Simulation of Mobile Robots in Virtual Environments

Jesús Savage [1], Emmanuel Hernández [2], Gabriel Vázquez [3],
Humberto Espinosa [4], Edna Márquez [5]

Laboratory of Intelligent Interfaces , University of Mexico, UNAM.
[1] savage@servidor.unam.mx, [2] deyael@mezcal.fi-b.unam.mx,
[3] bowser@mezcal.fi-b.unam.mx, [4] jhespinosa@ieee.org,
[5] edna@mezcal.fi-b.unam.mx

Abstract. In this paper we describe a system to simulate and operate Virtual and Real robots (ViRBot). The system consists of several layers that control the operation of real and virtual robots. A 3D graphic engine interface allows to test multiple virtual robots, that are a close simulation of the real ones, the virtual robots are able to execute the same commands, using a robot's programming language, that real robots can, including behaviors, movement equations and sensors readings.

Introduction

ıe development of algorithms to control mobile robots it is necessary to simulate using virtual robots, in order to test their performance before implementing in the real ones. This is done because sometimes it is too expensive or unfeasi- to execute hundreds of testing operations with a real robot. Therefore, we have :loped a system, named the ViRBot, where operational algorithms for mobile ıts can be tested using virtual robots with the same conditions as if they were the ones. The main idea is to be able to control the real and virtual robots indis- :ly and to test the mobile robots algorithms using first the virtual robot and the one later. The ViRBot system allows the testing of multiple virtual robots, at same time, of different forms and sizes, multiple viewpoints, wired frame es, shaded or textured polygons, interaction with the system using different faces such as keyboard, mouse, joy pad and internet. Every part of the ViRBot developed by the authors of this document at the laboratory of Human Inter-

he virtual environment was developed using a virtual reality toolkit called WTK :e8 that gave us a powerful tool to program 3D world applications using mod- 1]. Using this tool is also possible to create boxes, tables, buildings, factories, ;eons, labyrinths and everything needed to construct different environments the virtual robots can be tested.

:lbukh, M. Hernández Cruz (Eds.) Avances en la ciencia de la computación en México,

2 Mobile Robot Operation

There are several architectures to control mobile robots, our architecture is ba the Aura architecture developed by Ronald Arkin [2] [3]. In our approach the robot's operation is formed by several layers, see Figure 1, each one having cific function that in whole control the behavior of the robot.

