Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC-PapersOnLine 49-6 (2016) 069–074

# MARS: a Matlab simulator for mobile robotics experiments

**Marco Casini, Andrea Garulli**

*Dipartimento di Ingegneria dell'Informazione e Scienze Matematiche*
*Università di Siena*
*Via Roma, 56, 53100 Siena, Italy*
*E-mail: {casini,garulli}@ing.unisi.it*

**Abstract:** This paper describes a Matlab simulator for mobile robotics experiments called *MARS*. This tool allows the simulation of team of robots in structured and unstructured environments. Several kinds of experiments can be performed, ranging from single to multi-robots, from cooperative to competitive tasks, using both centralized and distributed controllers. Robots may be equipped with virtual sensors to detect obstacles in the environment. This project is mainly intended for educational aims, and thanks to the use of the Matlab language, it allows students to easily design control laws for teams of robots and to simulate their behavior through a suitable animation.



Keywords: mobile robotics, simulation, Matlab

## 1. INTRODUCTION

Mobile robotics is worldwide recognized as one of the main topic regarding robotics and control systems. Lots of research papers can be found in the literature addressing different problems, like localization, mapping, obstacle avoidance, formation control, rendezvous, etc. Due to the nature of this field, it is standard to validate theoretical results by suitable experiments involving real robots. Nevertheless, since experimental setups are usually costly and time demanding, simulation tests are often performed before the real verification. This is the motivation behind the development of a user-friendly mobile robot simulator. At present, several simulators are available, developed both by commercial companies and educational institutions. Examples of commercial products are Webots (Cyberbotics Ltd. [2016]) and V-rep (Freese et al. [2010]), while educational simulators can be found in Carpin et al. [2007], Vaughan [2008], Guzmán et al. [2008], just to cite a few.

In this paper, a simulator for mobile robotics called *MARS* is presented. Such a simulator has been implemented in Matlab and allows the simulation of a wide range of experiments involving one or more vehicles, centralized and distributed control, cooperative and competitive tasks. This project was launched with the aim of simulating the behaviour of mobile robots built with Lego Mindstorms bricks (The LEGO Group [2016]) embedded in the remote laboratory called Automatic Control Telelab (ACT), see Casini et al. [2009, 2011, 2014]. Through such vehicles students may performs experiments of various nature, involving obstacle avoidance (Casini et al. [2012]), pursuer evader game (Casini et al. [2013]), etc. From its preliminary scope, this tool has been extended to deal with generic robots working in user-defined environments and equipped with various sensors. This facility is mainly oriented to education and it is not intended to compete with above mentioned products in terms of animation reality. The main aim of this project is to provide a tool which can be easily used by students for the design and test of control laws to be embedded into real robots, by using a well known programming language as Matlab. A key feature is the ability of using and designing *addons*, i.e., special features related to a given topic. Students may take advantages by this feature by embedding in their projects predefined addons or addons designed by other students. In addition, robots may be equipped with sensors simulating real devices. *MARS* has been successfully adopted in undergraduate engineering courses at the University of Siena, and despite its simplicity, it may be also used as a research tool, allowing researchers to exploit the power of Matlab to design and test complex controllers before the actual implementation on real vehicles.

The paper is organized as follows. In Section 2, an overview of the simulator and its capabilities is reported. The *MARS* software architecture is described in Section 3, while in Section 4 an overview of the main features and of possible experiments is provided. A simulation session regarding Pursuer-Evader Games is described in Section 5, while conclusions and future developments are drawn in Section 6.

## 2. SIMULATOR OVERVIEW

The *Multi-Agent Robot Simulator (MARS)*[1] is a software simulator whose aim is to provide an easy-but-powerful platform for simulating team of mobile robots in structured/unstructured environments. It is able to simulate team of robots moving in a two dimensional workspace.

[1] Website: http://mars.diism.unisi.it


10.1016/j.ifacol.2016.07.155

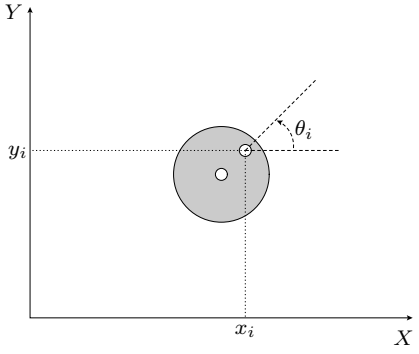


Fig. 1. Pose for unicycle-like robots.

