Faculty of Power and Aeronautical Engineering
Department of Robotics

Mobile Robot Report

Task 2

Jafar Jafar,
Pratama Aji Nur Rochman

supervised by
dr. Wojciech Szynkiewicz, & Dawid Seredyński

Contents

1 Introduction

The second trajectory the youBot robot has to follow is a circle. The robot should keep the constant orientation while moving along a circle (as in the figure). The radius of the circle is passed to the control program through the variable-length argument list in the run_simulation function, e.g. running the function run_simulation(@solution2, false, 1.234) should generate the movement of the robot along a circle of radius 1.234 m.

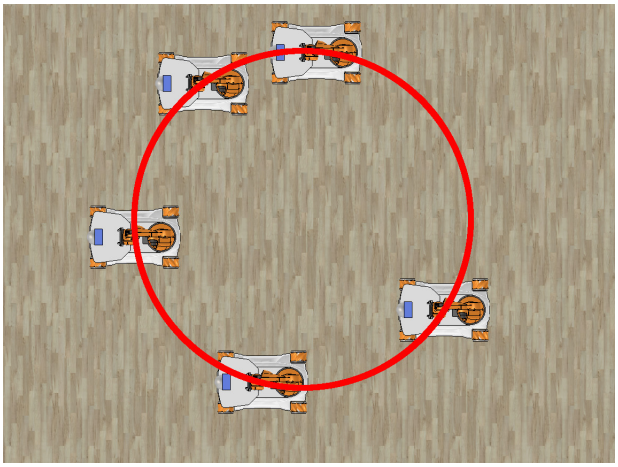

Figure 1: Orientation position

2 Task requirements

- The name of the control callback function should be solution2,
- The control program can be implemented as a two state finite state maching (FSM), i.e. aside of the initial state you it will be required to create another state responsible for computation of the required linear and angular velocities.
- It is suggested that in the initialize state you should store the coordinates of the center of the circle - they can be computed once, based on the current (initial) position of the robot and desired radius of the circle.
- Both position and orientation of the robot should be controlled by regulators (P regulators with limited output)
- The robot can move infinitelly.
- The radius of circle should be equal to the value provided as an argument to the run_simulation function.
- Use a clear environment with no obstacles, e.g. file /emor_trs/youbot/vrep_env/exercise01.ttt.

Please Note that the regulator of angular velocity and regulators of linear velocities are independent, so the following cases are acceptable:

- The robot reaches the desired orientation before reaching the desired point (as in the central picture)
- The robot reaches the desired point before reaching the desired orientation (as in the picture on the right)

Still, at the end, both desired orientation and position must be reached within some tolerance, e.g. 0.05 meters for position and 5 degrees for orientation.

3 Solution

3.1 Code

The code was named solution2.m and located in youbot directory. The callback function for this task was declared as:

```
1  function [forwBackVel, leftRightVel, rotVel, finish] = solution2(pts, contacts, position,
       orientation, varargin)
2  % The control loop callback function
3
4      % get the destination point
5      if length(varargin) ~= 1,
6           error('Wrong number of additional arguments: %d\n', length(varargin));
7      end
8      dest_r = varargin{1};
9
10     %decribe the wanted origin of the circle
11     circle_center = [0 , dest_r ];
12
13     % declare the persistent variable that keeps the state of the Finite
14     % State Machine (FSM)
15     persistent state;
16     if isempty(state),
17         % the initial state of the FSM is 'init'
18         state = 'init';
19     end
20
```

```
21      % initialize the robot control variables (returned by this function)
22      finish = false;
23      forwBackVel = 0;
24      leftRightVel = 0;
25      rotVel = 0;
26      P_v_r = 2;
27      Max_v_r = 1;
28      Max_v_t = 1;
29
30      % persistent variables used in FSM
31      persistent init_position;
32      persistent r;
33      % manage the states of FSM
34      if strcmp(state, 'init'),
35          state = 'move';
36          fprintf('changing FSM state to %s\n', state);
37          % save the initial position
38          init_position = position;
39          %%r = [position(1) + dest_r  position(2)];
40
41      elseif strcmp(state, 'move'),
42          center_position = circle_center - position(1:2);
43          u = sqrt(center_position(1)^2 + center_position(2)^2);
44          dir = center_position / u;
45          R_err = u - dest_r;
46          pos = R_err * P_v_r;
47          v_r = max(min(pos, Max_v_r), -Max_v_r);
48          v_t = Max_v_t;
49          vel_global = (v_r * dir) + (v_t * dir * [0 -1; 1 0]);
50
51                  %frame transformation
52                  theta = orientation(3);
53          R = [cos(theta), -sin(theta); sin(theta), cos(theta)];
54          vel_local = vel_global * R;
55          leftRightVel = vel_local(1);
```

```
56          forwBackVel = vel_local(2);
57      else
58          error('Unknown state %s.\n', state);
59      end
60  end
```

To run the simulation and the control program using with the callback control function solution1. The destination point is passed to the control program through the variable-length argument list in the run simulation function, e.g. run_simulation(@solution2, false, 1.234) where the arguments are:

- solution2 is the name of the control callback function,
- false - do not display the sensor data (simulation runs faster),
- 1.234 - generate the movement of the robot along a circle of radius 1.234 m

4 Result

The radius of the circle is passed to the control program through the variable-length argument list in the run_simulation function run_simulation(@solution1b, false, 1.234).

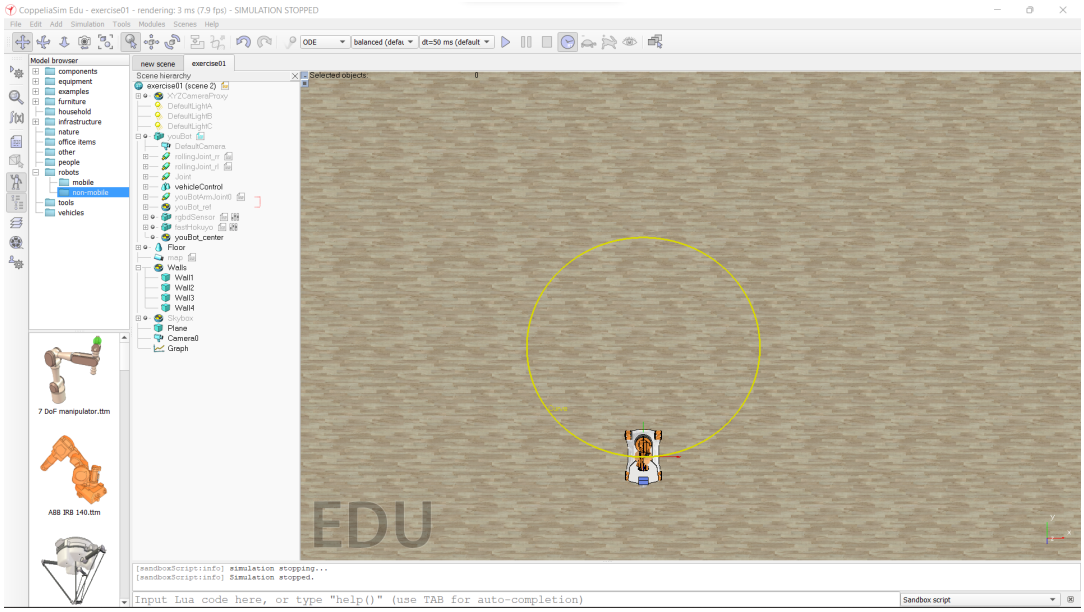

Figure 2: Starting point

As it is seen from the above pictures the robot follows circular trajectory infinitely without changing the orientation of the robot