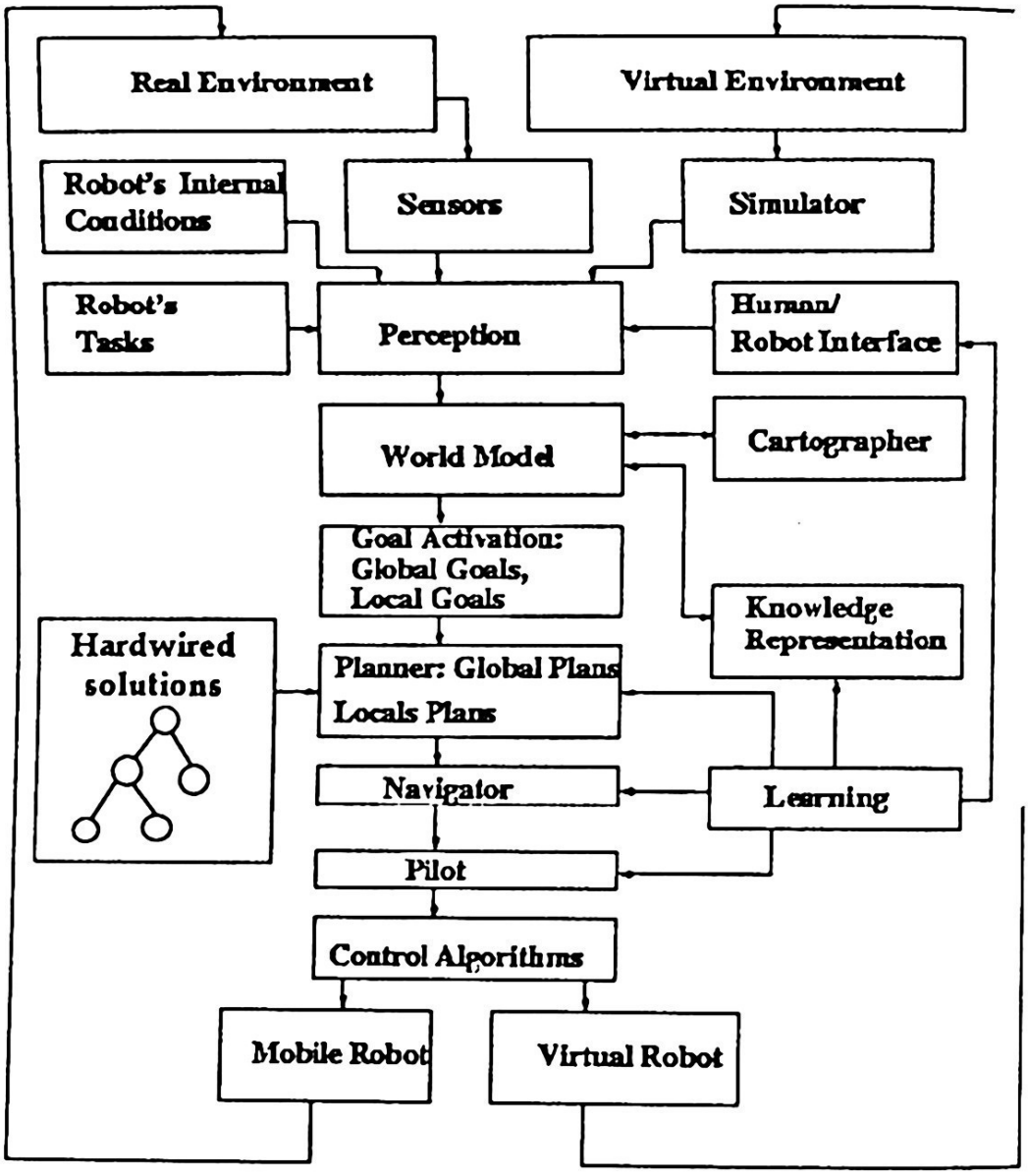

Fig. 1. Virtual and Real Robot (ViRBot) System.

Each of the layers are described briefly in the following sections with mo tails on the virtual environment and the simulator.

Perception

of the main objectives of the perception module is to obtain a symbolic presentation of the data coming from the robot's internal and external sensors, om the Human/Robot interface, from the robot's tasks and if chosen from the nulator. The symbolic representation is generated after applying digital signal ocessing algorithms on the data generated by the sensors. With this symbolic presentation a belief is generated, later with the world representation situation cognition is created.

Internal Sensors

ch of the sensors is represented by a data structure that has the following ele-nts: sensor's name, sensor's type, sensor's position in the robot and a set of the sor's values. We use a B14 robot, developed by Real World Interfaces, that it has wheel encoders and a battery level detector.

External Sensors

robot has an array of 8 infrared sensors, 8 sonar sensors, 20 tactile sensors and a leo camera [8].

Simulator

the ViRBot system, it is possible to simulate each of sensors' signals according to hysical model of the sensors. Thus, when the system is tested using the virtual ot, the simulator provides each of the sensors' values. A user may include his simulation algorithms [3]. The simulation of the sensors is explained in detail ection 3.

Human Robot Interfaces

communication between the robot and a user is done by several means: speech ognition, text provided by keyboard and control pads.

Robot's Tasks

robot is programmed to perform certain tasks, such as picking objects and de-ring them to another place, during the day at a certain time.

2.7 World Model

The world model module uses the belief generated by the perception modu together with the information provided by the cartographer and the knowledg resentation it generates a situation that needs to be solved.

