A Distributed Simulator for Intelligent Autonomous Robots

Lorenzo Sommaruga, Ignacio Merino, Vicente Matellán, José Manuel Molina

Laboratorio de Agentes Inteligentes (LAI)
Departamento de Informática
Avenida Butarque 15, Leganés-Madrid, SPAIN Phone: +34 1 624 94 31
Fax +34 1 624 94 30 E-mail: {vmate.molina}@ia.uc3m.es
Universidad Carlos III de Madrid, España

1 Introduction

This paper describes the SimDAI system developed at the Intelligent Agent Laboratory of University Carlos III de Madrid [Merino95], as a continuation of the ERA project [Sommaruga95a]. SimDAI is a working prototype of a mobile robots simulator fully implemented as a distributed system. The main objective of the project is the development of a multi-robot simulation environment for experimenting with robot navigation and control algorithms, cooperation and interaction among robots in a group. In particular, design and implementation issues for the simulation of autonomous mobile robots have been emphasized. Each mobile robot is completely independent, can navigate and interacts with other robots in a 2-D simulated world of obstacles, which is separately monitored.

2 Previous Works and Background

A number of basic architectural principles, which derived from the Distributed Artificial Intelligence (DAI) field and, in particular, from multi-agent systems and cooperative architecture for autonomous agents, have been employed in this work [Sommaruga95b]. In addition, the design of the SimDAI architecture has required a detailed study of various features of previous simulators. This phase aimed to enrich the new simulator by exploiting all the positive characteristics of previous simulators, and trying to overcome their limitations. The characteristics and limitations of the major influential works, which were a MS-DOS simulator prototype, the ERA project, and the Bug World system, are mentioned below.

A MS-DOS prototype [Blazquez96] was initially developed at LAI for the simulation of a mobile robot. It allows to define open maps of the world, without confined border, where robots and mobile objects can move around following predefined trajectories. On the other hand, in this simulator, due to its implementing platform, multi process is not allowed, a data structure of the world map does not explicitly exist, and the information necessary for simulating - sensing the environment - has to be taken directly from the display.

A second prototype developed at LAI was the ERA simulator [Sommaruga95a]. This simulator was a distributed workbench for autonomous mobile robots. It provided guidances about multi processing of robots, and about maintaining the simulation independent from its visualization. Some limitations of this system consist of executing all the processes on a single computer, and not providing a complete interface for the end user interaction, especially concerning the data input.

Other interesting ideas have been adapted from the Bug World simulator [Almassy93], such as task sharing through interconnected processes over a computer network, although communication between the robots is not provided in this simulator. In general, this software is quite portable, except for the process which shows the simulation evolution that depends on the host architecture.

The SimDAI interface design and implementation was mainly guided by an analysis of the results of the ERA experience. In particular, the UNIX operating system, X Window, and fault tolerance issues [Jalote94] [Brown94] have been considered because of the need for parallel processes, in order to provide communication among the robots, and a graphic and secure environment.

Finally, in order to guarantee the independence of a robot from the others, from the external simulated world and its visualization, the robot structure has been based on the cooperative autonomous agent model of CooperA-II [Sommaruga95b].
3 The SimDAI Architecture

The SimDAI software allows multiple and various mobile robot systems to be represented, simulated and experimented with in a two dimension environment. The simulation world consists of a rectangular map of user defined dimensions where particular objects are located. Two types of objects have been considered: obstacles of different shapes, such as circles, segments or polygons; and mobile robots.

The SimDAI system mainly provides support for three types of activities: simulating the robot, simulating the world environment, and monitoring the world. Each type of activity is carried out by a distinct process (see Figure 1). These processes are independent and communicate by means of sockets using a particular message protocol.

These components are integrated together in order to form a real distributed system. Each process or activity can be run on a different host and is interconnected with the others by means of a computer network.

Communication facilities have been also provided to the robots. Thus, basic actions of a robot, such as controlling right and left wheel speeds, sensing distances and light intensity, have been completed by a message communication mechanism among different individuals.

The communication facility implemented is based on a centralized, one to one model. When a robot wants to send a message to another robots, this message is sent to the process that simulates the world who is in charge of sending it to the destination. This communication model has been chosen because it allows us to modify or improve it easily, maintaining the modifications implemented transparent to the robot process and because the other kinds of communication can be implemented using this method.

The communication facilities has been defined as a communication protocol among processes. A robot process and a world process consider 32 different messages such as to send a message from a robot to another robot (REM), or to Read the measures returned by a Sonar Sensor (RSS). Another group of messages has been defined to carry out the communication between a world and a monitor, consisting of 21 different messages such as when the world process sends the objects definition to a monitor process (MEF), or when a monitor suspends its activity (MHP).

