Faculty of Power and Aeronautical Engineering
Department of Robotics

Mobile Robot Report

Task 4

Jafar Jafar,
Pratama Aji Nur Rochman

supervised by
dr. Wojciech Szynkiewicz, & Dawid Seredyński

Contents

1 Introduction

The goal of this task is to introduce reactive control. The reactive control of the youBot robot is realized by using data acquired from the LIDAR sensor to avoid obstacles in a specified task (moving along the walls). In this tutorial the 'reactive control' refers to the control of the speed of the robot without any trajectory planning.

You should write a program that enables youBot to move along walls, as presented in the figure below. The robot should use laser scanners in order to detect the wall, move along it (i.e. keep the distance) and avoid collisions.

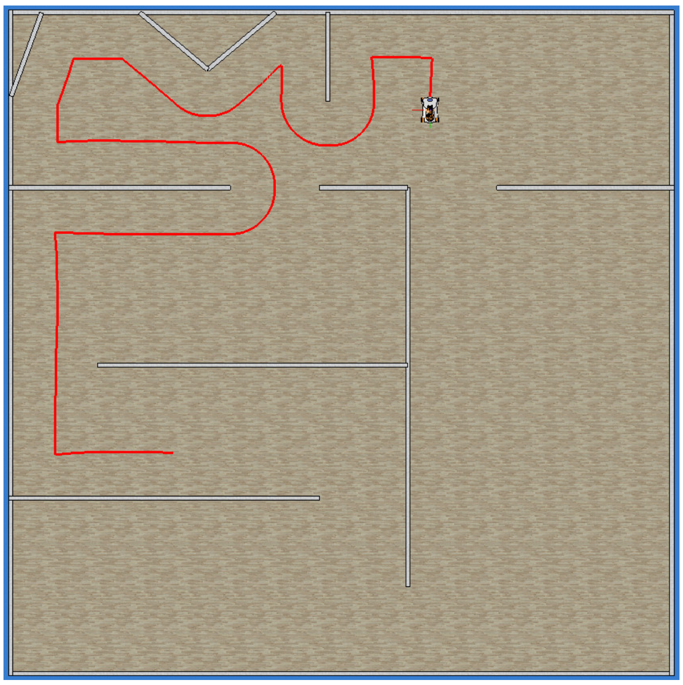


Figure 1: Orientation position

Similarly to the previous tasks, you should write a control function solution4 that implements proportional regulators that controls the robot movement.

And analogically, we also prepared a simulation environment in CoppeliaSim stored in file exercise02.ttt. You must open it in CoppeliaSim before running your matlab script.

Similarly, as in the previous tasks, the motion of the robot should be controlled by variables representing the relative velocities. Additionally, you should gather and analyse data from the lidar sensors as described on page Sensors.

Write a script solution4.m that implements a callback function solution4. Use the environment file /emor_trs/youbot/vrep_env/exercise02.ttt. The parameters are passed to the control program through the variable-length argument list in the run_simulation function, e.g. run_simulation(@solution4, false, 1) where the arguments are:

2 Task requirements

- Analyse the lidar readings in order to find the nearest obstacle (please consider the XY plane readings only).
- The robot should keep given distance to the wall (e.g. 1 m), hence you must compute the relative errors in directions perpendicular and parallel to the wall.
- The resulting robot speed should be computed on the basis of those errors.
- Additionally, the robot must always face the wall, hence you must also adequtelly control the angular robot velocity.
- Do not use the input variables position and orientation, as they are not relevant.

3 Solution

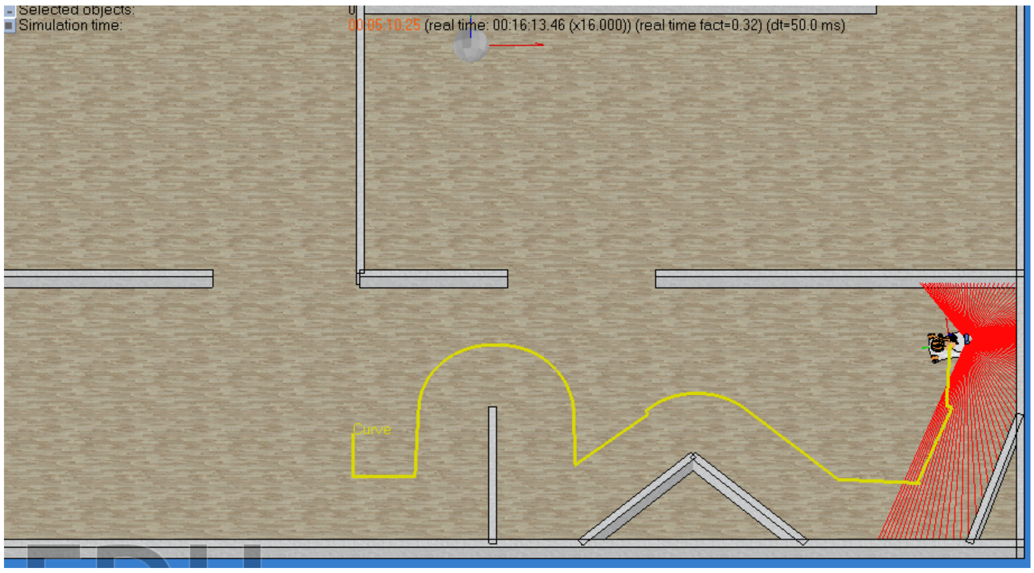
3.1 Code

The callback function for this task was declared as:

```
function [forwBackVel, leftRightVel, rotVel, finish] = solution4(pts, contacts,~,~,
    varargin)
    % get the parameters
    if length(varargin) ~= 1
         error('Wrong number of additional arguments: %d\n', length(varargin));
    end
    param_dist=varargin{1};



    % laser coordinates
    points = [pts(1,contacts)' pts(2,contacts)'];
    % distances calculation
    dists = (pts(1,contacts)'.^2 + pts(2,contacts)'.^2).^0.5;
    [min_value, min_index] = min(dists(:));
    % angle point
    angle = atan2(points(min_index,1),-points(min_index,2));
    %orientation co-ordinates
    theta=atan2(points(min_index,2),points(min_index,1));


    persistent state;
    if isempty(state),
        % the initial state of the FSM is 'init'
        state = 'init';
    end

    % initialize the robot control variables (returned by this function)
    finish = false;
    forwBackVel = 0;
    leftRightVel = 0;
    rotVel = 0;

    % manage the states of FSM
    if strcmp(state, 'init')
        state = 'move_forward';
        fprintf('changing FSM state to %s\n', state);

    elseif strcmp(state,'move_forward')
       rotVel=0;
       error_dist=param_dist-min_value;
       vel_local=5*error_dist;
       if vel_local>1
           vel_local=1;
       end
       if vel_local<-0.5
           vel_local= -0.5;
       end
       leftRightVel=0;
       forwBackVel=vel_local;

       if abs(error_dist)<0.05
        state='move_turn';
        fprintf('changing FSM state to %s\n', state);
       end
```

```

  elseif strcmp(state,'move_turn')
      rotVel=0;
      leftRightVel=3;
      forwBackVel=0;
      if abs(rad2deg(angle))>2
          state='move';
          fprintf('changing FSM state to %s\n', state);
      end

   elseif strcmp(state,'move')
       forwBackVel = 0;
       leftRightVel = 0;
       rotVel = 10 * angle;
       if rotVel > 0.5
           rotVel = 0.5;
       end
       if rotVel < -0.5
           rotVel = -0.5;
       end

       if abs(rad2deg(angle))<2
           state = 'move_forward';
           fprintf('changing FSM state to %s\n', state);
       end

   else
       error('Unknown state %s.\n', state);
   end

end
```

4 Result



```
Selected objects:    0
Simulation time:     00:05:10.25 (real time: 00:16:13.46 (x16.000)) (real time fact=0.32) (dt=50.0 ms)
```

Figure 2: Orientation of robot