2.8 Cartographer

For every room of the working environment there is a representation of how the rooms are interconnected between them, as well as each of the known ob included in them. The links between rooms are used later to form a tree that the root the origin room and one or several leafs have the destination room.

The known obstacles are defined as polygons which consist of a clockw dered list of its vertex. Representing the obstacles as polygons makes easi search for a solution to navigate from one point to another. They are represer a fact in an expert system as follows:

(polygon type obstacle' room obstacle' name x0 y0 ... xn yn)

where type represents the type of obstacle, such as a wall, a desk, etc. The xi represents the coordinates of the obstacle. From this obstacle representation den areas are created, which are areas not allowed for the robot to enter, built by growing the polygons by a distance greater than the radius of the ro consider it as a point and not as a dimensioned object [7]. Figure 2 shows bidden areas of the experiment's room. It is possible to create the config space in this way because our robot has cylindrical shape.

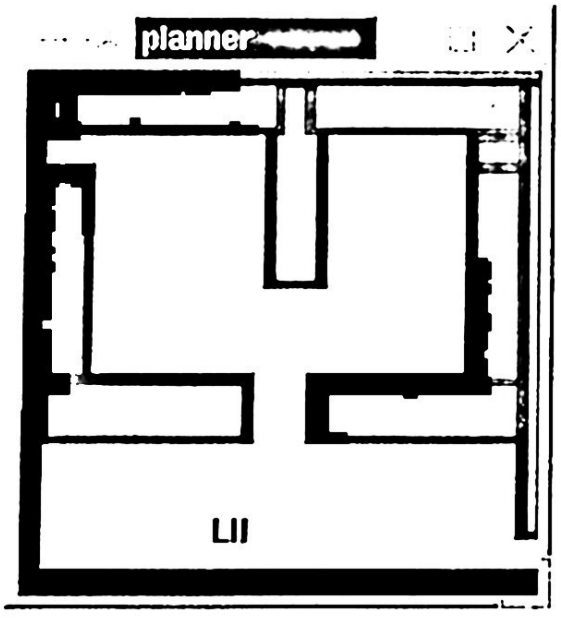

Fig. 2. Symbolic representation of the experiment's room.

Knowledge Representation

expert system is used to represent the robot's knowledge. In an expert system, knowledge is represented by rules, each one contains the encoded knowledge of expert, that is, the actions that the robot would do if certain conditions were met. The environment was defined as facts in an expert system, we used the expert stem shell called CLIPS, developed by NASA [4], in which we add a TK graphics vironment and sockets communication that allows to send data to it and to the bot through Internet.

0 Goal Activations

ven a recognized situation, a set of goals is activated in order to solve them.

Options

of hardwired procedures that solve, partially, specific problems.

Planner

basic problem of planning the robot's movements is reduced to: Given the ini- position and heading of the robot **A** in space **W**, a path τ specifying a continuous quence of positions and headings for the robot **A** must be found in order to avoid lisions with the obstacles $\mathbf{B}_i$'s, beginning in the initial position and heading and ishing in the goal position and heading. If there is not such path, the impossibility solve the problem is reported.

Planning is defined as a procedure or guide for accomplishing an objective or This requires searching a space of configurations to find a path that corre- nds to a set of the operations that will solve the problem. For example, in our tem we may want the robot to move from one place to another and to take a ture. This problem will be solved by finding a sequence of operations that leads m an initial state to a goal state. Then the objective of planning is to find a se- nce of physical operations in order to achieve the desired goal.

The robot's planning organization consists of several hierarchical layers. The top er is the planner, which takes as input an initial and a final state and an environ- nt description, and produces an overall plan that will lead the robot from the ial to the final state. If the command was to move the robot from one room to ther, the planner would find the best sequence of movements between rooms the robot reaches its destination. Inside each room it also finds the best path sidering the known obstacles, that represent some of the objects in the room. h of the objects in the environment is represented by a set of polygons whose are by the planner, in order to find the best path avoiding those that interfere with goal.

