ENG5009 Advanced Control 5 – Model descriptions

RunModel.m

This is the main working file for the simulation. The file is split up into sections. The only sections that need to be altered by you are indicated by `*--------*` in the file.

DynamicalModel.m

This file is a mathematical representation of a four wheeled robot. The robot is run by applying voltages to each of the motors.

The inputs to the model are:
`voltages` – voltages supplied to each motor. An array with four elements, `[voltageLeft voltageLeft voltageRight voltageRight]`. The first two are voltages applied to the left-side motor and the last two are voltages applied to the right-side motors. The voltage range is -7.4 to 7.4V and **YOU** should limit this within the simulation file.
`state` – is the current state of the mobile robot. This is calculated for you in the main working file. The variables of importance are:

`state(13)`:	forward velocity,	v, m/s
`state(18)`:	rotational velocity,	r, rad/s
`state(19)`:	current x-position,	x, m
`state(20)`:	current y-position,	y, m
`state(24)`:	heading angle,	psi, rad

`stepSize_time` – is the main simulation step size and should not be altered.

In the main file (`RunModel.m`), the outputs of the dynamical model are captured in `stateDerivative` and `state`.

Sensor.m

This function calculates the output of the sensors based on the obstacle matrix (which contains all walls). The function is used as follows:

```
[ sensorOutput ] = Sensor(robot_x, robot_y, headingAngle, obstacleMatrix)
```

in which:
`robot_x` – current x-position of the center of the robot
`robot_y` – current y-position of the center of the robot
`headingAngle` – current heading angle of the robot
`obstacleMatrix` – matrix containing all pre-defined obstacles (walls)
`sensorOutput` – returns the output of the sensor for the left and right sensor:
`[leftSensor, rightSensor]`, which are the distance to the nearest obstacle for that sensor.
Important note: when there is no obstacle, the sensor returns 1.

WallGeneration.m

This function updates the obstacle matrix by adding a wall. The function is used as follows:

```
[ wall, obstacleMatrix_new ] = WallGeneration(start_y, end_y, start_x, end_x,
direction, obstacleMatrix)
```

in which:
`start_y` – y-coordinate of start of wall
`end_y` – y-coordinate of end of wall
`start_x` – x-coordinate of start of wall
`end_x` – x-coordinate of end of wall
`direction` – direction of wall, either `'h'` or `'v'` (horizontal or vertical, respectively)
`obstacleMatrix` – obstacle matrix
`wall` – coordinates of wall (you can use this to plot them in figures)
`obstacleMatrix_new` – updated obstacle matrix