RESEARCHES IN THE DEVELOPMENT OF A SIMULATOR FOR THE TRAINING OF INTERVENTION ROBOT OPERATORS

Eng. Ioan ANDREESCU*, CS II, SC ICPSP SA Bucuresti,* ROMANIA
Eng. Nicolae MORARU*, CS I, SC ICPSP SA Bucuresti,* ROMANIA
Assoc. Prof., Ph.D., eng. Remus BRAD*, ULB Sibiu,* ROMANIA
Lecturer, Ph.D., eng. Beriliu ILIE*, ULB Sibiu,* ROMANIA
Alexandru DOROBANȚIU*, undergraduate ULB Sibiu,* ROMANIA
Andrei MARINICĂ*, undergraduate ULB Sibiu,* ROMANIA

Abstract*: The paper presents the software and hardware structure of a simulator model for the training of robot operators. Conceived and programmed in the XNA environment,, the application software contains 4 training levels and one demo. In the aim of a realistic training, the simulator is wired directly to the robot control panel, using an original designed micro-controller interface.*

1. Introduction

In order to take advantage of the capabilities offered by robots, first of all, their operators have to learn a completely new set of skills. These skills include understanding the state of the robots by interpreting sensor data, getting used to the visual data provided by a remotely operated robot and becoming aware of the capabilities and limitations of the robot they are operating. Even these basic set of skills can be difficult to apprehend and non-intuitive. At the moment, professional instructors teach these skills as part of a multi day training course.

A realistic simulator is a crucial tool in training EOD robot operators. EOD robots are remotely operated robots that feature multi-jointed arms with hand-like manipulators. They are called upon to perform tasks such as finding, removing and disarming explosive devices. They are expensive and cost upwards of 200.000 Euro per unit. Mistakes are costly both from a financial point of view, but also as regarding to safety. Moreover, the robots are in constant use, thusly their availability for training is limited. A high-fidelity simulator would serve to improve the skills and the level of confidence of the operators, and increase operational safety. As an example, EADS [1] (European Aeronautic Defense and Space Company) has chosen a simulator named Vortex for its high-fi delity robot simulation (figure 1) [2]. Other simulation software as WEBOTS 5 [3], open source [4], Matlab based [5] or commercial [6] are also available. A short overview of general-purpose robotics software platforms currently available for service robotics applications is presented in [7].

Figure 1. The Vortex simulation

On the other hand, simulation allows researchers, designers and users to construct robots and environments for only a fraction of the cost and manufacturing time involved by real systems. They differ significantly from traditional CAD tools in that they allow a better study of the geometry, kinematics, dynamics and motion planning involved.

Most existing dynamic simulators are either designed for specific types of environments or for simulating the motion of specific type of robots. The structure of this simulator is not limited. It is also possible to extend the functionality of the simulators, in order for them to be able to perform simulations of different paradigms, due to the flexibility of the design principle of the programs.

2. The simulator design and implementation

Virtual environments have been seen to be capable of accurately representing many real world scenarios. This type of simulator is an investigation of the use of virtual environments created using commercial software technology (examples: XNA Game Studio Express 1.0 Refresh, 3D Studio Max 9). Many problems associated with robot operator training can be reduced or eliminated using a virtual approach and many new opportunities are offered by this system. While they may never fully replace real life training, we believe that game applications that simulate the use of robots can become an important tool in the training of robot operators.

Virtual training provides several benefits over the traditional instruction. First of all virtual training is free of dangers both to the people and to the equipment involved. This means that a novice user can gain experience before he can operate a real robot, and an advanced user can practice riskier scenarios without taking any real risks. Virtual training also allows users to easily train in a variety of scenarios. Even the most bizarre and unlikely scenarios can be created in a virtual environment and rehearsed as often as wanted. Real world locations and obstacles can also be simulated using a game (in the form of commercial software). Thusly, robot operators do not have to use the equipment or visit a specially designed training center. With virtual robot simulations, robot training can occur at any time in any place and very cost-effectively. On the other hand, with today's commercial software gaming, common robot environments can be accurately simulated using as little as a personal computer.

Basically, a simulator implements both levels of robot locomotion available in operating systems:

- A high–level driving controller for straight driving and for curve segments, as well as turning on the spot for differential drive vehicles
- A low–level driving interface with direct simulation of motor actuators for vehicles with differential drive mechanisms

The high-level driving controller and the low-level driving interface implement the drive kinematics. The high-level driving controller implements drive functions for a $v - \omega$ (velocity – angular velocity) interface. These functions can include commands as: Drive Straightly, Turn, Turn On The Spot, Acknowledge Ready for Driving (see figure 2). In a future development they can include an implicit PID controller for velocity and control of the position.

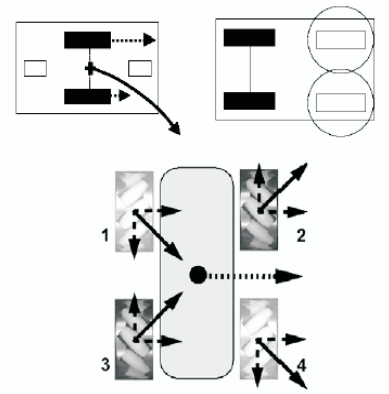

Figure 2. Different drive modes

A low level interface allows direct manipulation of vehicle motors, in a drive mode. It is known that the robots can implement Ackerman drive or omni-drive mode. In the future, a number of parameter files will determine the physical dimensions of robots, their performance and their modeled appearance. Such files will be used to describe each robot type. "Environmental" files will describe the shared driving scenery.

In our case, the application software can be divided into three main parts:

- The 3D simulation of the scenery and of the robot
- The implementation of graphics and of the algorithms that emulate physical phenomena
- The interface between the operator panel and the PC

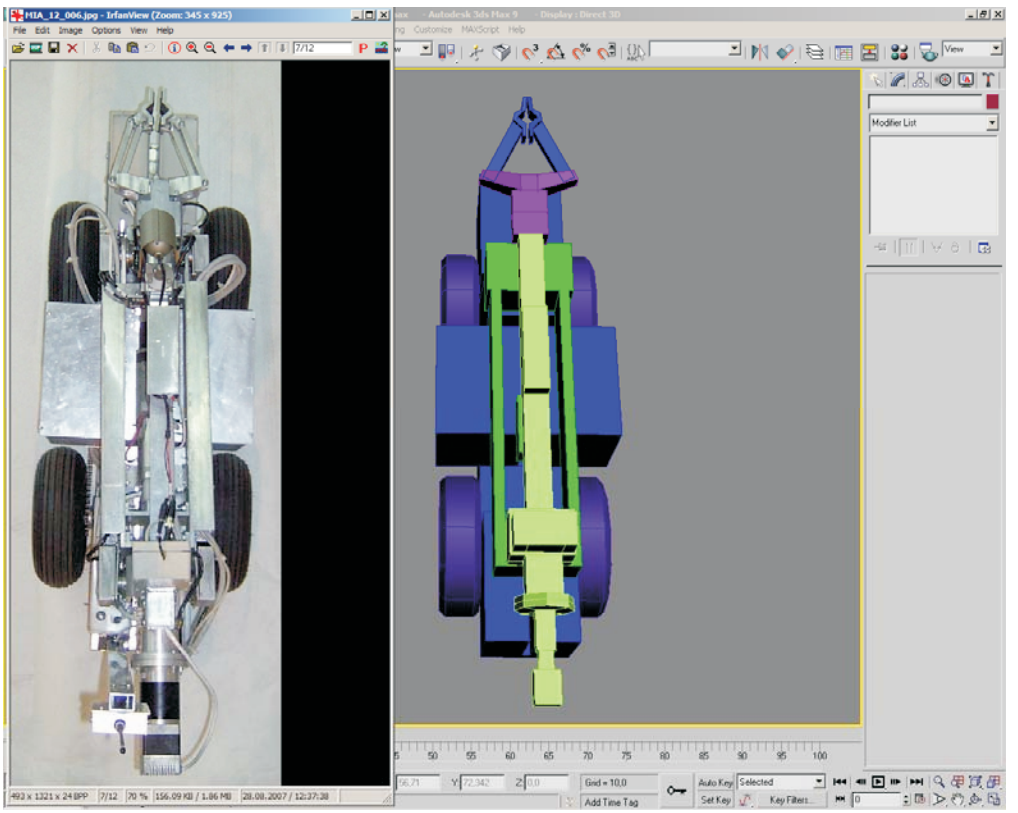
Figure 3. The real view of the robot and the corresponding 3D model

The scenery and the replica of the robot were designed using 3D Studio Max, as shown in figure 3. An efficient and accurate polygonal representation has allowed real-time processing on computers. There are 4 scenarios that comply with 4 difficulty levels: a simple maze (DEMO), a general store, an aircraft interior and a street (figure 4). A good texturing contributes to a very realistic look of the virtual representation (most textures originate from real pictures).