2.13 Navigator

The navigator takes the overall plan and finds a set of trajectories that cons distances and angles that the robot needs to execute in order to reach its destin Given a set *(x1, y1, x2, y2...xn, yn)* of points obtained by the planner and a tim visit them, the navigator finds the angle of rotation θi, a distance *di* and a speed reach all of them.

2.14 Pilot

The pilot takes the trajectories generated by the navigator and executes them. cally, it has to move the robot a distance D_i and turn θ_i radians in each step i search for unknown obstacles to the planner and tries to avoid them using beha control. look for

2.15 Controller

It controls the robot's motors and reads incoming data from the motors and ser The mobile robot that was used to test the ViRBot system is a cylindrical type bile robot equipped with a wheeled base that allows two types of motion: transl (movement parallel to the robot head alignment) and rotation (movement pe dicular to the robot head alignment), see figure 3. The robot has motion contrc that control its movements, it also has odometers to measure distances.

Fig. 3. The Robot B14.

Learning

system can learn how to solve new problems using genetic algorithms, babilistic methods, Hidden Markov models, etc.

Sensors' Simulation

simulation module allows the creation of virtual robots with multiple sensors of ferent types such as contact, reflective, sonar and infrared sensors.

Contact sensors. In order to simulate contact sensors we developed a simple polygon cross algorithm between a contact prism (sensor) and all the world (obstacles), so if the prism cross an obstacle the sensor's status change to on, indicating a collision with an obstacle.
Reflective sensors. These types of sensors are useful to create line followers robots. The virtual robot gets the track/floor values returned by the sensors in the robot's sensor position. To simulate them an average of the texels are calculated in the position where the sensors are. Texel is a texture element, the base unit of a texture over a 3D polygon [1].
Sonar sensors. Sonar sensors are simulated by getting the position where the sonar beam strikes an object.
Infrared sensors. As in the sonar sensors to simulate the infrared ones it is necessary to get the position where the infrared beam strikes an object.

Collisions with obstacles are checked all the time, the purpose of this is to simu- the robot's physical interaction with its environment. Thus the virtual robot can pass through any simulated object, if it strikes an obstacle then it needs to exe- additional behaviors related to the strike. The collision algorithm is based in nding box and polygon crossing.

user of this system can incorporate its own simulation of a particular sensor, a sonar specialist may test the operation of a particular sonar sensor whose sical behavior has been modeled and include it in the system using standard ++ routines.

andom noise can be added to the readings of the sensors, the user can select ei- if noise is added or not, as well as the mean and the variance of the Gaussian lom variables that are added.

Robot's Movements

simplest way to define the movement of a robot is through a sequence of ght lines [5], see figure 4. Basically to follow a straight line the robot requires a sition from a straight line to another, and this is obtain by coordinate transform a-

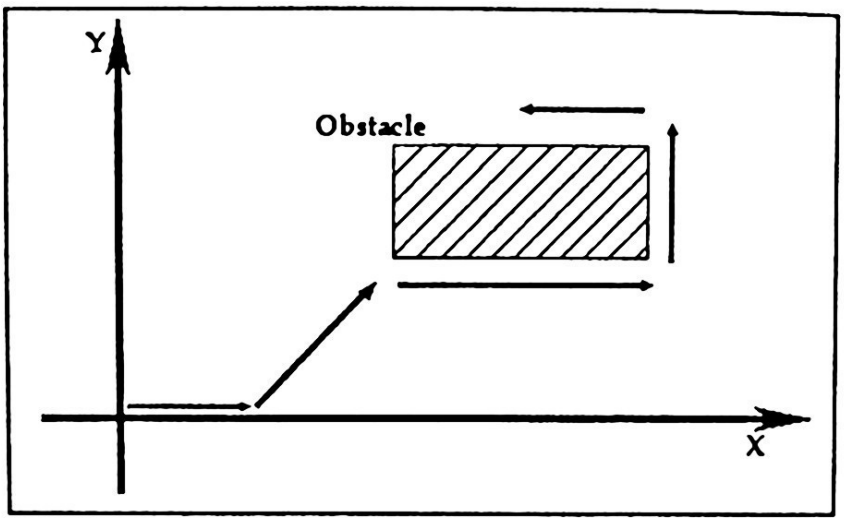

Fig. 4. Movement of the robot through straight lines.