In the SimDAI architecture, the world process is the core of the system activity. Due to the independence of all the processes, robot processes and monitor ones can be added to the world simulation as many as required and in any moment of its running. In this way, a flexible and dynamic simulating environment is provided. Moreover, more than one monitor process may be activated at the same time, for example, in order

---

**Fig. 1.** The SimDAI architecture
to display different regions of the world map and of their local happenings. The monitor processes are not indispensable for running a simulation, they just offer information about the state of affairs of the world.

In addition, the simulator permits the change of some parameters. These are set at the beginning of the simulation experiment by means of initialization files, such as the objects in the world, the attributes of the robots, etc. They may be also changed using parameters at the call to each process. For instance, the simulation steps to be run before the monitor displays situation changes in the world map can be fixed using the parameter \texttt{-p <steps>}. In the calls to robot or monitor processes the host name and the port number of the server where the world process is running can be set using the parameters \texttt{-s <server> -m <port>} in order to establish the connection.

Some settings can be also modified during the simulation by means of the interface menus of each process. For instance, in the monitor the scale rate used in the graphic representation or the origin of the window to display a different region of the world can be modified. From the window of a robot the display of the sensors exploration can be activated. It is also possible to display or not the track of robot in the world.

A fault tolerance mechanism has been also provided to the system for the connection of distinct processes. When a fault in a process is detected through a lost connection, the lost process stops itself and the remaining processes try to reconfigure the simulation. If the process that faults is the world simulation process, the whole system is stopped and all the processes are killed in a controlled way.

\subsection{The World Process}

The \textit{world} process is in charge of representing the real world aspects which are required for the simulation. It manages a world map in which it represents obstacles and robots. In addition, it deals with the system simulation of all the activities concerning the real world by carrying out the role of the real world, i.e., for example, providing data to the robots' sensors, rendering the movements, etc. There may be only one world process in the whole system, according to the idea that all the objects (robots and obstacles) exist in the same world. The world process also carries out some other tasks as the communication among robots, the connection and disconnection of robots and monitor.

One of the main tasks of the world process is to keep the representation of the world. This representation is based on a two dimension array. This array holds a position for each point of the world. Each one of these points can have 256 different values of color, where the first 32 are invisible for the robot. The invisible colors are used to represent the sensor range, the track of the robots or any other debugging information and are considered as empty regions. The rest of the colors can be used to define the objects.

\subsection{Robot Processes}

In SimDAI, the characteristics of the Turtle robot model [McKerrow91] and the physical restrictions of the Khepera prototype [Mondada93] have been considered. The turtle model has been used to define the model of the simulated robot, while the real restrictions of the Khepera has been taken into account because this mini robot is quite used in lab experimenting.

The \textit{robot} process represents a robot object within the simulation. This process carries out the initialization of the robot and the basically governs its control policy. For example, it may determine the speed variation of the two wheels according to its data sensing. Different control policies can be easily adopted by changing the control algorithm in this process.

Many robots may coexist in the same world, forming an interacting group of individuals with different characteristics.

In SimDAI, various robots can be simulated diversifying their dimensions, color, wheel distance, sensor number, type, range, and orientation. These robots are completely independent processes using their own control architecture. These processes are designed and built apart from the world process and only interact with it when the user establishes the connection between them. This means that the processes can be started in any order and that they can also be connected and disconnected dynamically. The same policy has been used in the design of the monitor processes: they are independent, dynamically connect and disconnected processes.

Currently, the simulation of sensors of object proximity and robot location allow parameters like linear distance, region occupancy, or position to be estimated. These correspond to real robot sensors such as lasers, infra red, or GPSs. Non ideal characteristics of real sensors could be modelled by defining new types of sensors and introducing an external noise to the sensors exploration output.
The definition of the robot is made in a file whose name is asked when this process is called. The configuration of a robot may be defined according to the robot definition format as follows:

\begin{verbatim}
name: name
initial position: x y angle
aspect: radio wheel-dist color track direction
laser-sensor: distance angle angle range
sonar-sensor: distance angle angle range ang_sector interval type
GPS sensor
\end{verbatim}

In particular, to define a robot like the one in Figure 2, the robot definition will be, where the distances are in pixels, angles in degrees and colors as integers in the accepted range:

\begin{verbatim}
nombre R06
posicion 150 80 180
aspecto 15 30 38 6 255
sensorlaser 15 135 135 50
sensorsonar 15 180 180 50 45 15 1
sensorlaser 15 225 225 50
sensorGPS
\end{verbatim}

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{robot_example.png}
\caption{Robot example in SimDAI}
\end{figure}

The name of the simulated robot is assigned in the first line (in the example R06). In the second line its initial position coordinates (x, y), and the orientation angle is fixed. Then, the physical characteristics are set in the aspect definition. The parameters are the radio of the robot platform, the distance between the wheels, the color of the robot, the color of the track and the color of the heading indicator. As already mentioned, the available colors are divided into two groups, visible and invisible, in order to let the sensors display and track of the robot be detected or not by robot sensors.

