Production Systems and Information Engineering
Volume 2 (2004), pp. 121-129

BEHAVIORAL ROBOT SIMULATION FOR PRODUCTION SYSTEMS

TAMÁS JUHÁSZ
Department of Control Engineering and Information Technology,
Budapest University of Technology and Economics,
Mobile and Microrobotics Laboratory,
Pázmány Péter sétány 1/d, I.B.411., Budapest, H-1117, Hungary
tjuhasz@seeger.iit.bme.hu

[Received November 2004 and accepted January 2005]

Abstract. The intelligent production systems have become more and more important for modern enterprises in the industry. Usually the design process of a new product has great many stages: each of them can introduce errors that are seriously retarding the heavy schedule. During the development of a new product a prototype is created usually in early design phase. This prototype can be used for verification purposes to hinder the unwanted events owing to design mistakes appearing in the final (commercial) product. Sometimes the manufacturing of the prototype can be a risky, expensive and time-consuming process – especially if the result does not meet our strict demands (thus we have to create a new prototype again). It is a modern approach to use virtual prototyping (by incorporating virtual reality) to cut down costs and increase productivity. This paper deals with this complex simulation and modeling problem through presenting our current simulator project at the department. Using our new kind of simulation architecture we can introduce physical hardwares to these tests to give better compliance with the behavior of the final product. The objectives and the future steps in the development of the software will also be discussed in this paper.

Keywords: Virtual reality, virtual prototyping, hardware-in-the-loop testing

1. INTRODUCTION

Virtual prototyping enables continuous iterative design and testing via simulation, allowing the developer of a complex manufacturing system to find errors earlier in the design phase. Today there is a wide variety of simulators on the market, but most of them are trade-, model- or application specific ones. A summary was created by Yiannis Gatsoulis (University of Leeds) who was claimed by the CLAWAR community to analyse the state-of-the-art of this field a few years ago [1]. It contains a snapshot of the available softwares (environment editors, image processing and control libraries, system simulators, etc.) that are in connection with

this research area. It assesses the cost, usability, expandability and rapid development abilities of the given applications.
In general the ideas and techniques developed during the simulation process yield to ideal conditions to raise synergy: a catalytic effect for discovering new and simpler solutions to traditionally complex problems. Using virtual prototyping by means of behavioral simulation reduces the time-to-market, as it shows the inconsistencies early in design phase. Abstract modeling with CAD tools, enhanced conceptual design and moving up life-cycle assessments by virtual prototypes allow devising the optimal layout and the best mechanical architecture of the system.

2. THE MICROSOFT .NET FRAMEWORK

The .NET platform (Figure 1) offers rapid application development using the new C# language [5] and its event-delegate based communication model. The .NET framework is a new development framework with a new programming interface to operating system services and APIs, integrating a number of technologies that emerged during the late 1990s [2]. Incorporated into .NET are COM+ component services; the ASP web development framework; a commitment to XML and object-oriented design; support for new web services protocols such as SOAP, WSDL, and UDDI; and a focus on the Internet.

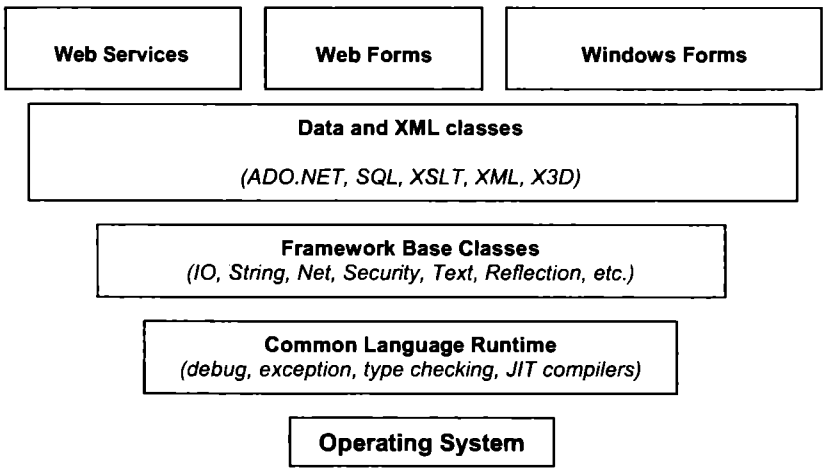

Figure 1: The .NET platform

Aside from embracing the Web, Microsoft .NET acknowledges and responds to the distributed computing and componentization trends within the software industry today:

- Distributed computing:
  Simplifies the development of robust client/server or Web-based applications. Current distributed technologies require high vendor-affinity and lack interoperation with the Web. Microsoft .NET provides a Remoting architecture that exploits open Internet standards, including the Hypertext Transfer Protocol (HTTP), Extensible Markup Language (XML), and Simple Object Access Protocol (SOAP).