The robot's position is described, in time i, by (x_i, y_i, θ_i) where θ_i represents the of the robot with respect of x axis. The system of coordinates x', y' represent new axis of coordinates after a rotation and displacement of the robot. Fig shows these systems of coordinates from which following equations are obtaine

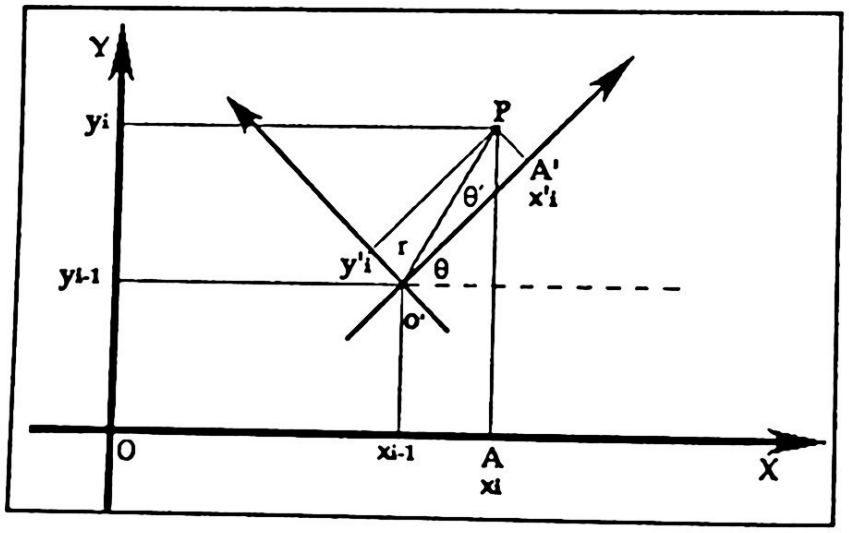

Fig. 5. Coordinate system.

$$x_i = \overline{OA} = x_{i-1} + r*\cos(\theta+\theta') = x_{i-1} + r*\cos(\theta)*\cos(\theta) - r*\sin(\theta)*\sin(\theta) .$$

$$y_i = \overline{AP} = y_{i-1} + r*\sin(\theta+\theta') = y_{i-1} + r*\sin(\theta)*\cos(\theta') + r*\cos(\theta)*\sin(\theta') .$$

$$x_i' = \overline{O'A'} = r*\cos(\theta') .$$

$$y_i' = \overline{A'P} = r*\sin(\theta') .$$

$$x_i = x_i'\cos(\theta) - y_i'\sin(\theta) + x_{i-1} .$$

$$y_i = x_i'\sin(\theta) + y_i'\cos(\theta) + y_{i-1} .$$

$$\theta_i = \theta_i' + \theta_{i-1} . \quad (7)$$

and y_i' represent the displacement of the robot and θ' the angle of rotation in the axis. If $y_i'=0$, then if the robot rotates θ_i radians and advance a distance $x_i=d_i$ the equations for the new position of the robot are:

$$x_i = d_i*\cos(\theta_i) + x_{i-1} . \quad (8)$$
$$y_i = d_i*\sin(\theta_i) + y_{i-1} . \quad (9)$$

An important part of the simulation are the movement equations, that have to be ved in any step of the simulation engine. These equations take in consideration speed and acceleration that the robot needs to follow, the next equations are ed:

Position: $$x(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 . \quad (10)$$

Speed: $$\frac{dx(t)}{dt} = a_1 + 2a_2 t + 3a_3 t^2 . \quad (11)$$

Acceleration: $$\frac{dx(t)^2}{dt^2} = 2a_2 + 6a_3 t . \quad (12)$$

Given the initial and final position, and the initial and final speed the a_i s con- ınts are found.

user of this system can incorporate its own simulation of the movement of the ɔot, thus a researcher wishing to test a control algorithm may include it in the stem, using standard C/C++ routines.

o make more realistic the movements of the simulated robots, random noise can added to the final movements, the user can select either if noise is added or not, well as the mean and the variance of the Gaussian random variables added.

Simulation Process

simulation process [6] is described as follows, see figure 6:

Create and initialize the 3D world. The world is created adding all the objects that will exist in it, including rooms, models, lights and textures.
Create the robot. Here, using simple objects or 3D models, all the robots that will be used are composed, loaded and initialized to work in the system.
Assign default behaviors to the robot. Default values and tasks needed to make the robots work are attached.
Initialize the communication. Here all the communication devices are initialized. If it is required, the communication with the real robot via Internet sockets starts.
Wait for an order. Constantly the data coming from an internet socket, joy pad and keyboard buffers are sensed, and the information received is analyzed.

- Execute an order. When a valid command is received it is necessary activ certain tasks needed to execute the command.
- Finish 3D world. Kill all processes attached to world; send finish commun tion signals and close all windows related.

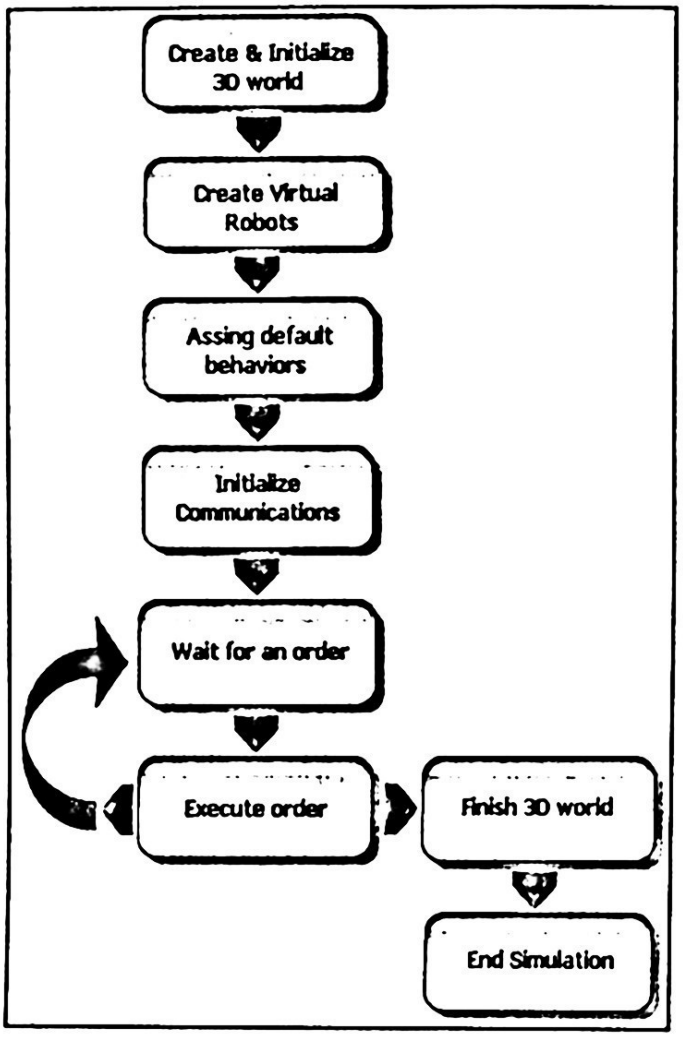

Fig. 6. Blocks diagram of the simulation process.

