| **Papildoma medžiaga:** | Nėra | | | | |
| **Istorija:** | Užregistruotas 2020 kovo 10 d. | | | | |

**2.6.5 lentelė** Funkcinis reikalavimas nr. 5

| | | | | | |
|---|---|---|---|---|---|
| **Reikalavimas #:** | **5** | **Reikalavimo tipas:** | **9** | **Įvykis/panaudojimo atvejis #:** | **4, 5, 6, 8** |

| | | | |
|---|---|---|---|
| **Aprašymas:** | Roboto judėjimo spartumas turi būti dinamiškas parametras | | |
| **Pagrindimas:** | Robotui pagal skirtingas aplinkas ar kitas savybes gali reikėti judėti greičiau ar lėčiau, todėl parametras turėtų būti dinamiškas | | |
| **Šaltinis:** | Jurgis Kišūnas | | |
| **Tikimo kriterijus:** | Parametras dinamiškai kinta pagal poreikius. | | |
| **Užsakovo tenkinimas:** | 3 | **Užsakovo netenkinimas:** | 3 |
| **Priklausomybės:** | Nėra | **Konfliktai:** | Nėra |
| **Papildoma medžiaga:** | Nėra | | |
| **Istorija:** | Užregistruotas 2020 kovo 10 d. | | |

**2.6.6 lentelė** Funkcinis reikalavimas nr. 6

| | | | | | |
|---|---|---|---|---|---|
| **Reikalavimas #:** | **6** | **Reikalavimo tipas:** | **9** | **Įvykis/panaudojimo atvejis #:** | **2** |
| **Aprašymas:** | Sistema privalo gauti vaizdą. | | | | |
| **Pagrindimas:** | Vaizdas privalomas analizei atlikti. | | | | |
| **Šaltinis:** | Jurgis Kišūnas | | | | |
| **Tikimo kriterijus:** | Ar gaunamas vaizdas iš kameros. | | | | |
| **Užsakovo tenkinimas:** | 1 | **Užsakovo netenkinimas:** | | | 5 |
| **Priklausomybės:** | Nėra | **Konfliktai:** | | | Nėra |
| **Papildoma medžiaga:** | Nėra | | | | |
| **Istorija:** | Užregistruotas 2020 kovo 10 d. | | | | |

**2.6.7 lentelė** Funkcinis reikalavimas nr. 7

| | | | | | |
|---|---|---|---|---|---|
| **Reikalavimas #:** | **7** | **Reikalavimo tipas:** | **9** | **Įvykis/panaudojimo atvejis #:** | **4** |

| | | | |
|---|---|---|---|
| **Aprašymas:** | Sistema privalo galėti važiuoti tiesiai | | |
| **Pagrindimas:** | Sistema, nevažiuojanti tiesiai, yra sunkiai ir nenuspėjamai kontroliuojama. | | |
| **Šaltinis:** | Jurgis Kišūnas | | |
| **Tikimo kriterijus:** | Ar su sprendimu „važiuoti tiesiai“ sistema važiuoja tiesiai. | | |
| **Užsakovo tenkinimas:** | 3 | **Užsakovo netenkinimas:** | 4 |
| **Priklausomybės:** | Nėra | **Konfliktai:** | Nėra |
| **Papildoma medžiaga:** | Nėra | | |
| **Istorija:** | Užregistruotas 2020 kovo 10 d. | | |

**2.6.8 lentelė** Funkcinis reikalavimas nr. 8

| | | | | | |
|---|---|---|---|---|---|
| **Reikalavimas #:** | **8** | **Reikalavimo tipas:** | **9** | **Įvykis/panaudojimo atvejis #:** | **5, 6** |
| **Aprašymas:** | Sistema privalo galėti sukti (važiuoti į kairę ar dešinę). | | | | |
| **Pagrindimas:** | Sistema, negalinti sukti, negali išvengti kliūčių ir manevruoti. | | | | |
| **Šaltinis:** | Jurgis Kišūnas | | | | |
| **Tikimo kriterijus:** | Ar sistema su sprendimu „sunkti į dešinę“ suka į dešinę, o su sprendimu „sukti į kairę“, suka į kairę. | | | | |
| **Užsakovo tenkinimas:** | 3 | **Užsakovo netenkinimas:** | | 4 | |
| **Priklausomybės:** | Nėra | **Konfliktai:** | | Nėra | |
| **Papildoma medžiaga:** | Nėra | | | | |
| **Istorija:** | Užregistruotas 2020 kovo 10 d. | | | | |

**2.6.9 lentelė** Funkcinis reikalavimas nr. 9

| | | | | | |
|---|---|---|---|---|---|
| **Reikalavimas #:** | **9** | **Reikalavimo tipas:** | **9** | **Įvykis/panaudojimo atvejis #:** | **7** |
| **Aprašymas:** | Sistema privalo galėti važiuoti atbulomis. | | | | |
| **Pagrindimas:** | Robotui reikia galėti važiuoti atbulomis. Tokioje situacijoje, kai nieko nebegalima daryti, vienintelė galimybė – važiuoti atbulomis. | | | | |
| **Šaltinis:** | Jurgis Kišūnas | | | | |
| **Tikimo kriterijus:** | Ar sistemoje robotas gali važiuoti atbulomis. | | | | |
| **Užsakovo tenkinimas:** | 3 | **Užsakovo netenkinimas:** | | | 5 |
| **Priklausomybės:** | Nėra | **Konfliktai:** | | | Nėra |
| **Papildoma medžiaga:** | Nėra | | | | |
| **Istorija:** | Užregistruotas 2020 kovo 10 d. | | | | |

**2.6.10 lentelė** Funkcinis reikalavimas nr. 10

| | | | | | |
|---|---|---|---|---|---|
| **Reikalavimas #:** | **10** | **Reikalavimo tipas:** | **9** | **Įvykis/panaudojimo atvejis #:** | **7, 12** |
| **Aprašymas:** | Sustabdoma sistema privalo sustabdyti robotą. | | | | |
| **Pagrindimas:** | Robotas, sustabdžius sprendimus, gali nuvažiuoti bet kur ir sukelti pavojų. | | | | |
| **Šaltinis:** | Jurgis Kišūnas | | | | |
| **Tikimo kriterijus:** | Ar sistemai sustojus ir baigus darbą sustojo ir robotas. | | | | |
| **Užsakovo tenkinimas:** | 2 | **Užsakovo netenkinimas:** | | | 5 |
| **Priklausomybės:** | Nėra | **Konfliktai:** | | | Nėra |
| **Papildoma medžiaga:** | Nėra | | | | |
| **Istorija:** | Užregistruotas 2020 kovo 10 d. | | | | |

### 2.6.2. Nefunkciniai reikalavimai

#### 2.6.2.1. Reikalavimai sistemos išvaizdai

**2.6.11 lentelė** Nefunkcinis reikalavimas nr. 12

| | | | | | |
|---|---|---|---|---|---|
| **Reikalavimas #:** | **12** | **Reikalavimo tipas:** | **10** | **Įvykis/panaudojimo atvejis #:** | **visi** |
| **Aprašymas:** | Klaidos turi būti akcentuojamos raudona spalva. | | | | |
| **Pagrindimas:** | Klaidos turi būti ryškios ir aiškiai matomos, kad naudotojas jas iškart pastebėtų. | | | | |
| **Šaltinis:** | Jurgis Kišūnas | | | | |
| **Tikimo kriterijus:** | Ar sistemoje klaidos atvaizduojamos raudonai. | | | | |
| **Užsakovo tenkinimas:** | 2 | **Užsakovo netenkinimas:** | | | 2 |
| **Priklausomybės:** | Nėra | **Konfliktai:** | | | Nėra |
| **Papildoma medžiaga:** | Nėra | | | | |
| **Istorija:** | Užregistruotas 2020 kovo 10 d. | | | | |

**2.6.12 lentelė** Nefunkcinis reikalavimas nr. 13

| | | | | | |
|---|---|---|---|---|---|
| **Reikalavimas #:** | **13** | **Reikalavimo tipas:** | **10** | **Įvykis/panaudojimo atvejis #:** | **visi** |
| **Aprašymas:** | Sąsaja paruošta, naudojant „Bootstrap“ komponentus. | | | | |
| **Pagrindimas:** | Sistemos naudojimas paprastas ir intuityvus. | | | | |
| **Šaltinis:** | Jurgis Kišūnas | | | | |
| **Tikimo kriterijus:** | Ar sistema suskurta naudojant *Bootstrap* karkasą | | | | |
| **Užsakovo tenkinimas:** | 2 | **Užsakovo netenkinimas:** | | | 2 |
| **Priklausomybės:** | Nėra | **Konfliktai:** | | | Nėra |

| | |
|---|---|
| **sPapildoma medžiaga:** | Nėra |
| **Istorija:** | Užregistruotas 2020 kovo 10 d. |

### 2.6.2.2. Reikalavimai panaudojamumui

**2.6.13 lentelė** Nefunkcinis reikalavimas nr. 14

| | | | | | |
|---|---|---|---|---|---|
| **Reikalavimas #:** | **14** | **Reikalavimo tipas:** | **11** | **Įvykis/panaudojimo atvejis #:** | **visi** |
| **Aprašymas:** | Sistemos valdymas turi būti aiškiai suprantamas, be jokio papildomo apmokymo. | | | | |
| **Pagrindimas:** | Sistemą galėtų išbandyti bet kas. | | | | |
| **Šaltinis:** | Jurgis Kišūnas | | | | |
| **Tikimo kriterijus:** | Ar sistema išdėliota aiškiai ir suprantamai eiliniam žmogui ir negaunama nusiskundimų. | | | | |
| **Užsakovo tenkinimas:** | 2 | **Užsakovo netenkinimas:** | | | 2 |
| **Priklausomybės:** | Nėra | | **Konfliktai:** | | Nėra |
| **Papildoma medžiaga:** | Nėra | | | | |
| **Istorija:** | Užregistruotas 2020 kovo 10 d. | | | | |

**2.6.14 lentelė** Nefunkcinis reikalavimas nr. 15

| | | | | | |
|---|---|---|---|---|---|
| **Reikalavimas #:** | **15** | **Reikalavimo tipas:** | **11** | **Įvykis/panaudojimo atvejis #:** | **visi** |
| **Aprašymas:** | Pagal nutylėjimą sistemos kalba – Lietuvių. | | | | |
| **Pagrindimas:** | Sistema kuriama Lietuviškoj institucijoje. | | | | |
| **Šaltinis:** | Jurgis Kišūnas | | | | |
| **Tikimo kriterijus:** | Ar pirmąkart įsijungus sistemą ji veikia lietuvių kalba. | | | | |

| | | | |
|---|---|---|---|
| **Užsakovo tenkinimas:** | 2 | **Užsakovo netenkinimas:** | 3 |
| **Priklausomybės:** | Nėra | **Konfliktai:** | Nėra |
| **Papildoma medžiaga:** | Nėra | | |
| **Istorija:** | Užregistruotas 2020 kovo 10 d. | | |

**2.6.15 lentelė** Nefunkcinis reikalavimas nr. 16

| | | | | | |
|---|---|---|---|---|---|
| **Reikalavimas #:** | **16** | **Reikalavimo tipas:** | **11** | **Įvykis/panaudojimo atvejis #:** | **visi** |
| **Aprašymas:** | Sistemos valdymas turi būti aiškiai suprantamas, be jokio papildomo apmokymo. | | | | |
| **Pagrindimas:** | Sistemą galėtų išbandyti bet kas. | | | | |
| **Šaltinis:** | Jurgis Kišūnas | | | | |
| **Tikimo kriterijus:** | Ar sistema išdėliota aiškiai ir suprantamai eiliniam žmogui ir negaunama nusiskundimų. | | | | |
| **Užsakovo tenkinimas:** | 2 | **Užsakovo netenkinimas:** | | | 2 |
| **Priklausomybės:** | Nėra | **Konfliktai:** | | | Nėra |
| **Papildoma medžiaga:** | Nėra | | | | |
| **Istorija:** | Užregistruotas 2020 kovo 10 d. | | | | |

### 2.6.2.3. Reikalavimai vykdymo charakteristikoms

**2.6.16 lentelė** Nefunkcinis reikalavimas nr. 17

| | | | | | |
|---|---|---|---|---|---|
| **Reikalavimas #:** | **17** | **Reikalavimo tipas:** | **12** | **Įvykis/panaudojimo atvejis #:** | **visi** |
| **Aprašymas:** | Nuo kadro gavimo iki sprendimo išdavimo negali praeiti daugiau nei sekundė. | | | | |
| **Pagrindimas:** | Priešingu atveju sistema bus per lėta bet kokiam naudojimui. | | | | |
| **Šaltinis:** | Jurgis Kišūnas | | | | |

| | | | |
|---|---|---|---|
| **Tikimo kriterijus:** | Ar sistemos atsako laikas <1s. | | |
| **Užsakovo tenkinimas:** | 2 | **Užsakovo netenkinimas:** | 2 |
| **Priklausomybės:** | Nėra | **Konfliktai:** | Nėra |
| **Papildoma medžiaga:** | Nėra | | |
| **Istorija:** | Užregistruotas 2020 kovo 10 d. | | |

### 2.6.2.4. Reikalavimai veikimo sąlygoms

**2.6.17 lentelė** Nefunkcinis reikalavimas nr. 18

| | | | | | |
|---|---|---|---|---|---|
| **Reikalavimas #:** | **18** | **Reikalavimo tipas:** | **13** | **Įvykis/panaudojimo atvejis #:** | **visi** |
| **Aprašymas:** | Sistema privalo veikti supaprastintinomis aplinkos sąlygomis. | | | | |
| **Pagrindimas:** | Supaprastintos sąlygos skirtos tokio tipo uždaviniui, sistema tokiuose rėžiuose privalo veikti. | | | | |
| **Šaltinis:** | Jurgis Kišūnas | | | | |
| **Tikimo kriterijus:** | Sistema atitinka duckietown aplinkos kriterijus. | | | | |
| **Užsakovo tenkinimas:** | 2 | **Užsakovo netenkinimas:** | | | 2 |
| **Priklausomybės:** | Nėra | **Konfliktai:** | | | Nėra |
| **Papildoma medžiaga:** | https://docs.duckietown.org/DT19/opmanual_duckietown/out/dt_ops_appearance_specifications.html | | | | |
| **Istorija:** | Užregistruotas 2020 kovo 10 d. | | | | |

### 2.6.2.5. Reikalavimai sistemos priežiūrai

**2.6.18 lentelė** Nefunkcinis reikalavimas nr. 19

| | | | | | |
|---|---|---|---|---|---|
| **Reikalavimas #:** | **19** | **Reikalavimo tipas:** | **14** | **Įvykis/panaudojimo atvejis #:** | **visi** |

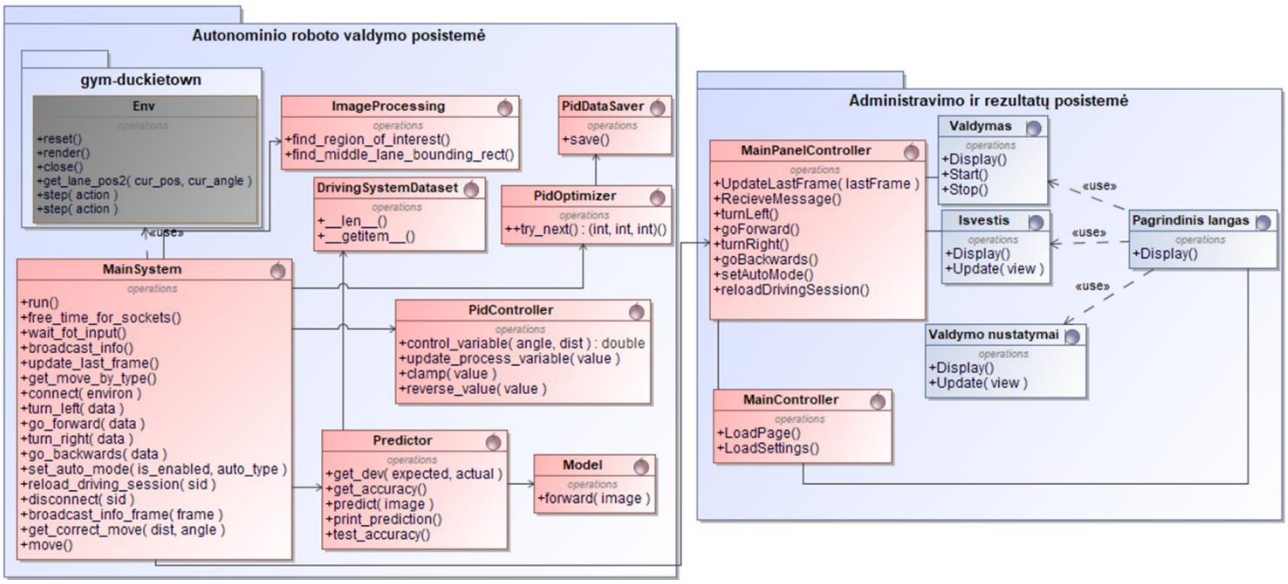
| | | | |
|---|---|---|---|
| **Aprašymas:** | Sistema turi būti nesunkiai paleidžiama, | | |
| **Pagrindimas:** | Lengvas sistemos paleidimas | | |
| **Šaltinis:** | Jurgis Kišūnas | | |
| **Tikimo kriterijus:** | Sistema paleidžiama kompiuteryje per mažiau nei 2 valandas. | | |
| **Užsakovo tenkinimas:** | 2 | **Užsakovo netenkinimas:** | 2 |
| **Priklausomybės:** | Nėra | **Konfliktai:** | Nėra |
| **Papildoma medžiaga:** | Nėra | | |
| **Istorija:** | Užregistruotas 2020 kovo 10 d. | | |

### 2.7. Paketų diagrama

Šis posistemių skaidymas atskiria  kalbomis. Mašininio mokymo ir sistemos simuliacijos dalis daroma su Python programavimo kalba, o vartotojo sąsajos su JavaScript.

Autonominio roboto valdymo posistemė atsakinga už roboto veiksmus. Šioje posistemėje, pagal išorinius veiksnius, pasiteikiant dirbtinį intelektą, sistema nusako kaip robotas judės supaprastintoje aplinkoje (2.7.1 pav. kairėje).

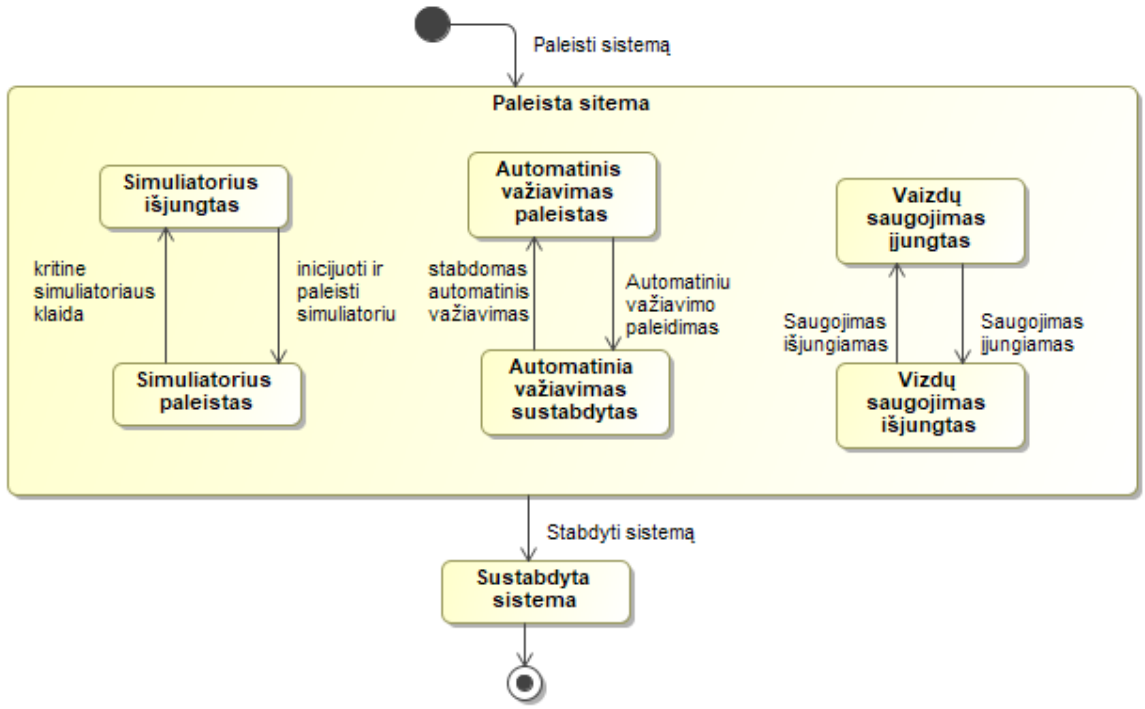
Administravimo ir rezultatų posistemė skirta sąveikai su autonominio roboto valdymo posisteme. Ši posistemė leidžia keisti pirmosios veikimo principą pagal tam tikrus parametrus, matyti sistemos išvestį, matyti roboto gaunamą vaizdą, bei paleisti ar stabdyti sistemą (2.7.1 pav. dešinėje).




**2.7.1 pav.** Paketų diagrama

## 2.8. Sistemos būsenos

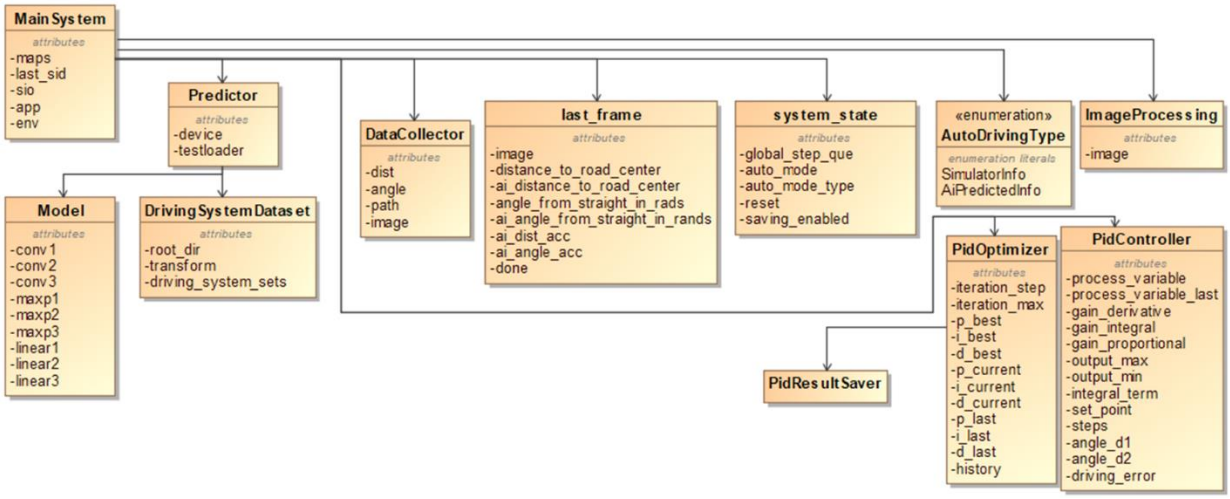
Sistemos būsenos pavaizduotos diagramoje 2.8.1 pav.




**2.8.1 pav.** Sistemos būsenos diagrama

## 2.9. Sistemos duomenys

Duomenų vaizdas sistemoje pateiktas klasių diagrama (2.9.1 pav.). Duomenų bazė sistemoje nėra reikalinga, todėl nenaudojama.




**2.9.1 pav.** Sistemos klasių diagrama

# 3. Autonominių transporto priemonių valdymo tyrimas

## 3.1. Tyrimo tikslas

Tyrimo tikslas yra nustatyti kuris iš pasirinktų mašininio mokymo modelių labiau tinka roboto pozicijai kelyje įvertinti. Tyrimo metu privaloma pririnkti aibę anotuotų nuotraukų, su kuriomis bus apmokomi konvoliuciniai neuroniniai tinklai. Numanoma, kad visos nuotraukos prieš analizę turi būti apdorotos, kad būtų paprasčiau suvokiamos algoritmui.

Tyrimas susideda iš dviejų esminių dalių:

- Kelio, roboto pozicijos kelyje aptikimas
- Roboto valdymas

Šios dalys bus nagrinėjamos atskirai. Pozicijos aptikimui, būtina išskirti duomenis, kuriais bus atvaizduojama pozicija, o renkant nuotraukas, pasirinkti parametrai bus naudojami anotacijai. Taip pozicijos radimo algoritmas bus apmokomas „su mokytoju“ ir aiškiai žinosime jo gražinamus rezultatus.

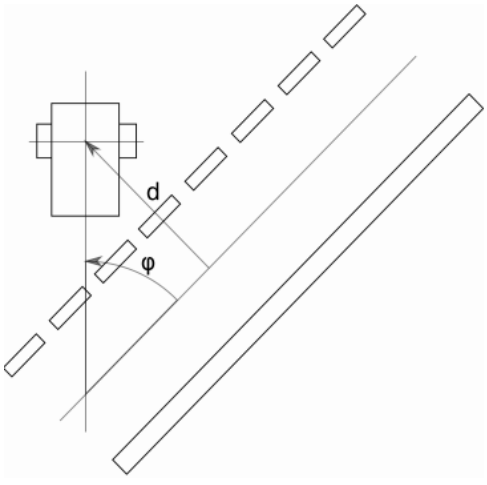
Roboto valdymo pirminis algoritmas robotą nukreips į vieną iš trijų krypčių (kairę, tiesiai arba į dešinę). O norint sumažinti vairavimo klaidą, bus tiriama kokia, geriausia PID reguliatoriaus kombinacija tinka roboto valdymui.

## 3.2. Pozicijos aptikimas

### 3.2.1. Tyrimo aprašymas

Atliekant pirminę analizę, buvo priimtas sprendimas naudoti konvoliucinius neuroninius tinklus. Tokio tipo tinklai paprastai nesunkiai prisitaiko prie tiriamos problemos ir pritaikyti darbui su nuotraukomis.

Naudojamame *DuckieTown* simuliatoriuje robotas turi keletą parametrų kuriais identifikuojama TP pozicija. Vienas iš jų yra pozicionavimas žemėlapyje. Nors tai gana tikslus, tačiau sudėtingas būdas roboto pozicijai įvardinti, nes būtina žinoti žemėlapį. Kitas būdas susideda tik iš dviejų parametrų: atstumo nuo juostos centro ir kampo tar roboto ir kelio (3.2.1 pav.). Pastarasis variantas labiau tinka tiriamai problemai, todėl ir bus naudojamas tyrimo eigoje.




**3.2.1 pav.** Roboto pozicija kelyje [23]

Modelių apmokymui privalu turėti aibę duomenų su minėtais dviem parametrais. Kiekviena nuotrauka turėdama informaciją apie savo poziciją bus panaudota modelio apmokymui

Pirminis modelis iš struktūros planuojamas panašus į *LeNet* modelį. Toliau modelis bus optimizuojamas ir bandomas išgauti kuo geresnis rezultatas. Siekiama sukurti modelį su 70% ar didesniu tikslumu. Kadangi mokymo laikas priklauso nuo duomenų kiekio dydžio ir modelio sudėtingumo. Bus sukurti du mašininio mokymo modeliai. Mažesnis modelis, su mažiau sluoksnių ir parametrų, kuris bus greičiau apmokomas, tačiau galimai mažiau tikslus ir didesnis, su daugiau sluoksnių.

### 3.2.2. Duomenų rinkimas

Jau žinant kaip anotuoti duomenis, pradedamas nuotraukų rinkimas. Tai paprastai yra ilgas ir varginantis procesas. Jį galima palengvinti, pilnai automatizuojant, nes naudojame simuliatorių. Nors ir naudojant automatizaciją, būtina apsibrėžti keletą taisyklių apie numatomas gauti nuotraukas:

- nuotraukoje privalo matytis kelias
- robotas privalo būti ant kelio
- kelias turi būti švarus ir be pašalinių objektų
- aplinka turi būti paprasta, kad algoritmui pavyktų išskirti kelią
- visos nuotraukos privalo būti iš *DuckieTown* stimuliatoriaus
- nuotraukos negali būti per didelės, kad neužimtų daug atminties
- nuotraukos negali būti per mažos (kai nebeišskiriamas logiškas vaizdas)

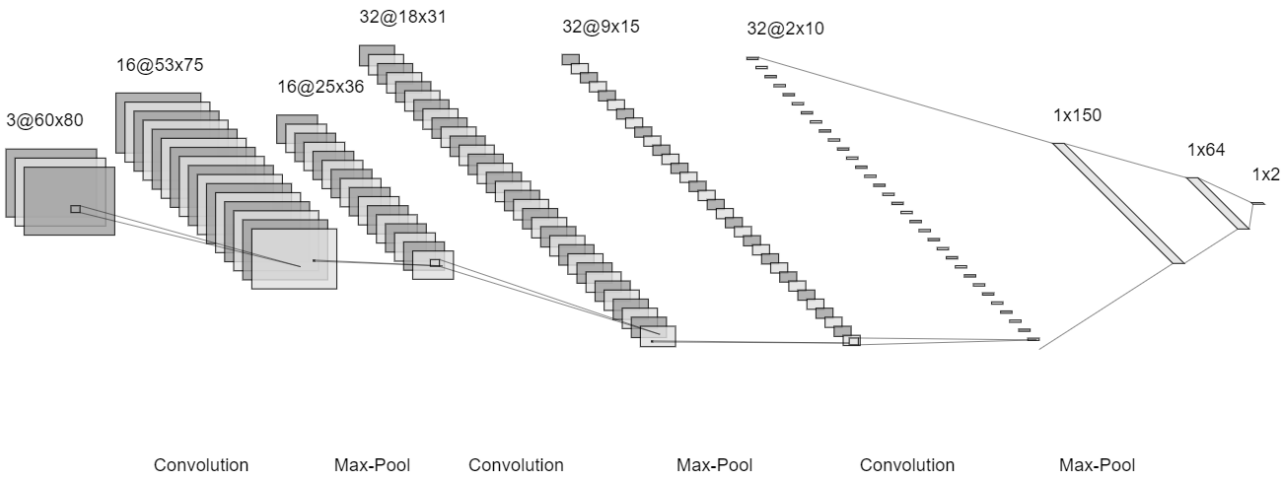
Pagal numatytas taisykles, automatizuotu būdu buvo surinkta 18 tūkstančių nuotraukų. Visos nuotraukos buvo peržiūrėtos ir patikrintos ar atitinka visas taisykles. Po patikrinimo liko 16,9 tūkstančio nuotraukų, kurios bus naudojamas modelių apmokymui.

### 3.2.3. Modelių apmokymas

Konvoliucinio tinklo, ar bet kokio kito neuroninio tinklo modelio kūrimo metu susiduriame su problema, kad modelis gali būti per didelis ir sunkiai apmokomas, ar per paprastas ir netikslus. Taip pat didelis modelis reikalauja daugiau vietos saugojimui, dėl didelio parametrų kiekio. Kuo didesnis modelis ir turi daugiau parametrų, tuo jo naudojimui reikia daugiau resursų. Nors dabartinė sistema leidžiama simuliatoriuje, ją galima perkelti į robotą, valdomą *RaspberryPi* kompiuteriu. Turint omenyje tokio kompiuterio ribotumą, reiktų rinktis kuo paprastesnį modelį.

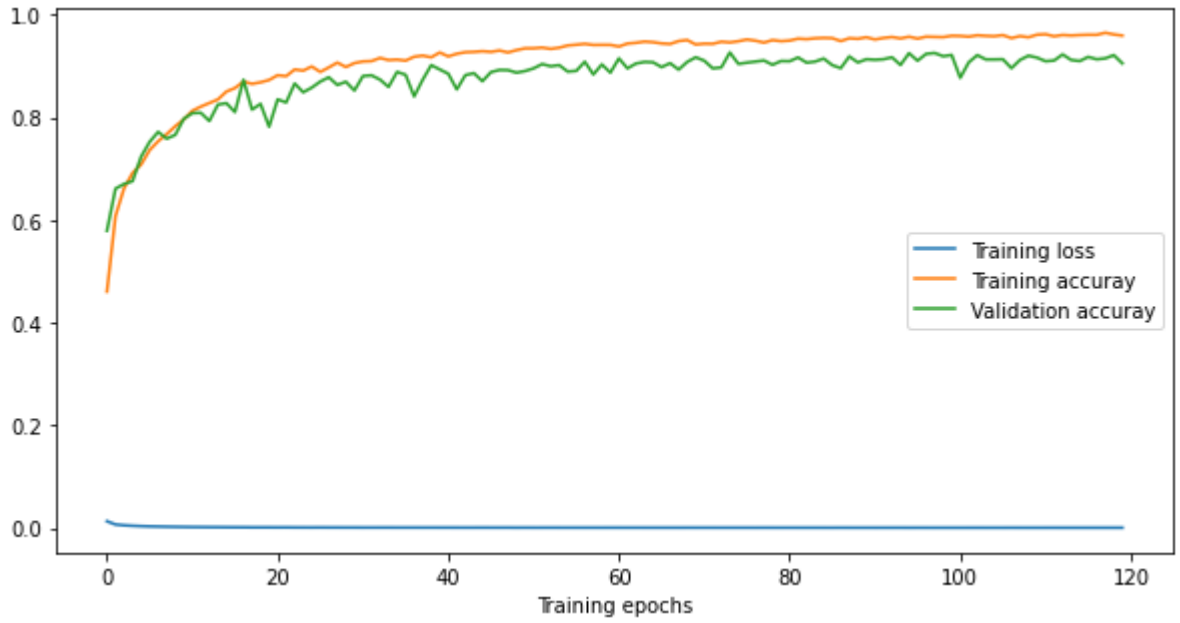
Sukaupus tinkamą duomenų kiekį buvo pradėtas apmokymas. Turimos nuotraukos buvo apdorojamos, kad tiktų mašininio mokymo bibliotekai *PyTorch*. Sukurti duomenų objektai, buvo padalinti į tris rinkinius: mokymui, testavimui ir validavimui.

Pirmasis modelis buvo kuriamas, semiantis idėjas iš *LeNet* tinklo. Jį sudaro trys konvoliuciniai sluoksniai, po kiekvieno sekant suminam kaupimui ir išlyginant duomenis trimis linijiniais sluoksniais (3.2.2 pav.). Modelio įvestis yra 80x60 dydžio nuotraukos - roboto matomas vaizdas. Išvestyje gauname du reikiamus parametrus, roboto nukrypimo nuo kelio kampą ir atstumą iki juostos centro.




**3.2.2 pav.** Mašininio mokymo modelio (Nr. 1) architektūra

Modelio optimizavimas buvo atliekamas skaičiuojant vidutinę kvadratinę klaidą, taikant *AdaMax* optimizavimo metodą, su mokymo greičiu 0,001 per 120 epochų. Tai padėjo gana greitai gauti gerus rezultatus, ir po 3,5 valandos mokymo, naudojant *ColabPro* įrankį gavome reikalavimus tenkinančius rezultatus (3.2.3 pav.). Jau 73 epochoje mokymo tikslumas siekė 94% (iš kurių 99% atstumui ir 94% kampui), o validavimo tikslumas 92% (iš kurių 98% atstumui ir 92% kampui).




**3.2.3 pav.** Mašininio mokymo modelio (Nr. 1) treniravimo rezultatų diagrama

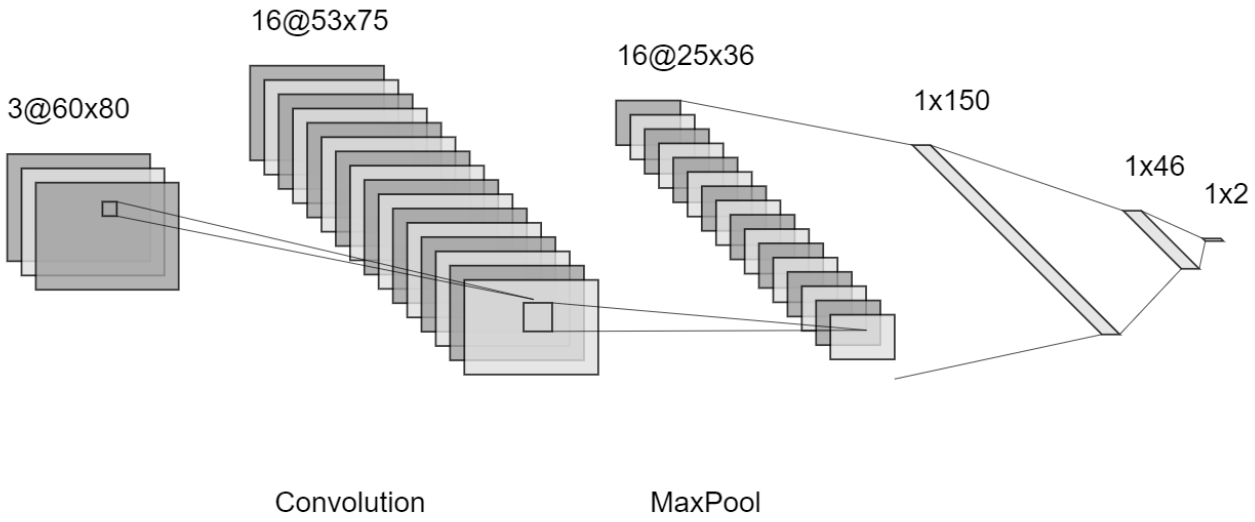
Sukurtas modelis išsaugojamas panaudojimui. Modelio savybės matomos 3.2.1 lentelėje.

**3.2.1 lentelė** Mašininio mokymo modelio (Nr. 1) savybės

| | |
|---|---|
| Bendri parametrai (vnt.) | 129256 |
| Apmokymo parametrai (vnt.) | 129256 |
| Neapmokomi parametrai (vnt.) | 0 |
| Įvesties dydis (MB) | 0,05 |
| Priekinės ir atgalinės sklaidos dydis (MB) | 0,77 |

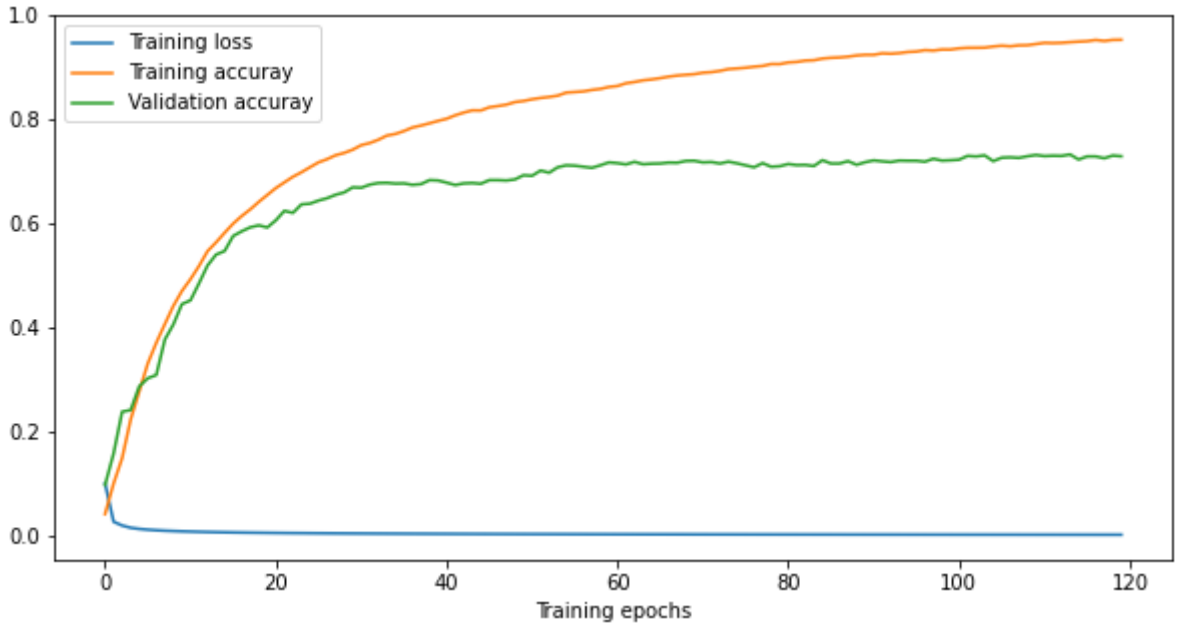
| | |
|---|---|
| Parametrų dydis (MB) | 0,49 |
| Numatomas bendras modelio dydis (MB) | 1,32 |

Pasiekus geresnius nei buvo tikėtasi rezultatus, nuspręsta modelį supaprastinti kuriant antrą jo variantą. Modelyje pakeistas konvoliucinių ir suminių sluoksnių kiekis (3.2.4 pav.) bei optimizavimo algoritmas. Iš turėtų 3 konvoliucinių ir suminių sluoksnių grupių liko viena.




**3.2.4 pav.** Mašininio mokymo modelio (Nr. 2) architektūra

Šis modelis apmokytas pritaikant *AdaDelta* optimizavimo algoritmą, su mokymo greičiu 0,001 per 120 epochų. Mokymas truko 1 valandą 26 minutes, taigi sumažėjo daugiau nei per pus. Jau 120 epochoje mokymo tikslumas siekė 95% (iš kurių 99% atstumui ir 95% kampui), o validavimo tikslumas 72% (iš kurių 93% atstumui ir 72% kampui). Modelio mokymo paklaidos laike matomos diagramoje (3.2.5 pav.).




**3.2.5 pav.** Mašininio mokymo modelio (Nr. 2) treniravimo rezultatų diagrama

Sukurtas modelis išsaugojamas panaudojimui. Modelio savybės matomos 3.2.2 lentelėje.

**3.2.2 lentelė** Mašininio mokymo modelio (Nr. 2) savybės

| | |
|---|---|
| Bendri parametrai (vnt.) | 2172264 |
| Apmokymo parametrai (vnt.) | 2172264 |
| Neapmokomi parametrai (vnt.) | 0 |
| Įvesties dydis (MB) | 0,05 |
| Priekinės ir atgalinės sklaidos dydis (MB) | 0,60 |
| Parametrų dydis (MB) | 8,29 |
| Numatomas bendras modelio dydis (MB) | 8,94 |

### 3.2.4. Modelių palyginimas

Modeliai abu įvertina roboto poziciją norima paklaida. Tačiau vienas apmokomas greičiau nei kitas, paaukojant kampo tikslumą. Ar tai turės daug įtakos vairavimo kokybėje priklausys nuo roboto valdymo algoritmo. Dalį netikslumų roboto algoritmas gali traktuoti kaip triukšmus ir juos išfiltruoti.

Pirmasis modelis puikiai atlieka užduotį pagal numatytus reikalavimus. Validavimo tikslumas 92%, daugiau nei dvidešimt procentų viršina užsibrėžtą ribą. Puikiai aptinkamas ne tik kelio kampas, bet ir atstumas iki kelio centro. Konvoliucinio tinklo vykdymas vidutinio galingumo kompiuteryje trunka pakankamai neilgai, vidutiniškai vos 1,054ms, o modelio dydis ~1,3MB šiais laikais labai nedidelis.

Antrasis modelis, yra mažesnis dvejais konvoliuciniais ir sumavimo sluoksniais. Deja, bet nors konvoliucinis sluoksnis neturi daug parametrų, jo konvertavimas į tiesinį sluoksnį sugeneruoja daugiau parametrų. Taip modelio dydis išauga iki ~9MB. Nepriklausomai/Tesiogiai priklausomai nuo dydžio, modelio vidutinis vykdymo laikas yra 0,578ms. Sumažinant tinklą gaunamas mažesnis

modelio tikslumas, kai validavimo tikslumas 120 epochoje tik 72%. Verta pastebėti tai, kad nuokrypis nuo juostos centro yra aptinkamas 93% tikslumu, o kampo aptikimas siekia vos 72%. Jei tai neturės daug įtakos galutiniam važiavimo tikslumui, modelis gali būti taip pat sėkmingai naudojamas kaip ir pirmasis.

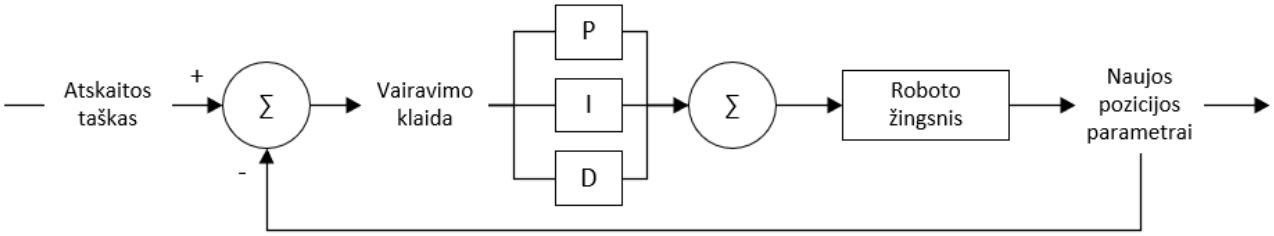
Pirmasis modelis yra gerokai tikslesnis už antrąjį ir daug tiksliau aptinka kampą tarp roboto ir kelio. Tuo tarpu yra ilgiau apmokomas ir naudoja daugiau sudėtingesnių skaičiavimų, dėl didesnio konvoliucinių modelio sluoksnių kiekio. Antro modelio vykdymo laikas vidutinio galingumo kompiuteryje per pus mažesnis. Nepaisant to, kad antrasis modelis turi daugiau parametrų, atliekami paprastesni skaičiavimai laidžia metodui greičiau gražinti rezultatą. Taigi pasirinkimas priklauso nuo norimos greitaveikos, norint greičiau veikiančio medelio tačiau mažiau tikslaus reiktų rinktis antrąjį modelį, o norint tikslumo, paaukojant greitaveiką naudojamas pirmasis modelis.

## 3.3. Roboto valdymas

### 3.3.1. Tyrimo aprašymas

Aptikus roboto poziciją kelyje, reikia nuspręsti kokio veiksmo reikia imtis, kad robotas toliau važiuotų keliu. Pats paprasčiausias sprendimas yra pagal atstumą ir kampą, nuspręsti į kurią pusę sukti ir visad pasukti vienodai. Tai paprastas tačiau neefektyvus sprendimas, nes pasiekus vidurio liniją bus vis sukama į priešingą pusę.

Geresnis pasirinkimas yra naudoti PID kontrolerį. Jis geba pasverti klaidą ir į ją reaguoti atitinkamai. Bendras PID kontrolerio (3.3.1 pav.) veikimo principas paremtas grįžtamojo ryšio įtaka sekančiam sprendimui. Valdiklyje yra trys parametrai kurie atitinkamai nusako, kaip bus naudojamos atskiros valdiklio dalys: p - proporcinė, i - integralinė ir d – diferencialinė.




**3.3.1 pav.** PID reguliatorius

Jei nors viena iš verčių lygi 0, tai ta dalis valdiklio nėra naudojama, taip galima sukonstruoti atskiras valdiklio parametrų kombinacijas, pvz.: P, PI, PID. Šiems valdikliams ir bus optimizuojami parametrai norint išgauti geriausią važiavimo tikslumą.

Roboto važiavimo kokybė gali pasirodyti pakankamai subjektyvus kriterijus, todėl privalu į jį pažvelgti objektyviai. Viena didžiausių važiavimo kokybės problemų yra, kad robotas esantis arti norimos trajektorijos, per daug suka tai į vieną tai į kitą pusę, todėl važiuojama ne pagal norimą trajektoriją o nuo jos vis nutolstamą į skirtingą pusę. Norint tai įvertinti, bus atsižvelgiama į roboto kampo pasikeitimą tarp žingsnių. Taip pat bus vertinamas nuokrypis nuo norimos trajektorijos, kurio, žinoma, visad tikimasi 0. Klaidos e skaičiavimas:

$$e = \frac{1}{n}\sum_{i=1}^{n} d_i + (\varphi_i - \varphi_{i-1}) \tag{7}$$

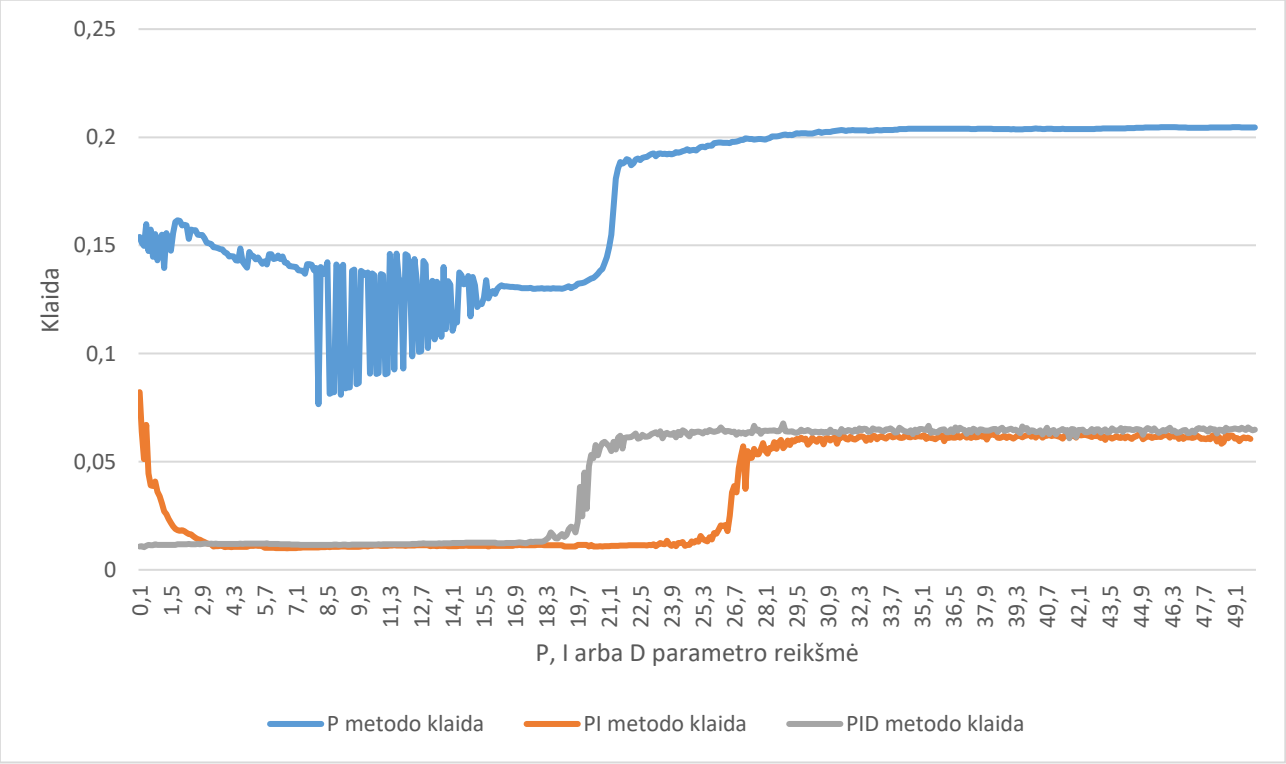
kai, $d_i$– atstumas iki juostos centro, $\varphi_i$– kampas tarp kelio ir roboto laiko žingsnyje $i$, $n$ – matuojamas laiko žingsnių skaičius.

Duomenys renkami, simuliacijoje paleidžiant robotą vis iš tos pačios vietos. Robotas maksimaliai gali atlikti 2000 žingsnių, kai vienas žingsnis yra priimto spendimo įgyvendinimas. Taip kiekviename žingsnyje yra įvertinamas roboto pasisukimo kampo pokytis ir atstumas iki kelio centro. Tas modelis kurio vidutinė klaida žingsniui bus mažiausia, objektyviai bus geriausias.

### 3.3.2. Parametrų optimizavimas

Turint vairavimo vertinimo funkciją buvo atliekamas PID koeficientų optimizavimas, naudojant gobšiąją tinklinę paiešką. Tinkle buvo naudojamas 0,1 žingsnis, su minimalia parametro reikšme 0, maksimalia 50, kai pradinės koeficientų vertės $P = 0{,}1$, $I = 0$, $D = 0$. Kadangi naudojama gobšioji paieška, tai surastas geriausias P koeficientas bus naudojamas geriausiam I metodui išrinkti, o geriausi PI parametrai bus naudojami D koeficiento radimui.

Atlikus tyrimą, buvo gautos skirtingos klaidos priklausomai, nuo naudojamų parametrų. Diagramoje (3.3.2 pav.) matyti, kad pirmasis roboto valdymo modelis P, turi didžiausią klaidą. Taip yra todėl, kad visi sprendimai drastiški ir nekompensuojami. Naudojant PI modelį, gaunami daug geresni rezultatai, nes atsižvelgiama į klaidos taisymo greitį, tai atlieka PI modelio integralinė dalis. Trečias modelis PID, grąžina panašius rezultatus kaip ir antrasis, todėl reiks rinktis tarp vieno iš šių metodų.




**3.3.2 pav.** P, PI ir PID modelių klaidos diagrama

### 3.3.3. Metodų palyginimas

Visos bandytos PID kontrolerio modelių kombinacijos gebėjo valdyti robotą taip, kad šis neišvažiuotų iš kelio. Pozicijos duomenys, kad neįsiveltų papildomų klaidų buvo naudojami tiesiai iš simuliatoriaus. Dažnai pramonėje pastebima, kad PID kontrolerio parametrai gali būti lygūs nuliui. Taip suformuojami P, PI, PID ir panašūs modeliai, o dažniausiai pasitaikantis - PI. Tyrime suformavus šiuos tris skirtingus modelius, pastebėjome, kad taip pat PI modelis, nors ir nedideliu atotrūkiu yra pranašiausias.

Geriausias $P$ modelio koeficientas yra $8{,}1$. PI modelyje geriausias $I$ koeficientas yra $6{,}9$. PID modelyje geriausias $D$ parametras yra $0{,}3$. Visa tai matome mažiausių modelio klaidų lentelėje (3.3.1 lentelė). Akivaizdu, jog geriausias pasirinkimas valdyti robotui yra gautas $PI$ modelis, kai $P = 8{,}1$, o $I = 6{,}9$.

**3.3.1 lentelė** Mažiausios modelio klaidos

| **Modelis** | **P** | **I** | **D** | **Klaida** |
|---|---|---|---|---|
| P | 8.1 | 0.0 | 0.0 | 0.07658 |
| **PI** | **8.1** | **6.9** | **0.0** | **0.00995** |
| PID | 8.**1** | 6.9 | 0.3 | 0.01047 |

Robotui valdyti pagal pozicijos duomenis gaunamus iš virtualios aplinkos mažiausiai klaidos generuojantis modelis yra PI modelis. Jei turimo metodo nepakanka, modelį dar tiksliau optimizuoti būtų galima pasitelkiant, išsamesnę klaidos vertinimo funkciją, bei tiriant daugiau verčių. Naudojant godžiąją paiešką, lieka neištirtų P, I ir D koeficientų. Galbūt būtų galima tirti ir kitas kombinacijas, tokias kaip PD ar ID, nors ir tai nėra dažnai taikoma praktikoje.

## 4. Roboto valdymo aptinkant jo poziciją kelyje eksperimentas

Atliktas tyrimas pateikė keletą skirtingų būdų roboto pozicijos išskyrimui, bei roboto valdymui. Tai atskiros komponentės iš kurių sujungiama transporto priemonės valdymo sistema. Norint išbandyti, kurių metodų kombinacija geriausiai valdys robotą, privalu atlikti eksperimentą.

### 4.1. Eksperimento tikslas

Šio eksperimento metu, bus lyginamos skirtingų modelių grupės, taip nustatant kokia grupė geriausiai geba valdyti robotą. Taip galima atlikti mokslu paremtą sprendimą, ir geriausia metodų grupe valdyti robotą kelyje su mažiausia vairavimo klaida.

### 4.2. Eksperimento aprašymas

Turimi skirtingi valdymo ir pozicijos aptikimo modeliai bus sugrupuojami ir bandomi kartu. Bus sudarytos šešios grupės kaip pateikta 4.2.1 lentelėje. Nors individualiai jau žinomi geriausi metodai, tačiau valdymo modeliai gali skirtingai reaguoti į pozicijos aptikimo klaidą ir paprasčiausias būdas išrinkti geriausią modelių kombinaciją – eksperimentas. Remiantis atliktu tyrimu tikimasi, kad geriausia grupė bus 2.

**4.2.1 lentelė** Pozicijos aptikimo ir roboto valdymo metodų grupės

| **Grupės indeksas** | **Pozicijos aptikimas** | **Valdymas** |
|---|---|---|
| 1 | Mašininio mokymo modelis Nr. 1 | P |
| 2 | Mašininio mokymo modelis Nr. 1 | PI |
| 3 | Mašininio mokymo modelis Nr. 1 | PID |
| 4 | Mašininio mokymo modelis Nr. 2 | P |
| 5 | Mašininio mokymo modelis Nr. 2 | PI |
| 6 | Mašininio mokymo modelis Nr. 2 | PID |

Išbandžius visus galimus variantus, bus apskaičiuojamos vairavimo klaidos kiekvienai modelių grupei. O pagal mažiausią klaidą išrenkamas geriausias modelis. Reiktų pastebėti, kad vairavimo klaidai skaičiuoti bus naudojamai simuliatoriaus duomenys, taip išgaunama klaida bus tiksliausia.

### 4.3. Eksperimentas ir jo rezultatai

Atliekant eksperimentą buvo naudojama ta pati klaidos vertinimo funkcija, kaip ir tyrimo metu. Todėl galime palyginti eksperimento rezultatus, su tyrimo rezultatais. Tyrime, pozicijos išskyrimas buvo atliekamas simuliatoriaus, todėl paklaida buvo nulinė.

Eksperimento vykdymo metu, robotas važiavo 2000 žingsnių 5 kartus iš tos pačios vietos. Bandoma buvo kelis kartus, nes šiuo atveju, turime pozicijos išskyrimo paklaidą, kuri daro įtaką bandymo rezultatams. Vidutinė pirmojo modelio aptikimo paklaida visų bandymų metu, naudojant skirtingus valdymo kontrolerius matoma 4.3.1 lentelėje, o antrojo 4.3.2 lentelėje. Kaip ir tikėtasi, antroji grupė gražino geriausią rezultatą. Tai parodo tyrimo ir eksperimento rezultatų pasikartojimą. Taip dar kartą patvirtiname, kad robotą geriausiai valdo PI modelis.

**4.3.1 lentelė** Grupių 1, 2 ir 3 bandymai

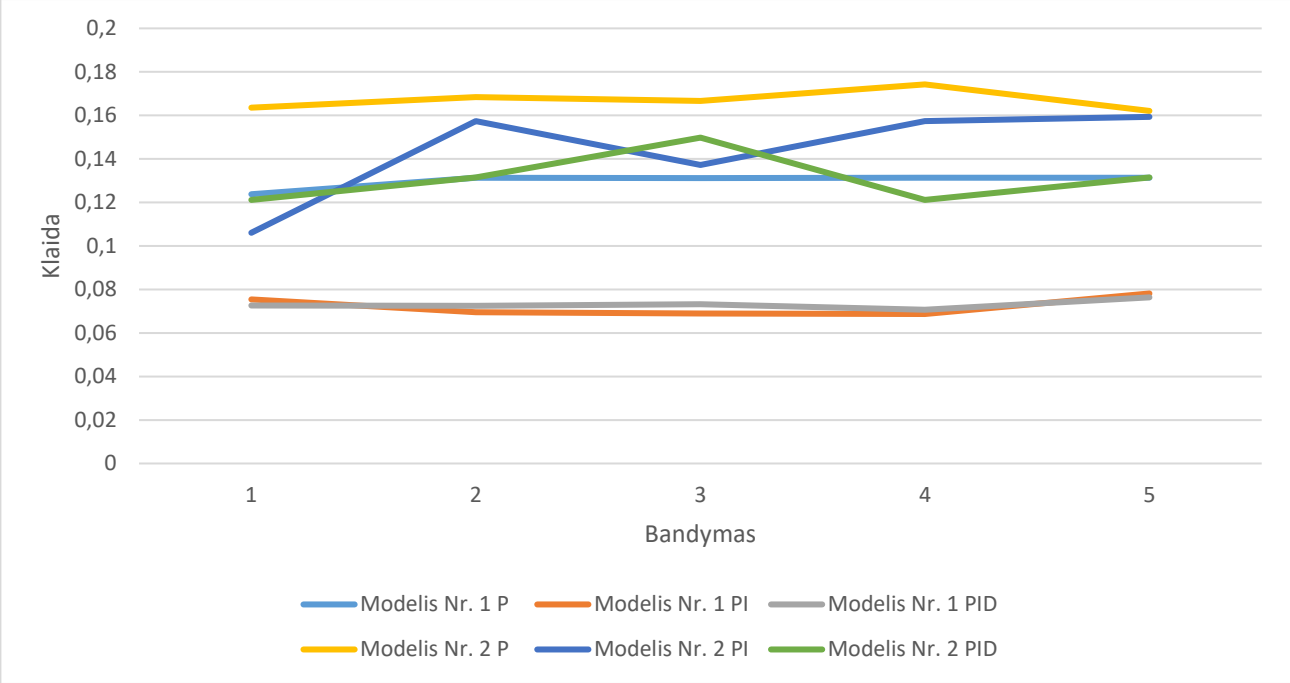
| **Bandymas** | **Modelis Nr. 1 P** | **Modelis Nr. 1 PI** | **Modelis Nr. 1 PID** |
|---|---|---|---|

| | | | |
|---|---|---|---|
| 1 | 0,123765452 | 0,075389608 | 0,072630176 |
| 2 | 0,131344124 | 0,069478063 | 0,072422005 |
| 3 | 0,131199155 | 0,068879553 | 0,07317269 |
| 4 | 0,131356432 | 0,068712611 | 0,070670175 |
| 5 | 0,131283317 | 0,07815407 | 0,076363974 |
| Vidutinė klaida | 0,129789696 | 0,072122781 | 0,073051804 |

**4.3.2 lentelė** Grupių 3, 4 ir 5 bandymai

| **Bandymas** | **Modelis Nr. 2 P** | **Modelis Nr. 2 PI** | **Modelis Nr. 2 PID** |
|---|---|---|---|
| 1 | 0,163610801 | 0,106045919 | 0,121154522 |
| 2 | 0,168459728 | 0,157364043 | 0,131539599 |
| 3 | 0,166665276 | 0,137154199 | 0,149783986 |
| 4 | 0,174246839 | 0,157371847 | 0,121154522 |
| 5 | 0,162066336 | 0,159323077 | 0,131539599 |
| Vidutinė klaida | 0,167009796 | 0,143451817 | 0,131034446 |

Grafike (4.3.1 pav.) matome aiškią pirmosios ir antrosios grupės atskirtį. Detaliau analizuojant duomenis matoma, kad tokią įtaką klaidai turėjo roboto drebėjimas. Robotas pradeda drebėti dėl pozicijos aptikimo paklaidos. Ši paklaida klaidina valdymo kontrolerį, ir taip gaunama vis skirtinga klaida. Pirmoje ir antroje grupėje į šią paklaidą kontroleris reaguoja atlaidžiau.




**4.3.1 pav.** Pozicijos aptikimo ir valdymo paklaidos bandymų metu

Lyginant tyrime gautus kontrolerio duomenis su eksperimento duomenimis (4.3.3 lentelė), galime įvertinti grubią pozicijos aptikimo paklaidą. Simuliatoriaus duomenis, laikant kaip etalonu, jas galime atimti iš gautų eksperimento rezultatų. Tai preliminarus vertinimas, tačiau galime pastebėti, kad

paklaida teigiama. Iš to sprendžiame, kad robotas valdomas mažesne paklaida, kai naudojami tikrieji pozicijos duomenys, o ne iš mašininio mokymo modelio.

**4.3.3 lentelė** Tyrimo ir eksperimento duomenų palyginimas

| **Pozicijos išskyrimas / Valdymo metodas** | **Simulatoriaus duomenys** | **Modelis Nr. 1** | **Modelis Nr. 2** |
|---|---|---|---|
| P | 0.07658 | 0,129789696 | 0,167009796 |
| PI | 0.00995 | **0,072122781** | 0,143451817 |
| PID | 0.01047 | 0,073051804 | 0,131034446 |

Atliktas eksperimentas parodo, kad robotas gali būti valdomas PID kontroleriu ar jo modifikacijomis. Kaip ir tikėtasi, geriausiai veikia PI modifikacija, be diferencialinės dalies. Pozicijos aptikimas didina vairavimo klaidą. Norint gauti geresnius rezultatus, reikėtų tobulinti vairavimo klaidos skaičiavimo funkciją, bei PID kontrolerį optimizuoti kiekvienam pozicijos aptikimo modeliui atskirai. Dabar robotas geba važiuoti keliu nenuklysdamas, tačiau nukrypstant nuo trajektorijos. Realiomis sąlygomis toks nukrypimas gali būti kritinis, todėl sistema dar nepasiekia tokio lygio, kad galėtų dalyvauti eisme su kitais eismo dalyviais, tačiau gera žinia tai, kad metodai veikia, juos tereikia geriau optimizuoti.

Bendras valdymo ir pozicijos modelis pasiteisina. Roboto pozicija aptinkama dideliu patikimumu, o analizė netrunka ilgai. Mašininio mokymo modelis geba prisitaikyti prie skirtingų nuotraukų su skirtingais triukšmais, o PID reguliatorius sėkmingai optimizuoja kelionės trajektoriją link norimos.

# 5. Tobulinimas, galimi tolesni darbai

Nors projekto veikla šiuo metu stabdoma, tema neišsemta. Atlikus tyrimus matomos vietos, kur galima patobulinti valdymo algoritmą, spręsti daugiau problemų ir artėti prie aukštesnio autonomijos lygio.

## 5.1. Progresas simuliatoriuje

Duckietown simuliatorius siūlo plačias galimybes, todėl sistemą nesunkiai galima tobulinti virtualioje aplinkoje ir toliau. Kuo labiau ištobulintas algoritmas virtualiai, tuo mažiau problemų iškils realioje aplinkoje.

### 5.1.1. Pozicijos aptikimo tobulinimas

Pozicijos aptikimas simuliatoriuje veikia pakankamai dideliu tikslumu, tačiau reliatyviai paprastomis sąlygomis. Turint pradinį modelį, jį būtų galima toliau mokyti kaupiant sudėtingesnius duomenis. Tokie duomenys, turėtų daugiau šešėlių, kelio linijų nusitrynimus, nedidelius apvažiuojamus objektus kelyje ir pan. Iš tyrimo duomenų panašu, kad modelis tai gebėtų aptikti, tačiau mokymas sudėtingėja.

Pirmasis žingsnis, su skirtingais apšvietimais būtų pakankamai nesudėtingas, reikėtų anotuotų nuotraukų, tokių kokios buvo naudojamos ir šiame tyrime. Anotuojant vaizdus objektais ant kelio tektų numatyti trajektorijos pasikeitimus atsirandant naujiems objektams.

### 5.1.2. Roboto valdymo tobulinimas

Roboto valdymas pakankamai neblogas, tačiau ne idealus. Norint pasiekti geresnį važiavimo tikslumą, patartina išbandyti visas įmanomas PID modelio koeficientų kombinacijas. Kiekvienas iš pozicijos nustatymo modelių turėtų turėti atskirą PID modifikaciją.

Pastebint klaidas specifiniais atvejais, galima jas spręsti individualiai. Pavyzdžiui, turint didelį nuokrypį nuo trajektorijos pirmiausia atsisukti tiesiai į trajektoriją ir tik tada pradėti važiavimą. Matant kad vairavimo klaida nemažėja, stabdyti robotą ir pan.

Roboto valdyme visad buvo naudojamas vienodas greitis. Tobulinant roboto valdymą rekomenduojama valdyti ne tik posūkio kampą, bet ir roboto važiavimo greitį.

