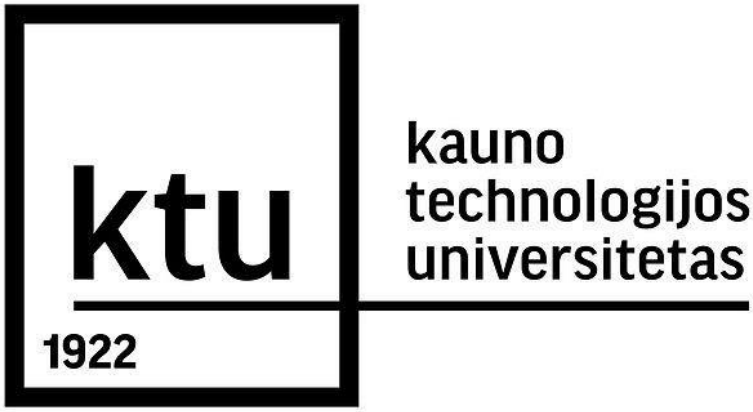


INFORMATIKOS FAKULTETAS

T125B158 Robotų programavimo technologijos

Antrųjų pratybų darbas

Studentas:
Eligijus Kiudys IFF-7/14

Dėstytojai:
Doc. Brūzgienė Rasa
Laurutis Remigijus

KAUNAS 2020

Turinys

Paveikslų sąrašas

1. Tikslas

V-REP(CoppeliaSim) aplinkoje pamodeliuoti roboto linijos sekimo algoritmą ir išanalizuoti roboto linijos sekimo veikimą skirtinguose trasose.

2. Darbo etapai

1. Atlikti linijos sekimo algoritmo modeliavimą ir tyrimą su skirtingomis trajektorijomis.
2. Aprašykite roboto elgseną (leiskite robotui įveikti trajektoriją mažiausiai 3 kartus).
3. Pakeiskite roboto veikimo algoritmą kad įveiktu skirtingas trajektorijas ir stebėkite roboto elgseną.

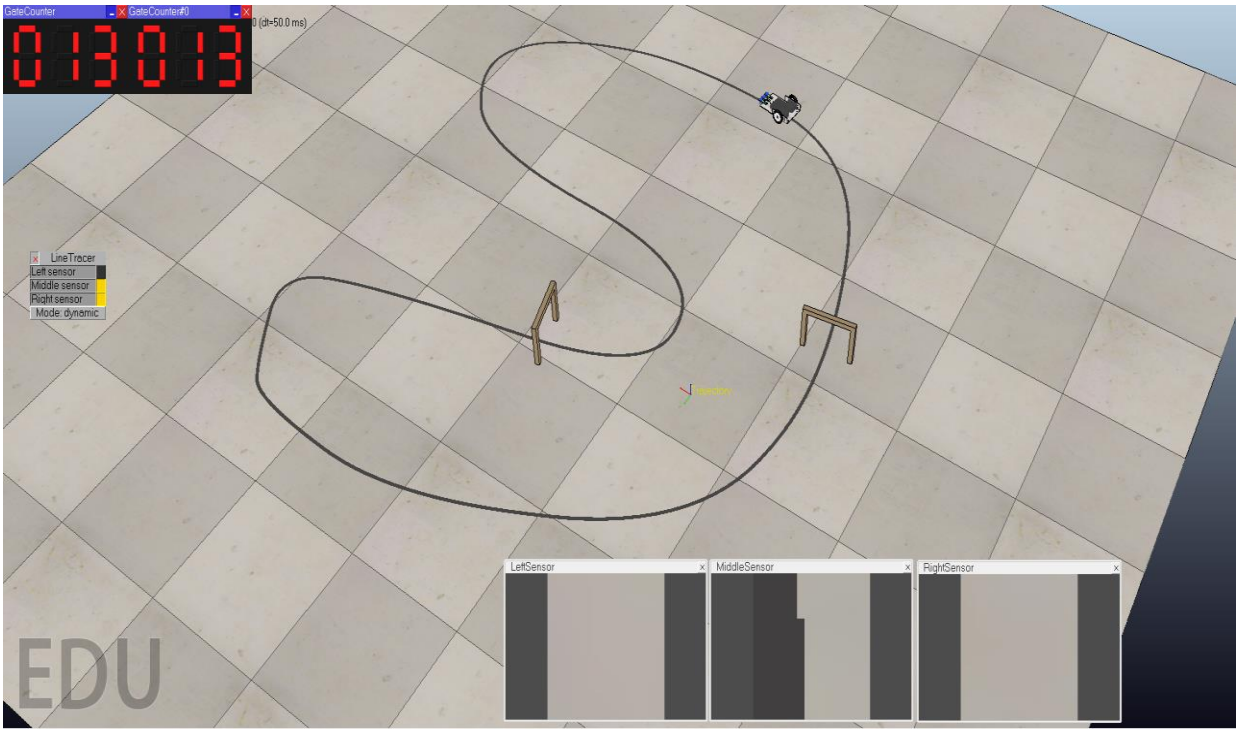
3. Užduoties Atlikimas

3.1. Trasos įveikimas ir greičio keitimas

Pasileidę duotą simuliacijos failą line_fallower matome duotą trasą ir robotą kuris važiuos duota trasa.

Pav. 1 Paleistas duotas failas.

Paleidus simuliaciją matome jog robotas pavažiuoja tiesiai ir tada pradeda sekti aptiktą trasą. Robotas trasą seka puikiai. Apačioje Pav. 2 matome kaip robotas apvažiavo trasą trylika kartų.



Pav. 2 Roboto linijos sekimas

Kadangi robotas apvažiavo duotą trasą puikiai pabandome pakeisti jo važiavimo greitį. Važiavimo greitį pakeitėme iš 0.3 į 0.6 nominalLiniearVelocity. Pakeitus greitį paleidžiame simuliaciją.

```
-- Initialization:
display=simGetUIHandle("sensorDisplay")
setLeds(display,false,false,false)
objHandle=sim.getObjectAssociatedWithScript(sim.handle_self)
result,robotName=sim.getObjectName(objHandle)
simSetUIButtonLabel(display,0,robotName)
lineTracer=sim.getObjectHandle("LineTracer")
leftSensor=sim.getObjectHandle("LeftSensor")
middleSensor=sim.getObjectHandle("MiddleSensor")
rightSensor=sim.getObjectHandle("RightSensor")
leftJoint=sim.getObjectHandle("LeftJoint")
rightJoint=sim.getObjectHandle("RightJoint")
leftJointDynamic=sim.getObjectHandle("DynamicLeftJoint")
rightJointDynamic=sim.getObjectHandle("DynamicRightJoint")
nominalLinearVelocity=0.6
wheelRadius=0.027
interWheelDistance=0.119
initialVehicleZpos=sim.getObjectPosition(objHandle,sim.handle_parent)[3]
previousSimulationTime=sim.getSimulationTime()
```

Pav. 3 Pakeistas roboto ratų greitis

Paleidžiame simuliaciją ir analizuojame ar yra pasikeitimu.

Pav. 4 Simuliacija kurioje roboto greitis pakeistas

Paleidus simuliaciją matome kaip Pav. 4 robotas pravažiuoja trasos liniją. Robotas šitą liniją spėja pamatyti bet nespėja prisitaikyti prie linijos dėl jo didelio greičio.

Atstačius greitį matome, kad robotas važiavo kaip ir prieš padidinant greitį. Įveikia trasą be problemų.

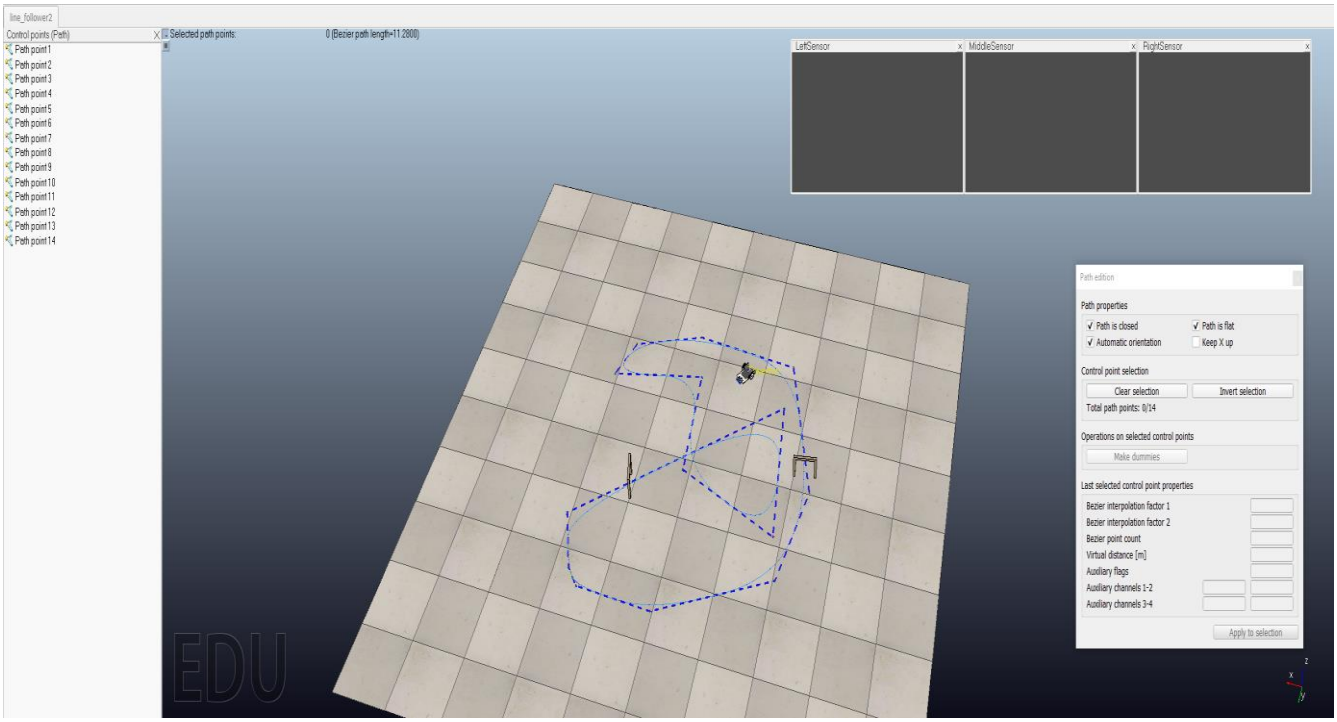
Pav. 5 Roboto važiavimas simuliacijoje su atstatytu greičiu

3.2. Trasos koregavimas ir kodo modifikavimas jos perėjimui

Turime pakeisti trasą taip, kad atsirastų trasoje kelio kilpa, bei smailesnis kampas. Taigi atliekame koregavimus pasirinkę „path“ objektą ir paspaudę kelio koregavimo mygtuką:

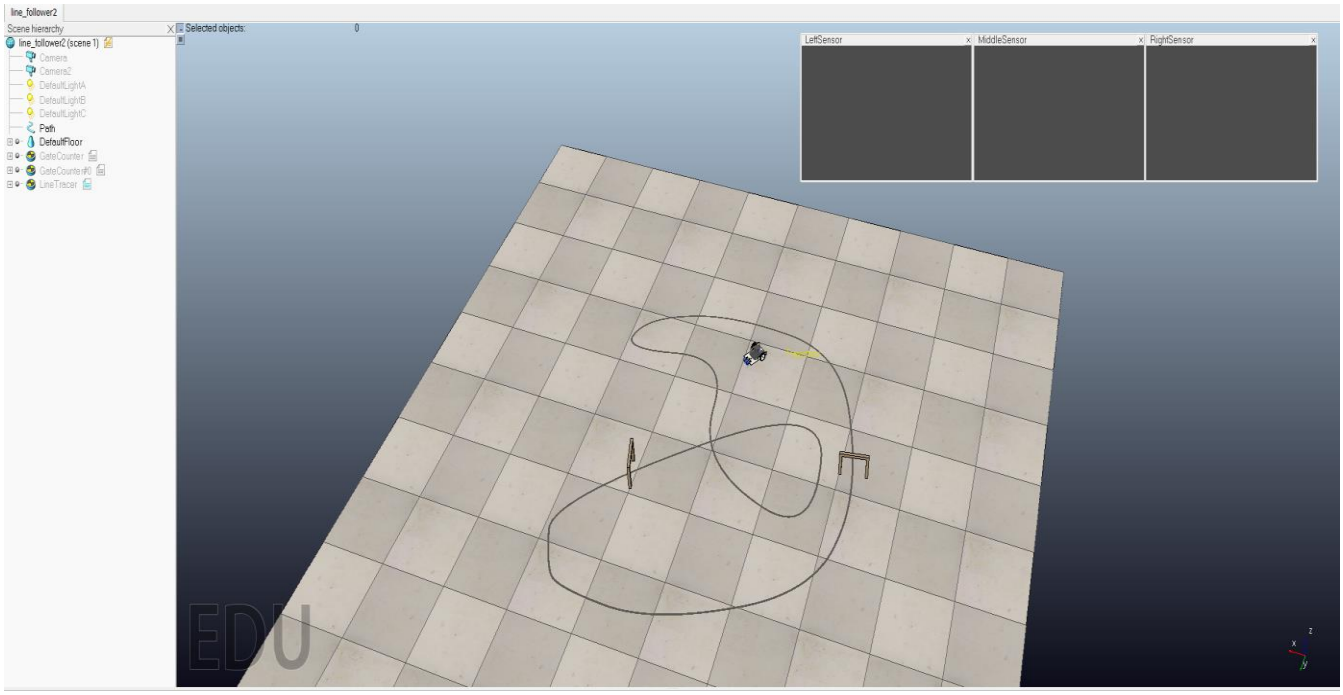
Pav. 6 Kelio konfigūravimo mygtukas

Pakoregavę kelią pagal nurodytus punktus gauname pakeistą kelią.



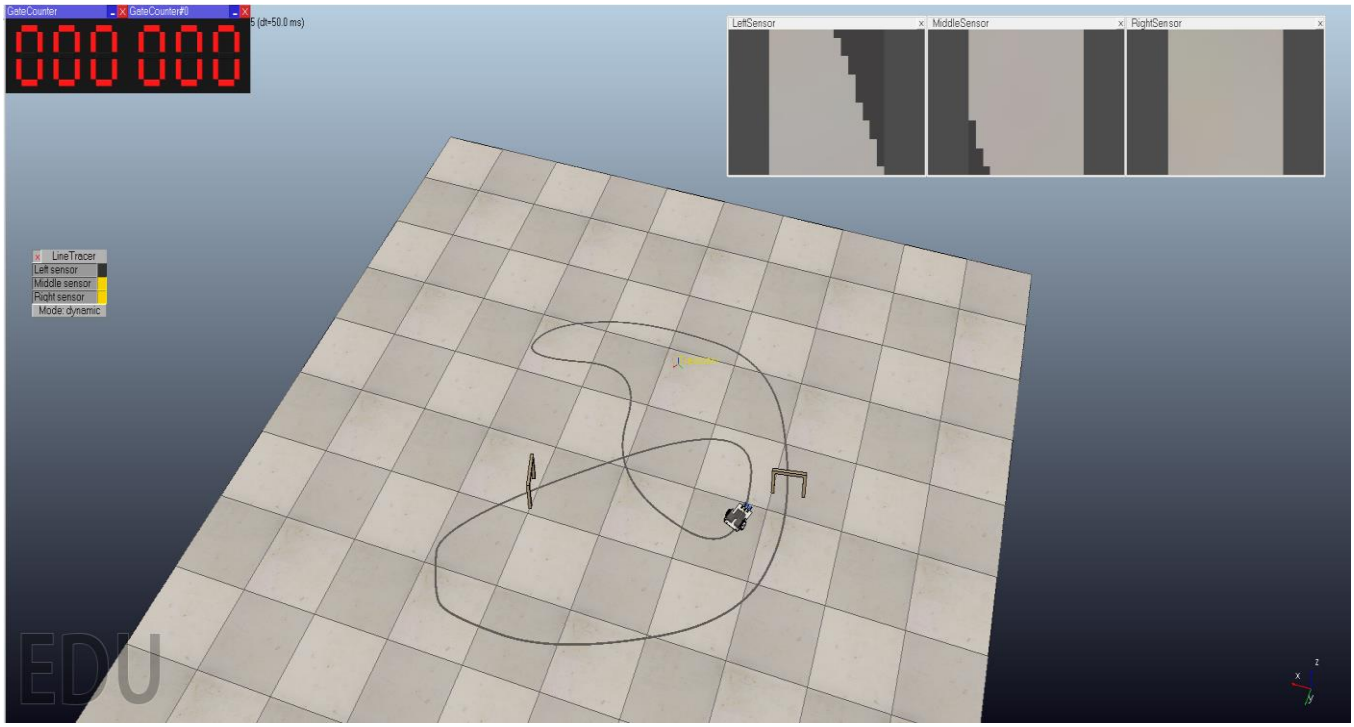
Pav. 7 Pakoreguotas kelias

Matome kaip atrodo sugeneruotas pakoreguotas kelias su smailesniu kampu ir kilpa.



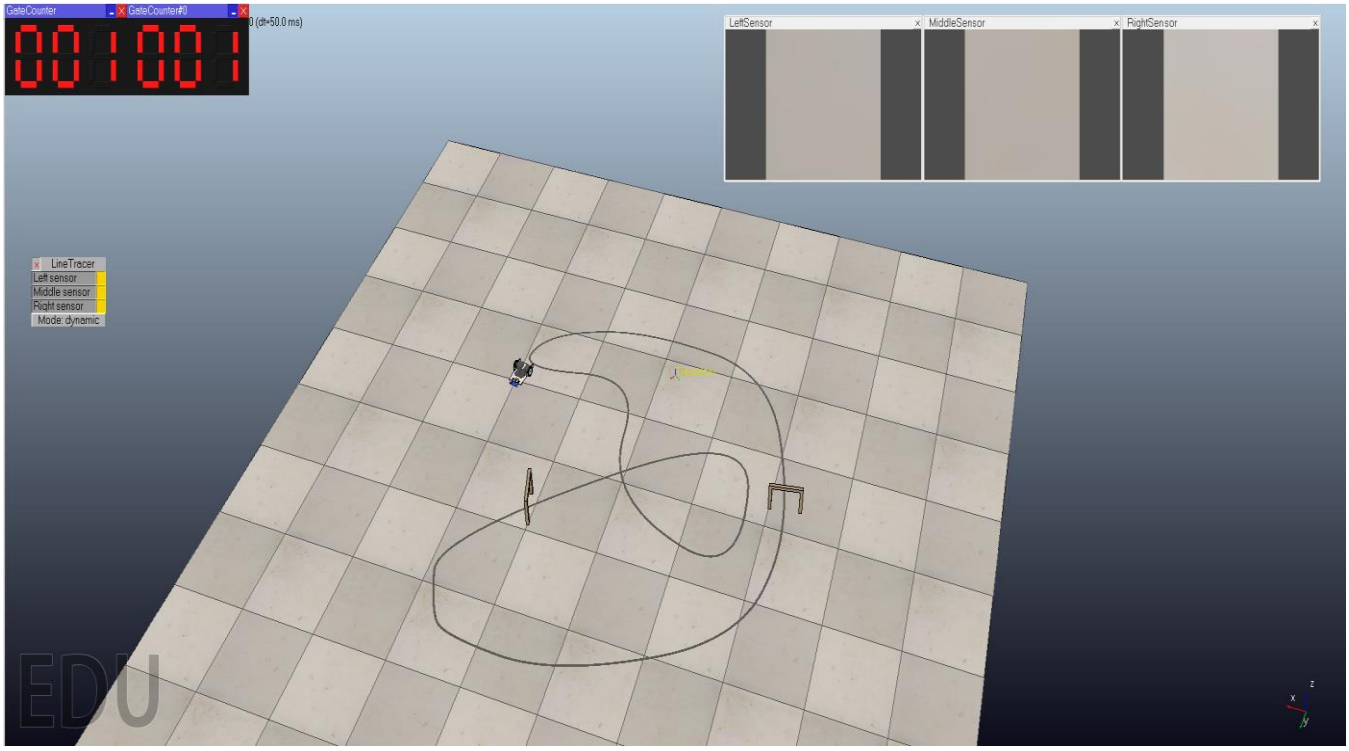
Pav. 8 Sugeneruotas kelias su kilpa ir smailiu kampu

Paleidus simuliaciją matome, kad robotas pravažiuoja kilpą be problemų.



Pav. 9 Roboto važiavimas kilpa

Robotui pravažiavus kilpą robotas važiuoja toliau. Robotui privažiavus smailesnį kampą ir bandant sekti smailiu kampu robotas išvažiuoja iš trasos.



Pav. 10 Robotas išvažiuoja iš trasos važiuojant smailiu kampu

Kadangi robotas nepravažiuoja smailaus kampo reikia kodo modifikacijos kuri padės staigiau pasisukti. Pirmiausia susikūriau kintamąjį kuris skirtas pasakyti ar robotas įvažiavo į trasą, jei robotas įvažiavo į trasą robotas gali atlikti kita paaiškintą veiksmą. Kai robotas važiuoja keliu ir kai nei vienas sensorius nebemato linijos robotas pradeda suktis į kairę pusę, kad robotas grįžtu į trasą. Grįžęs robotas į trasą važiuoja kaip važiavo iš pradžių.

Kode matome, kad pridėjau vieną papildomą sąlyga kurioje tikrinu ar robotas yra pradėjęs važiuoti keliu ir ar jo sensoriai kelio nebemato. Sensoriams kelio nematant robotas pradeda suktis į kairę pusę.

```
if (sensorReading[1] == false) then
    fallowLine = true
    linearVelocityLeft = linearVelocityLeft * 0.3

end

if (sensorReading[3] == false) then
    fallowLine = true
    linearVelocityRight = linearVelocityRight * 0.3


end


if (sensorReading[1] == true and sensorReading[2] == true and sensorReading[3] == true and fallowLine == true)
    fallowLine = true
    linearVelocityLeft = linearVelocityLeft * -0.3
    linearVelocityRight = linearVelocityRight * 0.3
end
```

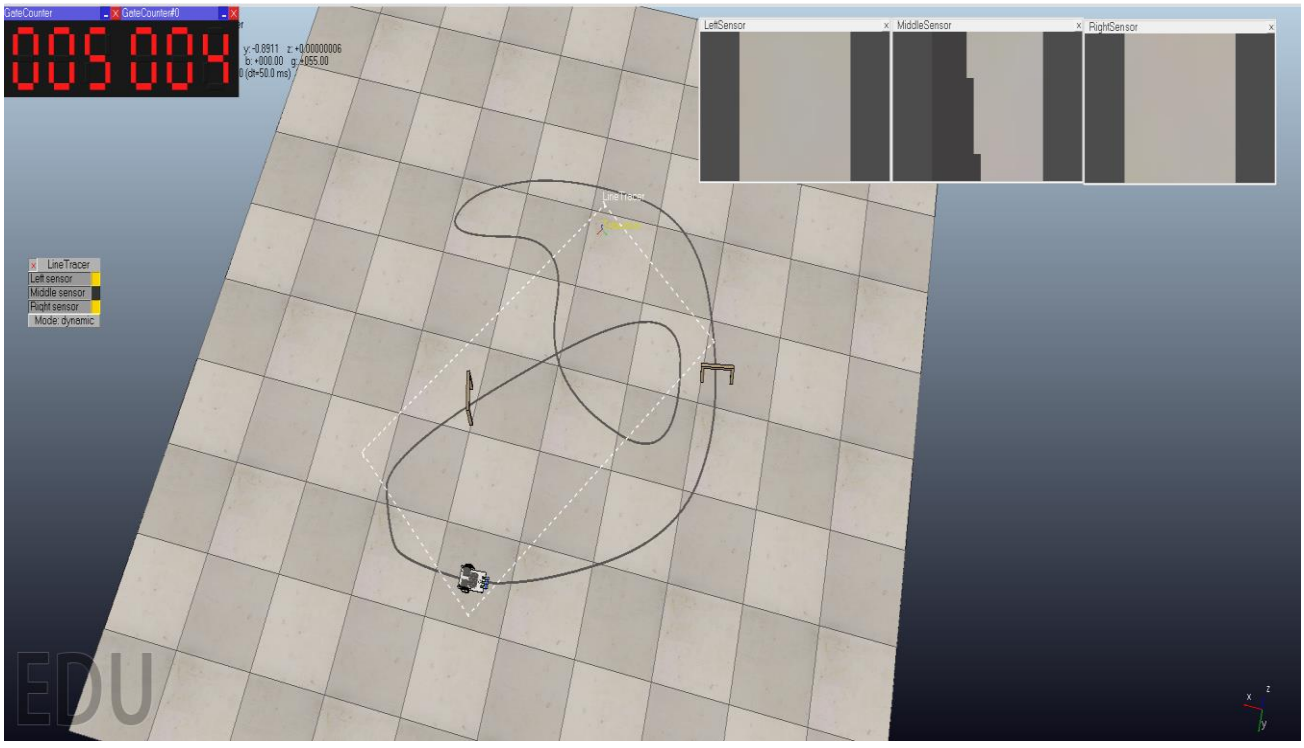
Pav. 11 Kodo modifikacija roboto pasisukimas