6 Robel

To improve the way to program and test robots, we have developed a robot' gramming language, robot behavior language or Robel, as an easy way to progr and assign basic tasks (commands) to the robots. In order to use Robel, it is nec sary that in the robot has been installed a Robel Virtual Machine (RVM), that cute commands written in that language. Basically any virtual robot in the simu tion is loaded with the RVM. We developed the RVM for a Real World Interfa robot, for a custom robot based on the 6811 micro controller and for the virt robots. The Robel's instruction set is composed by 16 basic instructions which

classified in movements, status, sensors' readings, internet interfacing and condi-
als:

Movements. These type of instructions are used to move the robot in its environ-
Instructions as moving, rotating and moving after rotating are examples of
For example the command mv d_i, θ_i, t_i rotates the robot θ_i radians, then the
advances a d_i distance, in decimeters, and in t_i seconds the action is executed.

tatus. As in real world, it is necessary that the robot knows where in the world it
ocated. Instruction as getting and setting position are examples of it.

ensors. Maybe the most important commands are the sensor related commands,
it is possible to get the values of all the defined sensors that the robot has.

onditionals. Instructions that control the flow of the program.

ser's functions. An user can include his own subroutines written in C/C++ al-
ng him to include complex behaviors to the virtual robots. Figure 7 shows the
lation of the robots on the virtual environment. The upper left window shows a
view of the 3D environment; the lower left shows a faraway view of one of the
ts; the upper right window shows the point of view of one the robots; and the
right window shows a console where the commands are typed.

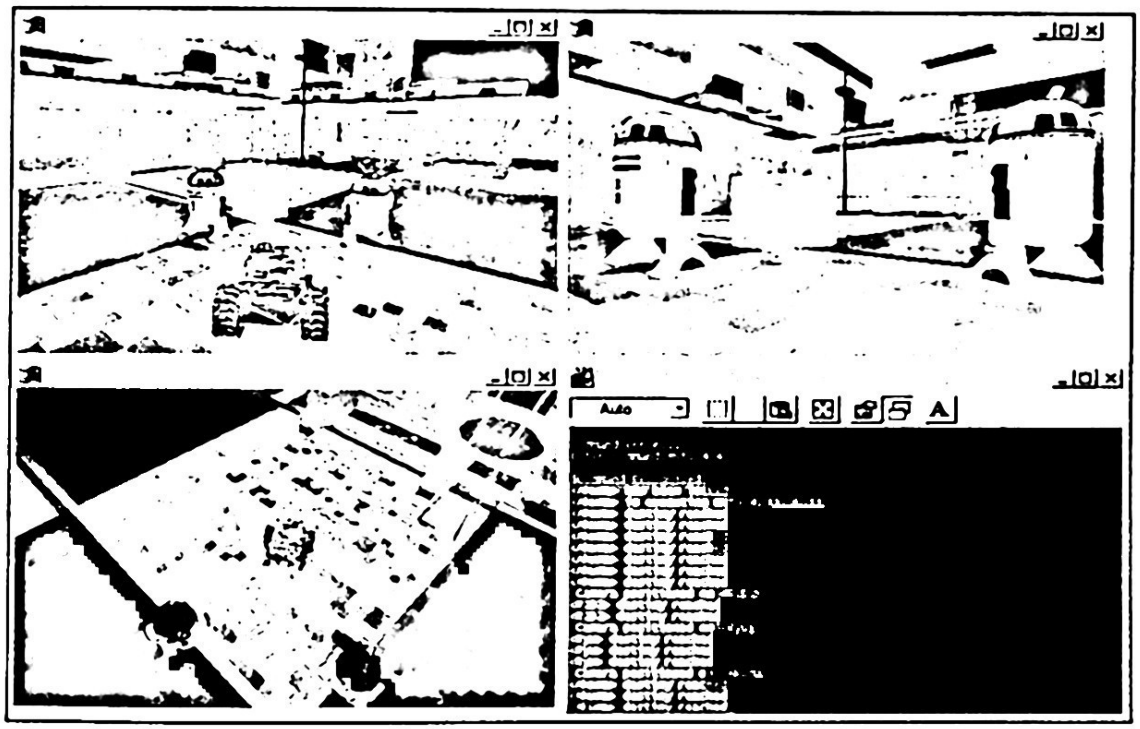

Figure 7. Simulation of mobile robots.