- Componentization:
  Simplifies the integration of software components developed by different vendors. The Component Object Model (COM) has brought reality to software plug-and-play, but COM component development and deployment are too complex. Microsoft .NET provides a simpler way to build and organize components.

The so called .NET Remoting enables easy communication for individual applications. It uses open Internet standards (HTTP, XML, and SOAP) at its core to transmit an object from one machine to another across the Internet. In fact, there is bidirectional mapping between XML and objects in .NET. For example, a class can be expressed as an XML Schema Definition (XSD); an object can be converted to and from an XML buffer; a method can be specified using an XML format called Web Services Description Language (WSDL); and an invocation (method call) can be expressed using an XML format called SOAP.
Another important feature that makes the .NET platform and the C# language suited for creating a powerful simulation application is the ability to use native modules that are exploiting the hardware through embedded operating system devices. For example we can use Universal Serial Bus (USB) drivers to communicate with external hardwares in case of hardware-in-the-loop testing (discussed later, in the section 3.5 of this paper) in a straightforward way.

3. THE GLBOT.NET SIMULATOR PROJECT

Cooperating mobile robots are commonly used in modern manufacturing environments. They usually do logistics tasks by carrying payload such as assembly parts, tools for service robots and so on. The functionally integrated mobile platforms play a key role in the manufacturing procedure as their efficiency gives a significant part of the overall performance concerning the entire production cell. Thus realistic simulation of their behavior is basically important during the planning of the target process.

Now there is an ongoing simulator project [3, 4] (called: GLBot.NET) at the Department of Control Engineering and Information Technology in Budapest University of Technology and Economics. Our component-based simulator is being written in the new generation C# language [5]. It is utilizing the platform-independent .NET framework and the X3D environment that is a standard 3D scene description language introduced by the Web3D Consortium [6].

3.1. Our objectives

In these days the key objectives for virtual prototyping systems are modularity (concerning all devices and aspects) and interoperability (using the output from- or serve input to other related applications). We are trying to give general solutions for the upcoming problems staying apart from model- or application-specific constraints. One of our main objectives is to develop a simulator that maintains testing of individual components by means of hardware devices as well (so to support hardware-in-the-loop testing). The main components of the simulated system are proposed to be connected through a standard interface and designed such way that they could be replaced by a corresponding physical hardware.
An obvious effort has to be taken to use standard data formats for describing the manufacturing environment (as well as the mobile robots being embedded in it), which makes easier to separate the notional design from other manufacturing steps. The new X3D standard is a powerful and extensible open file format for 3D visual effects, behavioral modeling and interaction. It can be considered as a successor of the well-known VRML format. By providing an XML-encoded scene graph and a language-neutral Scene Authorizing Interface, it makes scene verification much easier and allows 3D content to be easily integrated into a broad range of applications. Its base XML language lets incorporating 3D into distributed environments, and facilitates moving 3D data between X3D-aware softwares.
The Extensible Modeling and Simulation Framework (XMSF) is defined as a set of Web-based technologies, applied within an extensible framework, that enables a new generation of modeling and simulation (M&S) applications to emerge, develop and incorporate [7]. Distributed simulation through XMSF makes possible the efficient testing of cooperating platforms which are commonly used in modern applications. The XMSF will enable simulations to interact directly over a highly distributed network – which can be achieved through compatibility with Web technologies – and will be equally usable by humans and software agents. XMSF must therefore support composable, reusable model components. The Extensible Markup Language (XML) is the cross-cutting technology for root data structure representations, with Resource Description Framework and ontology-tagset support for semantics. Some of the primary challenges for the XMSF are: providing open and extensible M&S capabilities, improving speed of development by stimulating

rapid growth of interoperable simulations and providing support for all types and domains of M&S (constructive, live, virtual and analytical).

3.2. Modularity in GLBot.NET

A modular mobile robot system has generally three major components (sensing, processing and locomotion) that are communicating with each other. All mobile robots use locomotion that generates traction, negotiates terrain and carries payload. Some robotic locomotion also stabilizes the robot's frame, smoothes the motion of sensors and accommodates the deployment and manipulation of work tools. The locomotion system is the physical interface between the robot and its environment. It is the means by it reacts to gravitational, inertial and work loads. Thus the locomotion system is the basis of a mobile robot's performance.
Each main robot component (sensing, processing and locomotion) can be considered as a separate module in the application. These modules are connected through a standard communication interface (Figure 2).

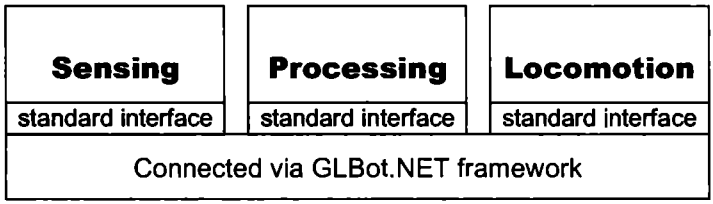

Figure 2: The modular simulator architecture

Telerobots, teleoperators, and remotely operated vehicles [8] belong to a class of machines used to accomplish a task remotely, without the need for presence on site. The modular architecture of the simulator, the .NET Remoting and the standard communication interface between these modules involve that the given sensing and locomotion components can be far away from the intelligent control component (running on different computers that are connected via a communication channel). Thus remote operation can be carried out in GLBot.NET.

3.3. The user interface of the simulator

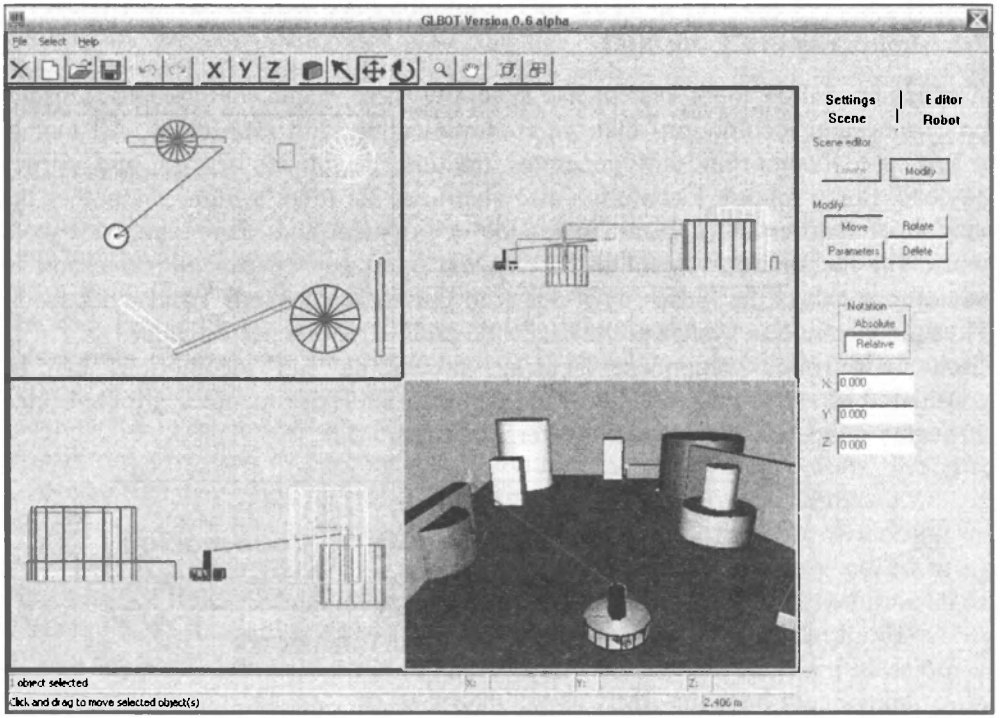

Figure 3: The user interface

Figure 3 shows a snapshot of the main screen of the application which is under construction at the moment. The example differential-driven mobile robot is equipped with a laser range finder and it is measuring the distance of a known marker using that device.

The program uses a CAD interface that is similar to the well-known 3DStudio MAX® software (proprietary of Discreet™) to let you easily design the desired virtual environment and mobile platforms. Many primitive object types are available (box, plane, cylinder, cone and sphere) and general 3D mesh objects can be imported from standard X3D files. All 3D objects (even cameras and lights) are nodes in a tree structure that are handled and stored hierarchically in the X3D scene description file. Thus one object can be a parent of another, where each object passes its transformation to its children. By construction all objects are the child of an unyielding root object (the one that has no parent at all), and they can be re-linked to their new parent (with the Link tool) as desired.

The nodes that are representing the robot platforms themselves do not differ much from other general nodes. You can mark objects (3D objects, joints, cameras, etc.) as a part of a mobile platform in the Robot Construction Dialog (located in the right Control Panel). Generally there is a base chassis object that is the parent of the driving wheels, onboard cameras, etc. in each platform. Thus these parts can be manipulated independently within the user-defined simulation program.
After finishing up with the environment construction, you can switch to simulation mode ("Editor" on the right hand side Control Panel) on the Control Panel. Here you can load, edit and execute your own high-level program (a .NET library) that will control the robot. Of course the data of the onboard sensors (CCD cameras, sonars, tilt sensors, etc.) are at your disposal during the simulation.