A variable number of sensors can be defined for each robot. In this first version of the simulator, there have been built three kinds of sensors. The first one simulates a distance laser based sensor. This sensor is supposed to return the distance to the nearest obstacle with high precision, but in a narrow and fixed direction. In order to define a sensor, its position within the robot is provided in the form of the distance and the angle from the center of the robot. Then, the direction in which the sensor scans has to be fixed. The last parameter is the maximum reach of the sensor.

The sonar sensor is similar to the laser, but in this case the distance returned indicates that an obstacle has been detected into its sector of vision range. A simulated sensor of this type is obtained by a multiple exploration of a laser sensor. It is defined with the same parameters as the laser, plus the vision range of the sensor, the density of the scans in the vision sector and the type of the measurements returned (minimum, average or maximum of the laser explorations).

The third type of sensor simulates a GPS (Global Positioning System) sensor. A real GPS device returns its position in longitude and latitude. Our simulated sensor returns its position in coordinates of the simulated world. The definition of this sensor has no parameters because its supposed to be in the center of the robot.
Fig. 3: A SimDAI simulation

3.3 Monitor Processes

The monitor process carries out the interfacing task, by presenting the end user a window which graphically shows the evolution of the world situation. More than one monitor can be run in a simulation, in order to monitor different parts of the world. For instance, in Figure 3 a simulation with 2 robots is presented. In the lower monitor the whole world is presented while the other monitor process is showing a specific part of the world.

The window of the monitor process offers a number of useful options for a simulation task. The first one is the refreshing of the window. This option is very useful when there is a group of robots with the track and sensor displaying option on. This makes that in a short time the world gets completely covered by those information.

There are also other interesting configurable options. For example, the scale rate used to represent the world map can be fixed in a direct way or can be increased or decreased in a pre-determined size. Other options are intended to fix the origin point for the representation of the world map. That origin, joined to the representation rate, define the part of the world that is visualized in the window monitor.

Figure 4 presents an example of the output of a monitor process, which displays a world with some obstacles and robots, and a pop-up menu for selecting one of the previously mentioned commands. In this
Fig. 4: The SimDAI monitor

figure it can also be noticed the way the sensors and the track of a robot are displayed. In the figure the sensors of the robot named R04 and the track of the robot R02 are shown. It can also be seen that in the bottom of the window some of the parameters of the representation are shown. In particular, the scale (Escala), the origin coordinates (Origen x, y) and the status of the process (Parado, i.e. stopped) are shown.

4 Implementation

The implementation of this work has been carried out attending requirements of portability, autonomy and distribution. In order to facilitate the portability, the different processes has been implemented in C++ using the gnu C++ compiler. These processes have been compiled and tested under different UNIX operating systems: SunOS, HP/UX, Linux, and NetBSD.

A distinct process has been assigned to each robot in order to provide autonomy. An individual process is also used to implement each monitor to provide flexibility, distribution and remote monitoring of the simulation. On the other hand, the world simulation process will be unique and it has been implemented as a single process.

Concerning communication, sockets which uses the TCP/IP protocol supported by UNIX, have been used. The graphical user interface has been developed using the X Window system. Once again, in order to provide portability to the simulator, two interfaces have been implemented: a basic one using Xlib, for its portability; and another one using X Toolkit Intrinsics, for advanced facilities.
5 Conclusions and Further Works

We have designed and implemented the SimDAI packages as a complete software for the distributed simulation of mobile robots which resulted easy to use. It allows the behaviour of a group of mobile robots to be tested or foreseen in particular situations, without the need for a physical implementation. A number of experiments can be easily defined and run in order to evaluate different world, robots, sensors, and control algorithms.

We have used this simulator to define various experiments and to test their viability before implementing them in the real Khepera mini-robots, such as the experiments on fuzzy communication between robots [Matellán96].

The SimDAI graphic interface provides various information to the user: checking robots’ parameter values, such as speeds, sensor data, etc.; having a global overview of the current world situation and of a part of it at the same time; following run-time changes during the simulation development; collecting final results for a posteriori analysis and comparisons, etc.

Future work is already in progress in order to connect a real robot, such as a Khepera, to the simulator. Moreover, an extension of the communication protocol for the cooperation and coordination in a group of robots is under development. The implementation in different platforms like Windows is under study.

References


[Sommaruga95b] Sommaruga, L. and Catanzari, N. *From Practice to Theory in Designing Autonomous Agents*, First Australian Workshop on Distributed Artificial Intelligence, 8th Australian Joint Conference on Artificial Intelligence (AI'95), Canberra (Australia), Nov. 1995. Also in Lectures Notes on Artificial Intelligence, LNAI-1087, pp. 130-143, Springer-Verlag, 1996.

This article was processed using the TeX macro package with SIRS96 style