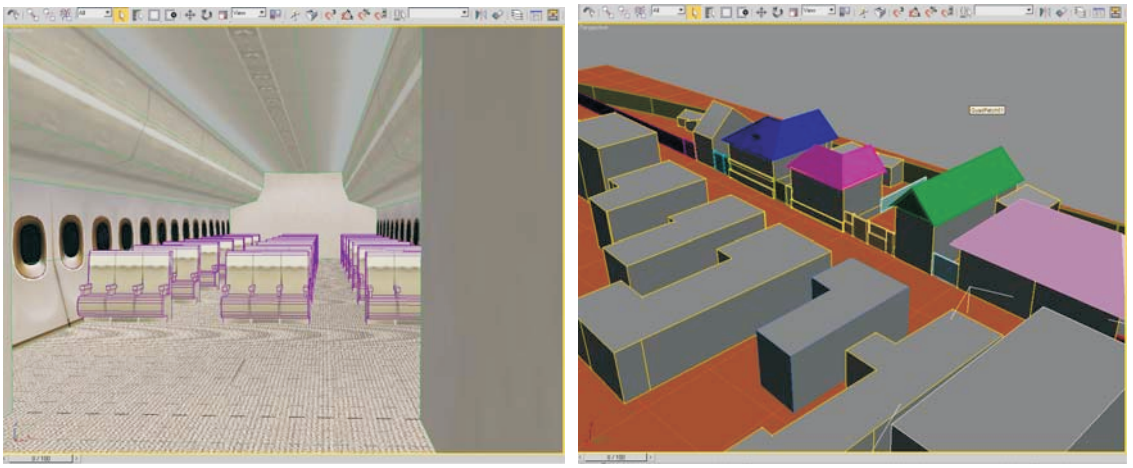

Figure 4. Modeling of an airplane interior and a town street in 3D StudioMax

The models were loaded in the application by functions of the XNA Framework [8]. In order for this to be possible, they had to be first exported to the DirectX format. This was done using Panda DirectX Exporter, a 3D Studio Max plug-in. For graphic programming a new Microsoft set of tools, named XNA Game Studio, was used [9]. The physical engine was designed by directly implementing mathematical algorithms (see figure 4 and 5).

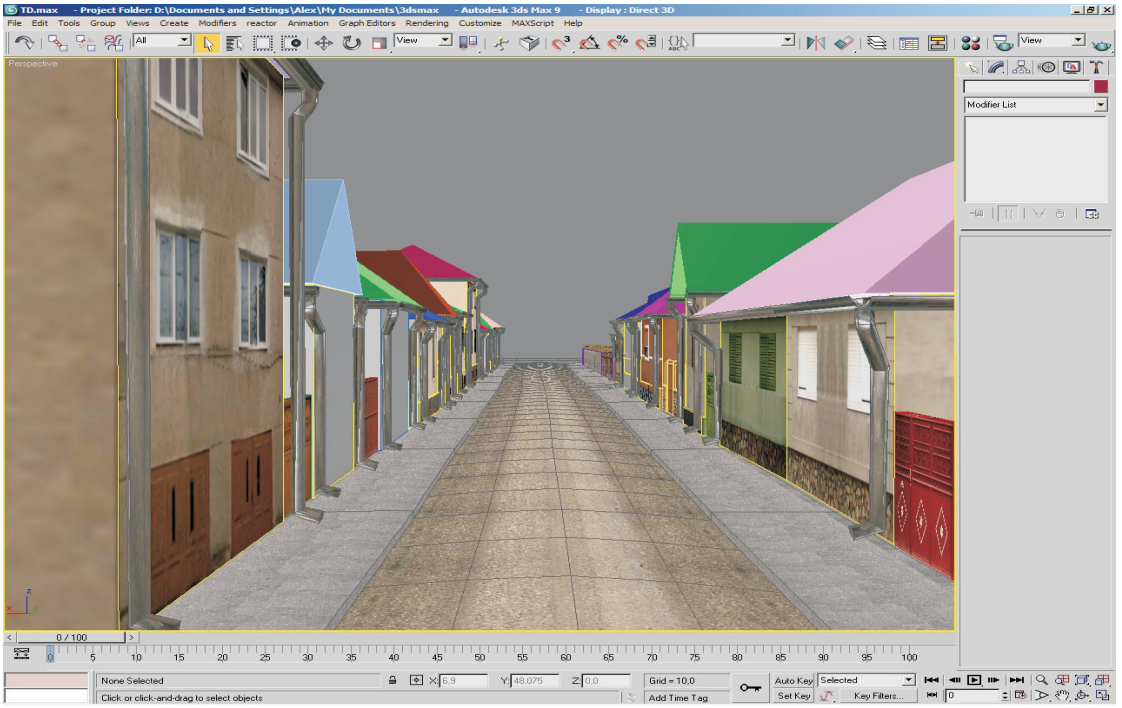

Figure 5. One of the simulation environments

An original aspect of the proposed simulator is the collision detection and processing technique, used in the following cases:

1. For the accurate simulation of the collision of the robot or of the "bomb" with map elements, a "height map" was used, i.e. a matrix that retains point heights regularly, at equal intervals on both horizontal axes. The actual tests assure that certain points on the robot ("collision points") do not enter "beneath" the map, i.e. they are not allowed to get to a height that is lower than the height map at the same horizontal coordinates. The height of the map in a random point is deduced with respect to the closest matrix nodes. Also, tests assure that certain points remain at any moment "beneath" the map, so that the robot doesn't fall off certain surfaces. The "Height map" is necessary in order to position the robot correctly and to turn it accurately on the slope. The collision constraints are generated relative to a maximum threshold, so that ceilings and other high objects be excluded.
2. For the collision processing between the manipulator and the robot platform, a similar "height map" is used, but this time taking into account the relative coordinates of the manipulator with respect to the robot.
3. For the collision of the robot with the "bomb", the "bomb" is modeled as a convex polyhedron and the collision testing is performed as a Boolean test for the inclusion of the points on the robot in the polyhedron. It is also necessary to discriminate between the "tight grip" and the mere "pushing" of the "bomb".

For an effective training of the robot operators, the commands are carried out from a real operator panel with joysticks and switches. The PC is linked with the operator panel by an interface equipped with a Microchip PIC 18 F 248 programmable micro-controller. The controller was programmed using a Microchip ICD2 programmer and MPLab 7 software. The ADCs and Boolean converters (which are incorporated in the microchip) read the state of the buttons and of the joysticks continuously. The data is processed within the controller and sent as a data stream via the serial port, through a RS232 COM port. The data stream is sent periodically (9 times by second) and is read and decoded by the application. The communication protocol consists of 7 bytes of data, encoded as shown on table I.

Table I. Communication protocol

Byte	Used Bits	Meaning
0	0-7	'0' – signals the beginning of the transmission
1	0-7	'X' – signals the beginning of the transmission
2	0-7	Position of the right joystick (absolute value)
3	0-7	Position of the left joystick (absolute value)
4	2, 4, 5, 6	Direction of joystick movement
5	0-7	Robotic arm movement
6	0-1	Robotic arm movement
	2-5	Fire, break, radio transmission, spotlight

3. Conclusions

Designed for real-time, interactive simulation, this simulation kit strikes a balance between fidelity and speed. We hope that it will prove itself useful in applications including operator training, product design and analysis, and a pioneer in the research for unmanned vehicles.

Bibliography

[1] *, available at http://www.eads.net/
[2] *, available at http://www.cm-labs.com/
[3] *, available at http://www.cyberbotics.com/products/webots/
[4] T. Ishimura, T. Kato, K. Oda and T. Ohashi, An Open Robot Simulator Environment Lecture Notes in Computer Science, vol. 3020, pp. 621-627, Springer Verlag, 2004
[5] M. Fridenfalk and G. Bolmsjö, The Unified Simulation Environment - Envision telerobotics and Matlab merged in one application, Proceedings of the Nordic Matlab Conference, pp. 177-181, Oslo, Norway, October 17-18 2001
[6] *, Microsoft Robotics Studio, available at http://msdn2.microsoft.com/en-us/robotics/
[7] M. Somby, A review of robotics software platforms, available at http://www.linuxdevices.com/articles/AT5739475111.html
[8] B. Nitschke, Professional XNA Game Programming: For Xbox 360 and Windows, Wiley Publishing 2007
[9] S. Cawood, P. McGee, Microsoft XNA Game Studio Creators Guide, McGrath-Hill 2007

Acknowledgments

This document is build on Intervention Robot work done by the S.C. ICPSP S.A. research and development group. Simulation researches were conducted by Assoc. Prof. Remus BRAD in "Lucian Blaga" University of Sibiu laboratories. The work that provided the basis for this publication was completed at ULB Sibiu and supported by funding under Research Grant no. 1371/1.06.2007 with S.C. ICPSP S.A. The final tests and simulation stand have been carried out in S.C. ICPSP S.A. laboratories.