Inicijuotas kintamasis fallowLine skirtas patikrinti ar robotas pradėjo važiuoti keliu.

```
display=simGetUIHandle("sensorDisplay")
setLeds(display,false,false,false)
objHandle=sim.getObjectAssociatedWithScript(sim.handle_self)
result,robotName=sim.getObjectName(objHandle)
simSetUIButtonLabel(display,0,robotName)
lineTracer=sim.getObjectHandle("LineTracer")
leftSensor=sim.getObjectHandle("LeftSensor")
middleSensor=sim.getObjectHandle("MiddleSensor")
rightSensor=sim.getObjectHandle("RightSensor")
leftJoint=sim.getObjectHandle("LeftJoint")
rightJoint=sim.getObjectHandle("RightJoint")
leftJointDynamic=sim.getObjectHandle("DynamicLeftJoint")
rightJointDynamic=sim.getObjectHandle("DynamicRightJoint")
nominalLinearVelocity=0.3
wheelRadius=0.027
interWheelDistance=0.119
initialVehicleZpos=sim.getObjectPosition(objHandle,sim.handle_parent)[3]
previousSimulationTime=sim.getSimulationTime()
fallowLine = false
```

Pav. 12 Kodo modifikacija naujas kintamasis

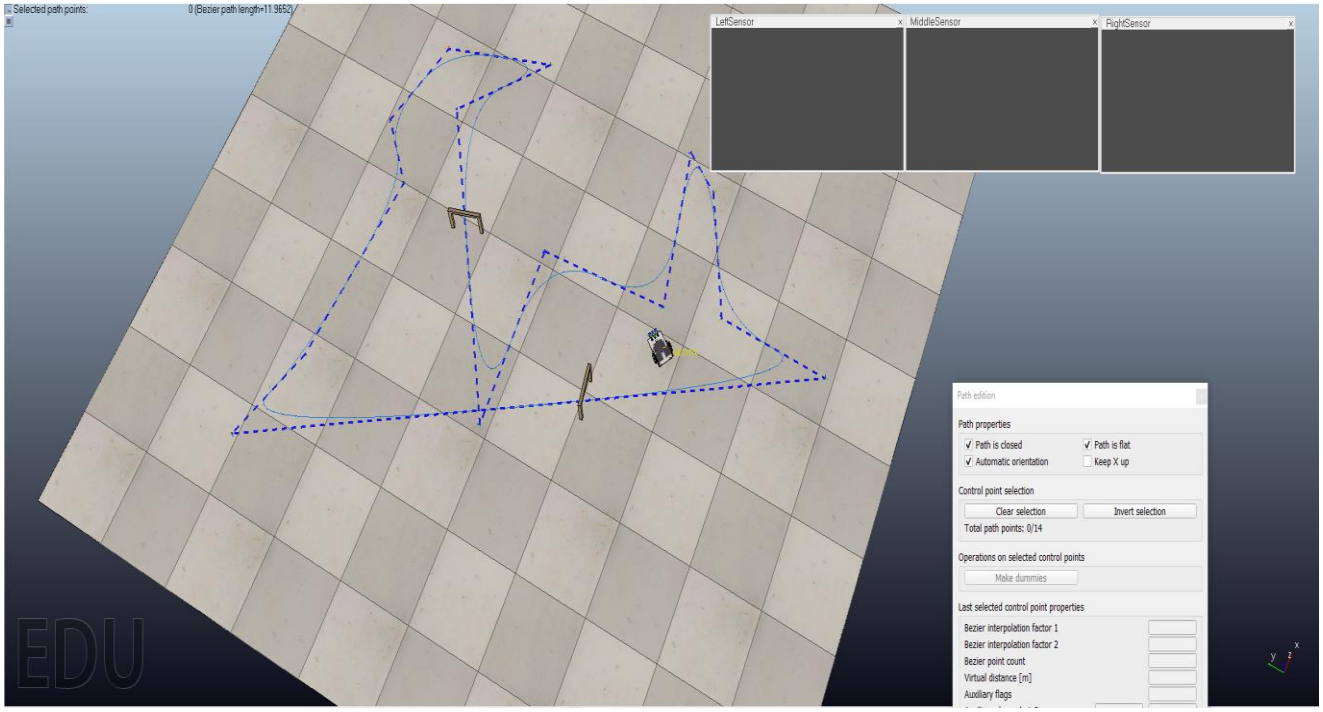
Padarius kodo modifikacijas robotas trajektoriją įveikia bent tris kartus.



