Virtual user interface for industrial robots off-line programming & simulation and video cam on-line remote operation

POPA STELIAN, NICOLESCU ADRIAN, IVAN MARIO, PASCU NICOLETA
Machines and Manufacturing Systems
University POLITEHNICA of Bucharest
313 Splaiul Independentei, Sector 6, Bucharest, CP 060042
ROMANIA
stelian.popa@gmail.com, afnicolescu@yahoo.com, andrei.mario@yahoo.com, nicoleta_pascu@yahoo.com, www.upb.ro

Abstract: This paper presents the main development steps of a virtual user interface dedicated for industrial robot’s off-line programming and simulation as well as remote on-line operation of a video cam, both of them integrated in the same software platform, as backgrounds of a virtual robotics laboratory developed inside Microsoft Robotics Developer Studio (MRDS). There are successively presented overall hardware – software concept, as well as stages of development for: the virtual prototype of existing robot developed in CATIA V5 solid modeler, the decomposition of IR’s virtual prototype in cinematic modules and their transformation in Blender software for achieving appropriate file types to be imported in MRDS, the redefinition of cinematic modules in MRDS environment as well as final results obtained by implementation of the direct and inverse cinematic algorithms for off-line IR’s programming and simulation. Complementary to previously mentioned facilities, the software module for remote on-line operation of the video cam monitoring the activity of the virtual laboratory is illustrated too.

Key-Words: industrial robotics, remote operation, simulation, virtual laboratory, MRDS

1 Introduction

Virtual laboratories represent today the most efficient and modern modality for sharing hardware resources existing in a single laboratory, training office or research facility. For getting access as well as operation of existing hardware facilities the most important issue is to have on disposal a dedicated user interface able to allow maneuvering but in the meantime preserving the security of the whole set of hardware facilities that are to be operated. From these points of view robotics laboratories have specificity in operation and involve configuration of the specific user interface, a possible approach to solve these problems being the results of the works presented in this paper.

2 Minimum required hardware / software structure of virtual lab

Different ways of developing such software & hardware platform were analysed [1], [2], [3], [4] and finally the following conclusions were revealed [5], [6]:

Hardware needed (Fig.1) to implement virtual laboratory structure must contain at least:
- a PC application server that performs functions associated with laboratory;
- industrial robot(s) and their controller equipped with facilities for interconnection to a PC / server;
- a PC / server for the industrial robot’s controller operation;
- a video camera, preferably IP-camera;
- a video server system;
- a PC on which the client user accesses the structure above.

In terms of necessaries components the software platform should include:
- a web server specific applications;
- an application server for industrial robot(s);
- a database server;
- a video server specific applications;
- a virtual user interface for offline programming and simulation of the industrial robot and IP camera on-line remote operation.
3 Virtual laboratory operation principles

From local machine (which can be in any location: classroom, office, home) the student calls via a web browser the URL of virtual laboratory. The homepage contains general information about the virtual laboratory. The website includes 3 main sections: administration, reserved area for logged in members and the public section. Public section can be accessed without any restrictions while the member and administration areas are protected with usernames and passwords.

Access in member’s area is granted based on username and password. Those credentials used by clients to login are previously generated by the system administrator format physical and the virtual lab. If the client/student does not have this data, he can apply by filling in a registration form which will be validated by the system administrator.

After successful authentication, the student will be able to access the member area that provides access to laboratory resources: course materials, program sequences, images and movies of practical applications of robots in the lab, tutorials, software...
application to operate the virtual laboratory. On the same section users can download the software application in which the student can edit and simulate off-line programs for virtual models of robots / complementary equipment existing in the laboratory as well as access and direct on-line control of a remote operation video camera.

To be able to work within virtual laboratory, the user needs to download and install the off-line operation application. Then he starts the basic operating tasks of programming and off-line simulation in parallel with manual interaction / tutorials specific to it, until reaching the level of skill / competency required to develop a full application for industrial robot(s) Kawasaki FS10 [10] and ABB IRB 140.

After editing and offline programming & simulation of a specific program (for the moment available only for Kawasaki FS10 industrial robot), the user can save his work on the application server. Saved programs in users account are reviewed and validated by the system administrator. If developed correctly, a program is loaded into real robot controller, allocating him a certain amount of time to run on the real system while the user can watch a live steam. Otherwise is informed that he needs to review and correct found errors. Verified and good programs are added to an execution queue on application server and each process has a time slot. Within this time slot, the author can login and view the execution of his program. After logging-in he can see how an execution is performed by the real system. At this stage the program is run without the intervention of the student in developing robot software (robot operation is done only on program previously validated by the system administrator). To view program execution user must connect their PC image acquisition system type camera connected to real lab equipment through dedicated user interface module (module connection / remote camcorder). During program execution student can directly control the camera angles modifying the direction and camera orientation or type commands that zoom-in/zoom-out.

The visualisation is made thru 2 windows: one with images captured from the video camera and the other one with virtual simulation of Kawasaki robot functioning.

Using simulation, users are able to study off-line direct and inverse kinematics of the robot (Kawasaki FS10) and track program execution step by step.

