Appendix A: Matlab functions in ARSim

The following is an alphabetical list of all the Matlab functions associated with `ARSim`. For each function, the library to which the function belongs is given, along with the interface of the function and a brief description. For further information, see the actual source code for the function in question.

AddMotionResults

Library: ResultFunctions
Interface:
`motionresults = AddMotionResults(oldmotionresults, time, robot)`
Description: This function updates the motion results by adding the current position, velocity, heading, and sensor readings of the robot.

BrainStep

Library: –
Interface: `b = BrainStep(robot, time);`
Description: The `BrainStep` implements the decision-making system (i.e. the brain) of the robot. The detailed form of this function will vary from experiment to experiment.

CalibrateOdometer

Library: RobotFunctions
Interface: `o = CalibrateOdometer(Robot)`
Description: In simulations in which an odometer is used, a call to `CalibrateOdometer` is made just before the start of the simulation, in order to set the correct position and heading of the robot.
See also: `CreateOdometer`

CheckForCollisions

Library: RobotFunctions
Interface: `coll = CheckForCollisions(Arena, Robot);`
Description: This function carries out a collision check, by running through all arena objects (polygons) line by line, and checking for intersections between the current line and the spherical body of the robot.

CreateArena

Library: ArenaFunctions
Interface: `arena = CreateArena(name,size,objectarray)`
Description: This function generates an arena, given an array of arena objects.
See also: `CreateArenaObject`

CreateArenaObject

Library: ArenaFunctions
Interface: `arenaobject = CreateArenaObject(name,vertexarray)`
Description: This function generates an arena object, given an array of coordinates for vertices.

CreateBrain

Library: –
Interface: `b = CreateBrain;`
Description: This function generates the brain of a robot. Its exact form will vary from experiment to experiment.

CreateCompass

Library: RobotFunctions
Interface: `c = CreateCompass(name,acc);`
Description: This function generates a compass, with accuracy `acc`.

CreateIRSensor

Library: RobotFunctions
Interface:
```
s = CreateIRSensor(name,relativeangle,size,nr,
                   openingangle,range,c1,c2);
```
Description: `CreateIRSensor` creates an IR sensor that uses the ray tracing procedure described above to obtain its readings.

CreateMotor

Library: RobotFunctions
Interface: `m = CreateMotor(name);`
Description: `CreateMotor` generates a DC motor, using settings suitable for a robot with a mass of a few kg.

CreateOdometer

Library: RobotFunctions
Interface: `o = CreateOdometer(name, eps, sigma);`
Description: This function generates an odometer, which, in turn, provides estimates for the position and heading of the robot. `eps` and `sigma` determine the odometric drift (see Eqs. (1.53) and (1.54)).

CreateRobot

Library: RobotFunctions
Interface:

```
robot = CreateRobot(name,M,I,r,rw,sensorarray,
                    motorarray,brain,odometer)
```

Description: `CreateRobot` sets up a robot, and computes the dynamical parameters typical of a robot with a mass of a few kg.

GetCompassReading

Library: RobotFunctions
Interface: `c = GetCompassReading(Robot, dt);`
Description: This function updates the compass readings of a robot.

GetDistanceToLineAlongRay

Library: RobotFunctions
Interface: `l = GetDistanceToLineAlongRay(beta,p1,p2,x1,y1);`
Description: This function, which is used by the IR sensors, computes the distance from a given point (x_1, y_1) to a line segment.
See also: `GetIRSensorReading,GetDistanceToNearestObject.`

GetDistanceToNearestObject

Library: RobotFunctions
Interface: `d = GetDistanceToNearestObject(beta, x, y, Arena);`
Description: This function, which is used by the IR sensors, determines the distance between an IR sensor and the nearest object along a given ray.
See also: `GetIRSensorReading`.

GetIRSensorReading

Library: RobotFunctions
Interface: `s = GetIRSensorReading(Sensor,Arena);`
Description: `GetIRSensorReading` determines the reading of an IR sensor.

GetMinMaxAngle

Library: RobotFunctions
Interface: `[amin,amax] = GetMinMaxAngle(v1,v2);`
Description: This function determines the direction angles of the vectors connecting the origin of the coordinate system to the tips of a line segment.
See also: `GetDistanceToNearestObject`.

GetMotorSignalsFromBrain

Library: RobotFunctions
Interface: `s = GetMotorSignalsFromBrain(brain);`
Description: This function extracts the motor signals (one for each motor) from the brain of the robot.
See also: `MoveRobot`.

GetOdometerReading

Library: RobotFunctions
Interface: `o = GetOdometerReading(Robot, dt);`
Description: This function updates the odometer readings of a robot.

GetSensorReadings

Library: RobotFunctions
Interface: `s = GetSensorReadings(Robot, Arena)`
Description: This function obtains the reading of all (IR) sensors of the robot.
See also: `GetIRSensorReading.`

GetTorque

Library: RobotFunctions
Interface: `m = GetTorque(motor, voltage);`
Description: This function determines the torque delivered by a DC motor, given a value of the applied voltage.

InitMotionResults

Library: ResultFunctions
Interface: `motionresults = InitResults(robot)`
Description: This function initializes a Matlab structure used for storing the results of the simulation, i.e. the position, velocity, heading, and sensor readings of the robot.

InitPlot

Library: PlotFunctions
Interface: `hp = InitPlot(Robot, Arena)`
Description: This function generates the plot of the robot and the arena.
See also: `CreateArena`, `CreateRobot`.

MoveRobot

Library: RobotFunctions
Interface: `r = MoveRobot(Robot,dt);`
Description: `MoveRobot` moves the robot according to the equations of motion for a differentially steered two-wheeled robot.

ScaleMotorSignals

Library: RobotFunctions
Interface: `v = ScaleMotorSignals(r,s);`
Description: This function scales the motor signals to the appropriate range, as set by the voltage requirements of the robot's DC motors.

SetPosition

Library: RobotFunctions
Interface: `r = SetPosition(Robot,pos,heading,vel,phidot);`
Description: This function places the robot at a given location, and also sets is direction of motion, velocity, and angular velocity.

ShowRobot

Library: PlotFunctions
Interface: `ShowRobot(plot,Robot)`
Description: `ShowRobot` updates the plot of the robot using Matlab's handle graphics: Each part of the plot of the robot can be accessed and its position can be be updated. `ShowRobot` also supports the plotting of an odometric ghost,

i.e. a plot showing the robot at the location determined by its odometer.
See also: `MoveRobot`.

UpdateMotorAxisAngularSpeed

Library: RobotFunctions
Interface: `r = UpdateMotorAxisAngularSpeed(robot)`
Description: This function determines the angular speed of each motor axis, using the wheel speed and wheel radius.

UpdateSensorPositions

Library: RobotFunctions
Interface: `s = UpdateSensorPositions(Robot);`
Description: This function updates the positions (and directions) of the sensors as the robot is moved.