Robel Program

following code is an example of Robel programming language. This algorithm
generated using genetic algorithms in order to evade dynamic obstacles [9].

```
#DEFINE MAX_VALUE 5.0
#DEFINE MIN_VALUE 4.5
int state
float s1, s2
state <- 0          @1
while 1 < 2         @2
READ:                                   @3
       shs sonar 2                      @4
       s1 <- RegF1                      @5
       shs sonar 4                      @6
       s2 <- RegF1                      @7
       if state = 0                     @8
            if s1 < MIN_VALUE           @9
                 state <- 1             @10
                 goto ETI1              @11
            endif                       @12
            if s2 < MIN_VALUE           @13
                 state <- 2             @14
                 goto ETI1              @15
            endif                       @16
       endif     @17
       if state = 1                     @18
            if s2 < MIN_VALUE           @19
                 state <- 2             @20
                 goto ETI1              @21
            endif                       @22
            if s1 > MAX_VALUE           @23
                 state <- 0             @24
                 goto ETI1              @25
            endif                       @26
       endif     @27
       if state = 2                     @28
            if s1 < MIN_VALUE           @29
                 state <- 1             @30
                 goto ETI1              @31
            endif                       @32
            if s2 > MAX_VALUE           @33
                 state <- 0             @34
                 goto ETI1              @35
            endif                       @36
       endif     @37
ETI1:                                   @38
       if state = 0                     @39
            mv 3 0                      @40
            goto READ                   @41
       endif     @42
       if state = 1                     @43
            mv 0 0.5236                 @44
            goto READ                   @45
       endif     @46
       if state = 2                     @47
            mv 0 -0.5236                @48
            goto READ                   @49
       endif     @50
BYE:                                    @51
wend                                    @52
```

Conclusions

ng the ViRBot system to control virtual and real mobile robots is an easy and way to develop complex behaviors in the robots. The same algorithms tested the virtual robots have been tested, with slight variations, in two different platns: a custom robot with a 6811 micro controller, and a real world interface morobot B14, with Linux operating system. The advantage of using simulated ots instead of the real ones is that we can test extensively robots' algorithms in simulated ones before the final versions are executed in the real robots.

ferences

oley J., Van Dam A., Feiner S. Hughes J. "Computer Graphics: Principles and Practice C" Addison Wesley, 1996.

.C. Arkin and R. Murphy. "Autonomous Navigation in a Manufacturing Environment." EE Transaction on Robotics and Automation 6(4): 445-452 1990.

.C. Arkin. Behavior-Based Robotics. Cambridge, MA: The MIT Press, 1998.

iarratano, J., and Riley, G. "Expert Systems: Principles and Programming", 3rd Edition, oston, PWS Publishing Company, 1998.

C. Latombe, "Robot Motion Planning", Massachusetts, USA: Kluwer Academic Pubshers, 1991.

orldToolKit Reference Manual Release 9. Engineering Animation Inc. 1999.

omás Lozano-Pérez, "An algorithm for planning collision-free path among polyhedral ostacles," Communications of the ACM, vol.22, pp.560-570, October 1979.

alavanis K, Hebert T., Kolluru R., and Tsourveloudis N., "Mobile Robot Navigation in Dynamic Environments Using an Electrostatic Potential Field," IEEE Transactions on stems, man, and cybernetics, Vol. 30, no. 2 pp 187-196. March 2000.

ade Mattias. "ETS Retro service Chalmers University of Technology". Sweden, 2002.