4 Virtual user interface design

Fully operable virtual laboratory has implemented following components [7]:

Web server:
- Linux operating system (CentOS 6.0);
- Apache HTTP server;
- MySQL database server;
- PHP application for platform management (website and administration tasks)

4.1. Offline programming and video cam remote operation applications development

Analyzing the complexity of application, decision to use one of many available frameworks was made and after comparison of advantages and disadvantages of different solutions, Microsoft Robotics Developer Studio (MRDS) [8], [9] was chosen. MRDS is a Windows-based environment for robot control and simulation. It is aimed at academic, hobbyist, and commercial developers and handles a wide variety of robot hardware. MRDS is based on CCR (Concurrency and Coordination Runtime): NET-based concurrent library implementation for managing asynchronous parallel tasks. This technique involves using message-passing and a lightweight services-oriented runtime, DSS (Decentralized Software Services), which allows the orchestration of multiple services to achieve complex behaviour.

Features include: a visual programming tool, Microsoft Visual Programming Language for creating and debugging robot applications, web-based and windows-based interfaces, 3D simulation (including hardware acceleration), easy access to a robot's sensors and actuators. The primary programming language is C#.

MRDS have four main components:
- CCR (Concurrency and Coordination Runtime)
- DSS (Decentralized Software Services)
- VPL (Visual Programming Language)
- VSE (Visual Simulation Environment)

Only CCR and DSS were actively used. The CCR is a programming model for handling multi - threading and inter - task synchronization and DSS is used for building applications based on a loosely coupled service model. Services can run anywhere on the network, so DSS provides a communications infrastructure that enables services to transparently run on different nodes using all of the same CCR constructs that they would use if they were running locally.
In order to allow graphical representation of industrial robots in MRDS, the virtual prototype had to be developed as an *.obj file type. However, it is usually more convenient to use a dedicated 3D solid modeller for developing IR’s virtual prototype. For benefiting about the advantages of using such kind of 3D solid modeller is necessary to have the possibility to export the virtual prototype and as well importing it as an *.obj file type. Because a direct export from CATIA it is not possible, was need to be used an intermediate software for converting prototype files.

To define the virtual prototype of a Kawasaki FS10E industrial robot have been used all technical information included in IR’s data sheet (Fig. 2) [10], as well as some specific dimensional measurements made directly on the IR.

Using CATIA V5 we created first the prototype for the robot (Fig 3a), then decompose it taking account of partially IR’s assemblies as well as specific IR’s major cinematic joint’s. Each partially assembly defined as above has been processed for exporting it from Catia, as *.wrl files extension (Virtual Reality Modeling Language) (Fig 3b). Afterwards *.wrl files were imported in Blender (a free and open-source 3D computer graphics software product used for 3D modeling) and it were transformed in *.obj files, in order to be compatible with MRDS software platform (fig. 4a). After importing them into MRDS each IR’s joint / IR’s degree of freedom, has been defined in MRDS accordingly a specific algorithm, (fig. 4b).

Figure 2. Kawasaki FS 10E robot characteristics [10]

Figure 3. Robot’s overall virtual prototype: a) in Catia V5, b) in Blender [7]

Figure 4. Robot’s partially assemblies virtual prototype: a) *.obj files in Blender corresponding to each partially assembly, b) IR’s joint definition in MRDS [7]

Figure 5. Robot’s off-line programing and simulation. User interface presenting specific robot’s configuration obtained by direct cinematic operation in MRDS [7]
4.2. Industrial robot’s offline programming and video cam remote operation in MRDS

System orientation used in MRDS

To define an articulated entity (two entities joined together) we used a “Joint” object. It has up to 6 degrees of freedom: three angular and three linear and can be represented as a joint frame that follows the right-handed convention. The three axes in a joint frame are called the local axis, the normal axis, and the binormal axis as shown above.

Properties definition of a Joint Object

The joint properties define how the joint behaves and how it attaches to the entities it joins. A joint may contain a reference to a “JointAngularProperties” class to unlock one or more angular degrees of freedom. It may also contain a reference to a “JointLinearProperties” class to unlock one or more linear degrees of freedom. It must always contain a two-element array of “EntityJointConnector” references to specify how it attaches to each entity.

Functions and classes used for writing services that interact with simulation engine extended base classes defined in “RoboticsCommon.dll” and “SimulationCommon.dll”. The most used are listed below:

- “Vector2”, “Vector3”, “Vector4”: are structures that contains number of floating-point values used to represent 2D and 3D objects;
- Quaternion - 3D rotations are represented by the physics engine as quaternions;
- “Pose”: A Pose defines the position and orientation of an entity within the simulation environment. It consists of a “Vector3position” and a “Quaternionorientation”. 
“Entity” is the base type for all entities in the simulation environment and contains all of the information common to both the simulation engine and the physics engine;

“EntityState”: This type contains information about the entity such as its “Pose”, “Velocity”, “Angular Velocity”, and “Name”

“SimulationState” contains information about the “Main Camera” and a list of all the entities in the simulation environment.

“Box Shape Properties” “Capsule Shape Properties” hold state information about each of the shape objects supported by the physics engine.

“BoxShape”, “CapsuleShape” are shapes created from their associated shape properties. They contain a reference to the actual physical shape representation in the AGEIA physics engine.

Simulation service

Building the simulation service implies definition of certain global entities like: a floor, a work scene, a model for the industrial robot. The simulation engine communicates with a view camera.

Conclusions:

This paper presents a synthesis of results obtained in PHD thesis “Research on the development of virtual user interfaces for applications in robotics remote operation” showing the contribution made in hardware-software platform of virtual laboratory for teaching Robotics, Current Developments in Technology-Assisted Education 2006

References: