МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное

учреждение высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

ИМ. Р.Е. АЛЕКСЕЕВА

ИНСТИТУТ РАДИОЭЛЕКТРОНИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Курс “Аппаратное и программное обеспечение роботизированных систем”

Отчет по лабораторной работе №2

Выполнили: Соков С.А.

Михеев И.А.

Проверил: Гай В.Е.

Нижний Новгород 2021

**Движение по шоссе (Highway driving)**

Целью теста является создать программу автономного управления модели автомобиля на трассе. Программа написана на языке программирования Python. модель робота – Lincoln MKZ. В данном случае метрика – это расстояние, пройденное за минуту без столкновений и съезда на обочину. Чтобы улучшить результат разработали алгоритм опережения впереди идущих автомобилей.

**Листинг**

"""Sample Webots controller for highway driving benchmark."""

from vehicle import Driver

from math import fabs

#Checking the presence of an object in front of the sensor

def Check(name,num):

if sensors[name].getValue() <= sensors[name].getMaxValue() - num:

return 0

else:

return 1

#Sensors determine whether there is free space for maneuver

def overtake():

rightflag = 0

leftflag = 0

leftflag = Check('front left 2',1) + Check('left',1)

if leftflag == 2:

driver.setSteeringAngle(-0.0165)

return 0.0195

else:

return 0

# name of the available distance sensors

sensorsNames = [

'front',

'front right 0',

'front right 1',

'front right 2',

'front left 0',

'front left 1',

'front left 2',

'rear',

'rear left',

'rear right',

'right',

'left']

sensors = {}

maxSpeed = 80

driver = Driver()

driver.setSteeringAngle(0.0) # go straight

# get and enable the distance sensors

for name in sensorsNames:

sensors[name] = driver.getDistanceSensor('distance sensor ' + name)

sensors[name].enable(10)

timer = 0

overTakeFlag = 1

rightLaneFlag = 0

direction = 0

rold = 0

#Main loop

while driver.step() != -1:

#Dynamic speed change depending on the distance in front of the vehicle

frontDistance = sensors['front'].getValue()

frontRange = sensors['front'].getMaxValue()

speed = maxSpeed \* frontDistance / frontRange

driver.setCruisingSpeed(speed)

if timer % 5 == 0:

rold = round(sensors['right'].getValue(),2)

#The part of the code responsible for starting the overtake maneuver

if Check('front',1) == 0 and overTakeFlag == 1:

if Check('left',1) == 1:

direction = overtake()

if direction != 0:

overTakeFlag = 0

timer = 0

#Defines movement in the right lane

if Check('front',2) == 1 and rightLaneFlag == 1:

if Check('front',1) == 0:

overTakeFlag = 1

rightLaneFlag = 0

print(round(sensors['right'].getValue(),2) - rightDiff)

deltaR = round(sensors['right'].getValue(),2) - rold

if deltaR < 0:

driver.setSteeringAngle(-1 \* fabs(deltaR))

else:

driver.setSteeringAngle(fabs(deltaR))

#Alignment to the middle lane after the start of the maneuver

if timer >= 250 and overTakeFlag == 0 and sensors['left'].getValue() <= 9:

driver.setSteeringAngle(direction)

deltaR = round(sensors['right'].getValue(),2) - rold

if sensors['left'].getValue() <= 8 or deltaR >= 0.2:

driver.setSteeringAngle(direction \* 2)

if timer == 350 and overTakeFlag != 2 and overTakeFlag != 1:

driver.setSteeringAngle(0)

deltaR = round(sensors['right'].getValue(),2) - rold

if deltaR < 0:

driver.setSteeringAngle(-1 \* fabs(deltaR))

else:

driver.setSteeringAngle(fabs(deltaR))

direction = 0

timer = 0

overTakeFlag = 2

#Shift to the right lane

if overTakeFlag == 2 and rightLaneFlag != 1:

FR1 = Check('front right 1',1)

R = Check('right',1)

flag = FR1 + R

if flag == 2 and sensors['left'].getValue() <= sensors['left'].getMaxValue():

driver.setCruisingSpeed(55)

driver.setSteeringAngle(0.0018)

if round(sensors['right'].getValue(),2) <= 7 and round(sensors['left'].getValue(),2) >= 10 and round(sensors['right'].getValue(),2) > 1:

driver.setSteeringAngle(-0.0019)

rightLaneFlag = 1

timer += 1

**Результаты**

