**Report of Homework III**

**Bauyrzhan Zhakanov**

There are two tasks about the implementation of mobile robot’s movement. The aim of the first task is to implement the visual simulation and the differential drive of mobile robot and its environment with obstacles, and it must reach to the goal point via Forward Kinematics. The aim of the second task is to implement simple Artificial Neural Network to move robot where two sensors: left sensor and right sensor avoid the obstacles including the edges of the environment. In order to run the code, you need to install:

import matplotlib.pyplot as plt

import numpy as np

from matplotlib.patches import Circle

from scipy.interpolate import CubicSpline

import math

**Task 1: Forward Kinematics**

As the first step, the environment with obstacles and the mobile robot were implemented. To implement that, matplotlib library has been used.

* To create the mobile robot, the points depending on the center point of robot, length is 12, width is 6, and tread is 5. Back wheels are 6. Length and width of wheel are 2.

Engineering drawing

Description automatically generatedChart, bubble chart

Description automatically generated

Figure 1. Mobile robot. Figure 2. The environment

An environment has dimensions 100x100. Obstacles has been drawn via *matplotlib.patches.Circle.* In code, they has been drawn in the function obstacles(). Radius of obstacles is 8. Their positions are in the points:

*x\_obstacles = [20, 50, 70, 90]*

*y\_obstacles = [90, 40, 50, 10]*

The differential drive of the robot implemented as following:

1. First of all, the values of left and right wheel speeds are declared as 5 m/s.

vl = 5

vr = 5

Thus, it is possible to find the values of angular and R value.

w = (vr - vl)/l

R = (l/2)\*(vr+vl)/(vr-vl)

1. ICC of the robot

ICCx = x - R\*np.sin(theta)

ICCy = y + R\*np.cos(theta)

Their values are sent to the *x\_robt, y\_robt*, and *theta\_robt* in order to move robot from its previous position to the new one.

An animation of the code:

if animation:

plt.cla()

obstacles()

plt.plot([x, x\_goal],[y, y\_goal], 'go')

robot(x, y, theta)

**Task 2: Simple ANN**

Left and right sensors have been declared. If they meet an obstacle in front of the mobile robot or it collides with edges of environment, it moves backwards.

if LEFT <= obst\_radius:

vr = -vr

if RIGHT < obst\_radius:

vl = -vl

Meeting with edges looks as following:

if leftSensor[0] >= 100 or leftSensor[1] >= 100 or rightSensor[0] <= 0 or rightSensor[1] <= 0:

vr = -vr

if leftSensor[0] >= 100 or leftSensor[1] >= 100 or rightSensor[0] < 0 or rightSensor[1] <= 0:

vl = -vl