Both holonomic and non-holonomic vehicles can be simulated. Sampling time is set by the user and it is assumed that robots can change their status at each time step.

Let $x_i(t)$ and $y_i(t)$ be the position of the $i$-th robot at time $t$, and $v_x(t)$, $v_y(t)$ be its velocities along the two coordinate axes. It is assumed that holonomic robots are driven by their velocities along the two coordinate axes, so their kinematics can be easily expressed as

$$\dot{x}_i(t) = v_x(t),$$
$$\dot{y}_i(t) = v_y(t).$$

Instead, non-holonomic vehicles have been implemented as unicycles, whose poses evolve as

$$\dot{x}_i(t) = v_i(t)\cos(\theta_i(t)),$$
$$\dot{y}_i(t) = v_i(t)\sin(\theta_i(t)),$$
$$\dot{\theta}_i(t) = \omega_i(t),$$

where $v_i(t)$ and $\omega_i(t)$ are the linear and angular speed, respectively, and $\theta_i(t)$ denotes the robot orientation (see Fig. 1).

Robots can be equipped with the same controller or with different controllers as well, like e.g., for experiment involving one leader and many followers. Several types of control strategies can be implemented, as:

**Centralized control with full information.** It is assumed that the complete information about the environment and all robot poses is known by a supervisor controller. At each time step, such a controller is in charge to compute desired velocities for all agents in the network.

**Centralized control with partial information.** The supervisor controller knows only partial information about robot status, like e.g., position but not orientation.

**Distributed control with full information.** Each vehicle computes its own speed based on its own pose and the poses of other robots.

**Distributed control with partial information.** Besides its own pose, each agent knows the poses of some of the other vehicles, for instance of its neighbors. They may also know partial information about other agents, like e.g., their position but not their orientation.

**Distributed control sensor information** Each robot is equipped with sensors which provide the only available information.

Other than robot networks, this tool can be used to simulate algorithms involving one robot. Assuming the robot is equipped with onboard sensors, typical examples regards path planning, localization, mapping, etc.

## 3. SIMULATOR ARCHITECTURE

*MARS* has been developed in Matlab and consists in a set of functions which can be divided into 5 main groups depending on their functionalities.

**Main file.** The file called *mars.m* represents the main function of the simulator and it is the function to be launched to run a simulation.

**Resource files.** These are functions which can be used by the main file and/or by users to provide specific functionalities (e.g., check if collisions occur).

**Sensor files.** These functions allow the simulation of virtual sensors mounted on the robots, like proximity sensors, laser range finders, etc.

**Addons files.** *MARS* allows one to use predefined and user-defined functions to simulate special features of interests, like e.g., presence of obstacles, Voronoi diagrams, etc. Such functionalities are implemented through addons provided by the simulator or designed by users.

**Demo files.** These files are useful to show demonstrations of the given functionalities. They can be used as a starting point to develop new experiments.

In addition, documentation files are also provided.

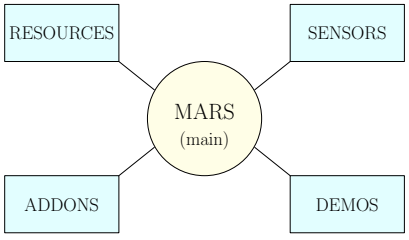


Fig. 2. *MARS* file classification.

Hereafter, the main file (*mars.m*) which is the core of the project is described in details. It consists in a Matlab function defined as follows.

*[out_data]=mars(exp_file, stop_time, speed, options)*

- *exp_file* refers to the function containing the experiment to be simulated, which has to be defined by the user. Demo files could be used as well. This is a mandatory input parameter.
- *stop_time* specifies the time duration of the experiment in seconds. It is an optional parameter.
- *speed* denotes the playback speed of the simulation. *speed=1* means that the simulation runs at actual time, i.e., the duration of the simulation is the same as that of the real experiment. Greater values can be used to increase the simulation speed, while lower values to slow it. Special values are *speed=0* which means "manual mode", i.e. simulation proceeds at each key pressed, and *speed='max'* which runs the simulation at the maximum speed allowed by the available computational power. If this parameter is not given, the value 1 is assumed by default.
- *options* is an optional structure defining options to be used in the simulation. All the fields of the experiment status structure (*Exp_status*) can be set.