Local Navigation Pseudocode

Basics of Mobile Robotics

Bubble Rebound Algorithm as proposed in this paper: I. Susnea, A. Filipescu, G. Vasiliu, G. Coman and A. Radaschin, "The bubble rebound obstacle avoidance algorithm for mobile robots," IEEE ICCA 2010, 2010, pp. 540-545, doi: 10.1109/ICCA.2010.5524302.

The pseudocode was inspired by the implementation of this algorithm on a simulator by a lua plugin that can be found here: https://github.com/BeBeBerr/bubble-rebound

The pseudocode written has some variations with what has been implemented due to numerous unknown parameters around the robot class implementation at the time.

Bubble Rebound pseudocode:

Function : Check For Obstacles	
Input : Sensor readings	**Output :** Binary value for obstacle

```
Function Check For Obstacles (sensors):
    threshold_array = VALUE
    for i from 1 to 5 with a step of 1:
        if sensors[horizontal][i] > threshold_array
            return true
        end if
    end for
    return false
end function
```

Function : Get angle	
Input : Sensor readings	**Output :** angle

```
Function Get angle(sensors):
    alpha = 80 / 5
    alpha_array = alpha * [-2 -1 0 1 2]
    tmp1, tmp2 = 0
    for i from 1 to 5
        tmp1=tmp1+alpha_array[i]*sensor[horizontal][i]
        tmp2=tmp2+sensor[horizontal][i]
    end for
    return tmp1/tmp2
end function
```

Function : Set speed

Input : linear, angular	**Output :** motor speed

```
Function Set speed(linear, angular):
    rspeed = linear + angular/2
    lspeed = linear - angular/2
    motor_speed=[rspeed lspeed]
    set motor_speed
end function
```

Function : Turn	
Input : robot position, turn_speed	**Output :**

```
Function turn(robot):
    while Check For Obstacles(robot sensors):
        turn_condition = true
        while turn_condition
            last_angle_pos = robot position[angle]
            sum_angle = 0
            current_angle_pos = robot position[angle]
            delta = absolute value(current_angle_pos-last_angle_pos)
            last_angle_pos = current_angle_pos
            sum_angle = sum_angle + delta
            if angle >= 0
                Set Speed(linear, -turn_speed)
            else
                Set Speed(linear, turn_speed)
            end if
            if sum_angle > angle
               Set Speed(linear,0)
            end if
end function
```

Function : main local navigation	
Input : sensor readings	**Output :** motor speed

```
Function main local navigation(sensor readings):
    while true
        if Check For Obstacles(sensor readings)
            Turn(Get Angle(sensor readings))
            sleep function
end function
```