Pav. 13 Robotas pravažiuoja kelią be problemų

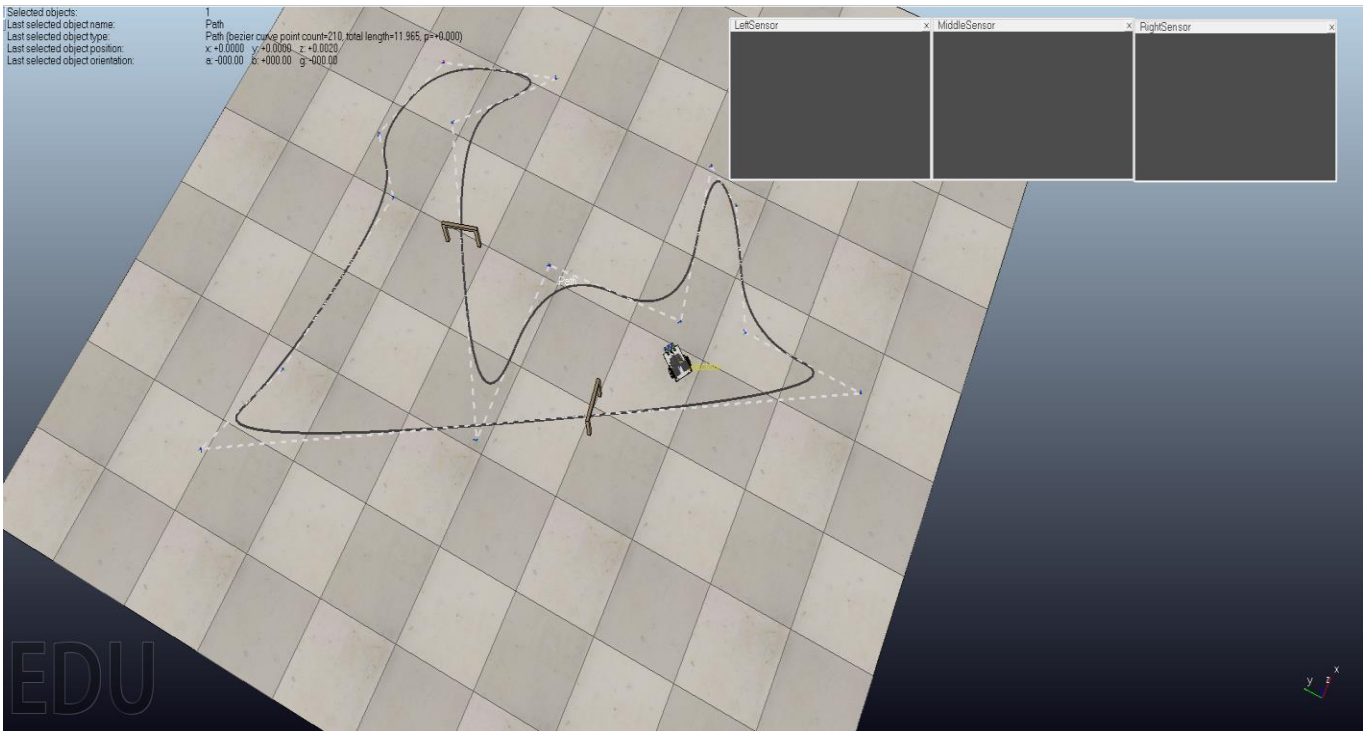
3.3. Sudėtingai koreguota trasa

Turime modifikuoti trasą taip, kad būtų trys, bei ypač smailūs posūkiai. Tą darome taip pat, kaip ir 3.2. punkte. Po modifikacijų trasa atrodo štai taip:



Pav. 14 Koreguota trasa su smailiais posūkiais

Pabandžius paleisti simuliaciją, robotas nesugeba pravažiuoti trasos naudojant praėjusio punkto kodą. Robotas suklysta ant vidinio posūkio, nes robotas per lėtai sukasi, robotui neišsukus posūkio jis išvažiuoja iš trasos apsisuka ir pradeda važiuoti atgal.



Pav. 15 Sugeneruota trasa su smailiais posūkiais

Redagavau kodą, kad robotas pasisuktu pagal paskutinį aptiktą sensorių. Jeigu robotas važiuoja ir nesugeba išsukti posūkio tada pagal paskutinį sensorių su kurio buvo aptiktas kelias robotas pasisuka į tą pusę kur kelias buvo aptiktas paskutinį kartą, robotas sukasi tol kol sensoriai iš naujo aptinka kelią. Susikūrėme du naujus kintamuosius kuriuose saugome ar vienas iš sensorių buvo paskutinis. Jeigu vienas iš sensorių buvo paskutinis tada pagal tą sensorių pradedame sukti robotą.

```
-- Initialization:
display=simGetUIHandle("sensorDisplay")
setLeds(display,false,false,false)
objHandle=sim.getObjectAssociatedWithScript(sim.handle_self)
result,robotName=sim.getObjectName(objHandle)
simSetUIButtonLabel(display,0,robotName)
lineTracer=sim.getObjectHandle("LineTracer")
leftSensor=sim.getObjectHandle("LeftSensor")
middleSensor=sim.getObjectHandle("MiddleSensor")
rightSensor=sim.getObjectHandle("RightSensor")
leftJoint=sim.getObjectHandle("LeftJoint")
rightJoint=sim.getObjectHandle("RightJoint")
leftJointDynamic=sim.getObjectHandle("DynamicLeftJoint")
rightJointDynamic=sim.getObjectHandle("DynamicRightJoint")
nominalLinearVelocity=0.3
wheelRadius=0.027
interWheelDistance=0.119
initialVehicleZpos=sim.getObjectPosition(objHandle,sim.handle_parent)[3]
previousSimulationTime=sim.getSimulationTime()
fallowLine = false
leftLast = false
rightLast = false
```