3.4. The virtual driving subsystem

The simulator will offer the following kinds of transmission models:

Differential drive with caster
Ackermann steering
Synchronous drive
Tricycle
Omni-directional (a.k.a. Swedish wheel)

The user can manually select and configure the desired driving subsystem in the previously mentioned Robot Construction Dialog. The simulator incorporates the dynamic model of these platforms to behave the same way as a physical platform would do.

3.5. Hardware in the loop testing

Expensive or unique systems are generally hard to test. Hardware-in-the-loop testing lets you build extremely realistic visualization tools by inserting physical hardware in the testing loop.
If we use standard communication interfaces between the implemented modules (Figure 2), then the core framework can treat virtual and physical components as an equal. By using hardware implementation in a given module, we can talk about hardware-in-the-loop testing which is an important application in a virtual prototyping manufacturing environment.
Let's assume the following scenario: we have a proper virtual representation of the real scene where our real robot will be operating (Figure 4).

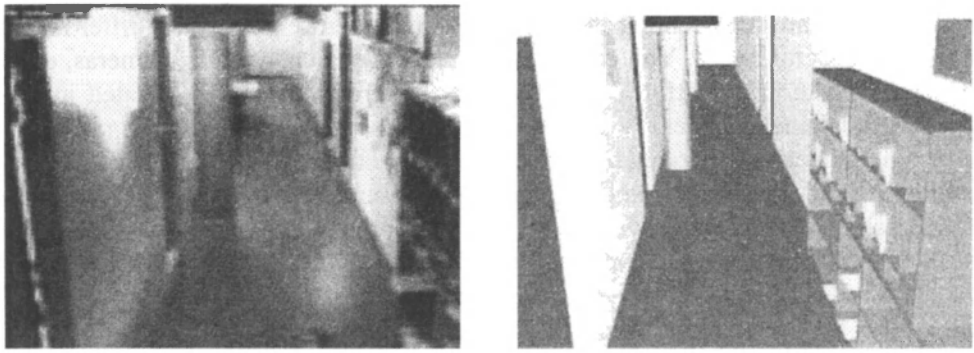

Figure 4: A well reproduced indoor environment

We place this physical robot into an empty room with phantom obstacles (e.g.: sketch drawings on the floor: to avoid any unpredicted collisions) and control it according to the virtual environment's visual information. Thus we can verify our model being used whether the physical platform reacts the same way as the virtual one upon the same command sequence (for example a new navigation algorithm). If we got satisfactory results we can take our platform out of the room and put it to the real situation.

4. CONCLUSIONS AND FUTURE PLANS

Using virtual prototyping and hardware-in-the-loop testing in a modern production system can reduce development costs and time-to-market. To achieve this we have to use realistic, behavioral simulation. The incorporation of the dynamic model of the cooperating mobile platforms is needed to realistic response to virtual forces and torques. Large manufacturing environments ask for distributed simulation that can be accomplished with the conventions of the Extensible Modeling and Simulation Framework [7]. These are the most important tasks for the forthcoming development.

ACKNOWLEDGEMENTS

This research is supported by the Hungarian Scientific Research Fund (OTKA) grant No.: T-042634-OTKA, and the CLAWAR (CLimbing And WAlking Robots) Research Training Network Mobile Robotic Demonstrators and Applications: G1RT-CT-2002-05080.

REFERENCES

[1] CLAWAR COMMUNITY: *Summary for WP2: Simulators*

[2] THAI T., LAM H. Q.: *.NET framework essentials*, 1st ed., 2001, ISBN 0-596-00165-7

[3] JUHASZ, T.: *OpenGL powered mobile robot simulator supporting research on landmark-based positioning*, Proceedings of MicroCAD'03 Conference, 2003, Vol. N., pp. 85-91, ISBN 9-636-61560-8

[4] JUHASZ, T.: *Graphics acceleration techniques for a mobile robot simulator*, Proceedings of CESCG'03: Central European Seminar on Computer Graphics, Section 5: Computer Vision, 22-24th April, 2003, Budmerice, Slovakia

[5] O'REILLY & ASSOCIATES: *Programming C#*, May, 2003 ISBN 0-596-00489-3

[6] WEB3D CONSORTIUM: *X3D overview*, http://www.web3d.org/x3d/overview.html

[7] XMSF HOMEPAGE: *Extensible Modeling and Simulation Framework*, http://www.movesinstitute.org/xmsf/xmsf.html

[8] JUHASZ, T.: *Surveying telerobotics and identification of dynamic model parameters*, Proceedings of MicroCAD'04 Conference, 2004, Vol. K., pp. 211-216 (in Hungarian), ISBN 9-636-61619-1