### 5.1.3. Papildomos taisyklės

Pasirodo važiuoti keliu savo kelio pusėje, nenuklystant nuo kelio yra pakankamai sudėtingas uždavinys. Tačiau jį įveikus, galima pridėti papildomų roboto valdymo taisyklių: atpažinti ir saugoti pėsčiuosius, atpažinti ir saugotis bei saugoti kitas TP, atpažinti ir paklusti šviesoforui, atpažinti kelių ženklus ir jais kliautis ir pan. Visi šie uždaviniai lyginant su vairavimu yra antraeiliai, bet svarbūs. Tokių taisyklių pridėjimas didina sistemos autonomiškumo lygį.

### 5.1.4. Navigacija

Šiuo metu sutikęs sankryžą robotas važiuoja į bet kurį kelią atsitiktinai. Tai puiki proga tobulėti: iš anksto žinant žemėlapį galima numatyti norimą maršrutą ir važiuoti pagal jį. Taip sistema būtų pritaikoma kelionei iš taško A į tašką B. Realiame gyvenime, tai kritinis reikalavimas, be jo savavaldė sistema neneša realios naudos.

## 5.2. Reali aplinka

Atlikus pakankamai bandymų virtualioje aplinkoje, galima pasiruošti realiai aplinkai. Tikėtina, jog apmokytas modelis simuliatoriuje galėtų veikti ir realybėje, tačiau žinoma su tam tikrais vaizdo išlyginimais. Simuliatoriuje gaunamas vaizdas tolygus ir tvarkingas, o realiomis sąlygomis deja ne.

Tinkamai paruošus robotą ir kamerą padėjus į tokią pat poziciją kaip simuliatoriuje, būtų galima atlikti bandymus ir nuspręsti ar sistema geba teisingai išskirti poziciją. Jei vis dėl to pozicijos išskirti nepavyktų, problema sudėtingėja. Tokiu atveju reiktų papildomos kameros virš fizinio žemėlapio kuriame važiuos robotas. Žiūrint iš viršaus ir atpažįstant robotą, būtų gaunamos realios jo pozicijos koordinatės, kurios būtų naudojamos anotavimui. Anotuotais vaizdais reiktų papildomai apmokyti modelį.

## Išvados

1. Gilinantis į kitų autorių darbus pastebėta, su kokiomis problemomis tenka susidurti: signalų filtravimas, klasifikavimas, efektyvus kelio ir aplinkos segmentų išrinkimas. Šios problemos supaprastintos naudojant virtualią aplinką.
2. Sistemos kaina gali greitai kilti, naudojant daug brangių sensorių, taigi reikia pagal tam tikrą biudžetą pasirinkti tik tokį biudžetą tenkinančius sensorių kiekius. Ši įranga dėvisi, todėl pirmiausia roboto tyrimai atliekami simuliacijoje.
3. Didžioji dalis aplinkos stebėjimo gali būti atlikta su kamera, tačiau kamera negali matuoti atstumo, tam naudojami kiti jutikliai.
4. Autonominis transporto priemonių judėjimas yra labai sudėtinga užduotis ir norint koncentruotis tik į šią užduotį, reikalinga supaprastinta aplinka. Supaprastintoje aplinkoje sprendžiamos tik su vaizdų atpažinimu ir autonominiu judėjimu susiję problemos.
5. PID kontroleris geba valdyti robotą išlaikant norimą kelionės trajektoriją. Šio modelio naudojimas optimizuoja važiavimo klaidą, o geriausiai veikianti konfigūracija yra PI.
6. Didesnis ir sudėtingesnis mašininio mokymosi pozicijos aptikimo modelis nors ir dvigubai ilgiau vykdantis aptikimą grąžina tikslesnius rezultatus ir leidžia valdyti robotą mažesne vairavimo paklaida.
7. Naudojant LeNet modelį kaip prototipą, pasiteisino prielaida, kad modelis labai gerai optimizuotas vaizdų apdorojimui, nes mažinant modelį mažėja aptikimo tikslumas.
8. Sklandžiam trajektorijos sekimui naudojant PID kontrolerį išmokti jo veikimo principai bei optimizavimo metodai.
9. Tyrimas įrodė, jog dabartinėje virtualioje aplinkoje geriausiai veikia PI reguliatorius su pirmuoju mašininio mokymo pozicijos aptikimu, vidutinei žingsnio klaidai esant 0,07212.
10. Klaidai tarp simuliatoriaus duomenų ir geriausios valdymo grupės išaugant apie 7 kartus, daroma išvada, kad kiekvienas pozicijos aptikimo modelis turi turėti atskirai optimizuotą PID valdiklį.
11. Automatinis duomenų kaupimas padėjo surinkti didelį kiekį duomenų, tačiau ne visi duomenys teisingi. Visas nuotraukas tenka peržiūrėti rankiniu būdu, taip eliminuojant aiškiai matomus neatitikimus ar sunkiai atpažįstamus vaizdus.
12. Sukurta paprasta vartotojo sąsaja padeda valdyti simuliatorių ir atlikti su sistema norimus veiksmus, su minimalia patirtimi ir neilgu mokymosi procesu.

## Literatūros sąrašas


[1] S.-C. Chen, "Multimedia for Autonomous Driving," IEEE, 2019.

[2] M. Aldibaja, N. Suganuma and K. Yoneda, "Robust Intensity-Based Localization Method for Autonomous Driving on Snow–Wet Road Surface," IEEE, 2017.

[3] K. Jo, J. Kim, D. Kim, C. Jang and M. Sunwoo, "Development of Autonomous Car—Part II: A Case Study on the Implementation of an Autonomous Driving System Based on Distributed Architecture," IEEE, 2015.

[4] E. D. Dickmanns and V. Graefe, "Dynamic Monocular Machine Vision," in *Machine Vision and Applications*, Springer, 1988, pp. 223-240.

[5] B. R. Page, S. Ziaeefard, B. Moridian and N. Mahmoudian, "Learning autonomous systems — An interdisciplinary project-based experience," IEEE, 2017.

[6] V. . Ilkova ir A. . Ilka, „Legal aspects of autonomous vehicles — An overview,“ , 2017. [Tinkle]. Available: http://publications.lib.chalmers.se/records/fulltext/249781/local_249781.pdf. [Kreiptasi 22 11 2019].

[7] A. Jadhav, "Autonomous Vehicle Market by Level of Automation (Level 3, Level 4, and Level 5) and Component (Hardware, Software, and Service) and Application (Civil, Robo Taxi, Self-driving Bus, Ride Share, Self-driving Truck, and Ride Hail) - Global Opportunity Analy," Allied Market Research, 2019.

[8] Global Market Insights, Inc., "Autonomous Cars Market 2019 to Showing Impressive Growth by 2024 | Industry Trends, Share, Size, Top Key Players Analysis and Forecast Research - America News Hour," America News Hour, 27 11 2019. [Online]. Available: https://www.americanewshour.com/2019/11/27/autonomous-cars-market-2019-to-showing-impressive-growth-by-2024-industry-trends-share-size-top-key-players-analysis-and-forecast-research/131385/. [Accessed 29 11 2019].

[9] S. Campbell, N. O'Mahony, L. Krpalcova, D. Riordan, J. Walsh, A. Murphy and C. Ryan, "Sensor Technology in Autonomous Vehicles : A review," IEEE, 2018.

[10] C. Ilas, "Perception in autonomous ground vehicles," IEEE, 2013.

[11] J. Lee, Y.-A. Li, M.-H. Hung ir S.-J. Huang, „A Fully-Integrated 77-GHz FMCW Radar Transceiver in 65-nm CMOS Technology,“ IEEE, 2010.

[12] L. Xiaoming, Q. Tian, C. Wanchun ir Y. Xingliang, „Real-time distance measurement using a modified camera,“ IEEE, 2010.

[13] D. Wobschall, M. Zeng ir B. Srinivasaraghavan, „An Ultrasonic/Optical Pulse Sensor for Precise Distance Measurements,“ IEEE, 2005.

[14] W. Rahiman and Z. Zainal, "An overview of development GPS navigation for autonomous car," IEEE, 2013.

[15] J. Borenstein, H. R. Everett and L. Feng, Where am I? Sensors and Methods for Mobile Robot Positioning, The University of Michigan, 1996.

[16] S. Marukatat, Image enhancement using local intensity distribution equalization, Springer, 2015.

[17] OpenCV, „Histogram Equalization - OpenCV,“ OpenCV, 2014. [Tinkle]. Available: https://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/histogram_equalization/histogram_equalization.html#what-is-histogram-equalization. [Kreiptasi 22 11 2019].

[18] OpenCV, "Camera calibration with OpenCV - OpenCV," OpenCV, 2014. [Online]. Available: https://docs.opencv.org/2.4/doc/tutorials/calib3d/camera_calibration/camera_calibration.html. [Accessed 22 11 2019].

[19] L. Paull, J. Tani, H. Ahn, J. Alonso-Mora, L. Carlone, M. Cap, Y. F. Chen, C. Choi, J. Dusek, Y. Fang, D. Hoehener, S.-Y. Liu, M. Novitzky, I. F. Okuyama, J. Pazis, G. Rosman and Valer, "Duckietown: An open, inexpensive and flexible platform for autonomy education and research," IEEE, 2017.

[20] T.-K. Chuang, N.-C. Lin, J.-S. Chen, C.-H. Hung, Y.-W. Huang, C. Teng, H. Huang, L.-F. Yu, L. Giarre ir H.-C. Wang, „Deep Trail-Following Robotic Guide Dog in Pedestrian Environments for People who are Blind and Visually Impaired - Learning from Virtual and Real Worlds,“ įtraukta *ICRA*, 2018.

[21] J. Tani, L. Paull, M. Zuber, D. Rus, J. How, J. Leonard ir A. Cens, „Duckietown: An Innovative Way to Teach Autonomy,“ Springer International Publishing AG, 2017.

[22] Duckietown Foundation, "The Platform - DuckieTown," Duckietown Foundation non-profit, [Online]. Available: https://www.duckietown.org/about/platform. [Accessed 22 11 2019].

[23] Classical Robotics Architectures using Duckietown, Duckietown Foundation US, Inc, 2020.

[24] J. EKESUND, "Self-driving car," kth.diva-portal, 2016.