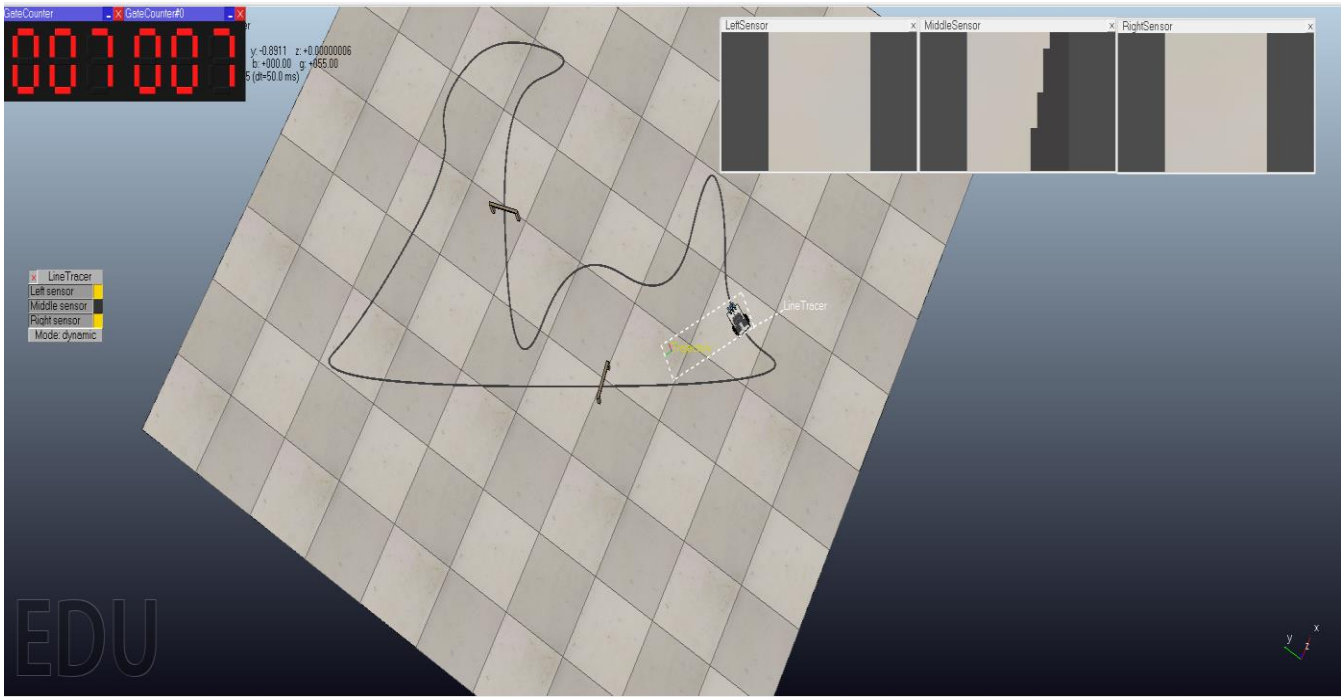
Pav. 16 Modifikuotas kodas nauji kintamieji

Roboto važiavimo pavyzdys. Robotas važiuoja tiesiai ir staiga turi pasisukti į kairę paskutinis sensorius su kuriuo aptiko kelią buvo kairys tai nei vienam sensoriui neaptinkant kelio robotas pradeda suktis į kairę.

```
    if (sensorReading[1] == false) then
        fallowLine = true
        leftLast = true
        rightLast = false
        linearVelocityLeft = linearVelocityLeft * 0.3
    end

    if (sensorReading[3] == false) then
        fallowLine = true
        rightLast = true
        leftLast = false
        linearVelocityRight = linearVelocityRight * 0.3

    end


    if (sensorReading[1] == true and sensorReading[2] == true and sensorReading[3] == true and fallowLine == true) then
        fallowLine = true
        if(leftLast == true or rightLast == false) then
            linearVelocityLeft = linearVelocityLeft * -0.3
            linearVelocityRight = linearVelocityRight * 0.3
        elseif (rightLast == true or leftLast == false) then
            linearVelocityLeft = linearVelocityLeft * 0.3
            linearVelocityRight = linearVelocityRight * -0.3
        end
    end
```

Pav. 17 Modifikuotas kodas pasukimas pagal paskutinį sensorių

Pamodifikavus kodą ir paleidus simuliaciją matome kaip robotas apvažiavo trasą šešis kartus ir septintą dar važiuoja. Vartai kurie skaičiuoja indikuoja kiek kartų robotas pravažiavo trasą nesuklystant. Robotas apvažiavo trasą be jokių sunkumų.



Pav. 18 Roboto važiavimas sudėtinga trasa.

4. Išvados

Pratybų darbo metu buvo išanalizuotas linijos sekimo algoritmas. Algoritmas buvo naudojamas keliose skirtingose trasose kurios skyrėsi sunkumo lygiu. Naudojant pirmąją duotą trasą keitėme algoritmo ratų greičio parametrą. Pakeitus parametrą išsiaiškinome, kad linijos sekimo algoritmas veikia tik su specifiniais parametrais. Pakeitus trasą į sunkesnę pamatėme, kad robotas nesugebėjo įveikti smailesnių posūkių, todėl algoritmas turėjo būti patobulintas. Patobulinus algoritmą robotas antrąją trasą su vienu smailiu posūkiu įveikė be problemų. Pakeitus trasą į kur buvo trys staigūs posūkiai kurie buvo smailūs reikėjo algoritmą dar kartą patvarkyti, nes robotas nesugebėjo apvažiuoti trasos. Dar kartą patvarkius algoritmą jis puikiai suveikė ir robotas įveikė sunkiausią trasą tris kartus. Nemodifikuotas algoritmas veikia puikiai tik trasose kurios yra be smailių posūkių. Norint naudoti šitą algoritmą kituose trasose reikia jį modifikuoti. Po kelių algoritmo modifikacijų jį pritaikėme smailių ir staigių posūkių trasoms kuriose algoritmas veikia puikiai.