ISSN:1692-7257 Volumen 1 No 3 - 2004
Revista Colombiana de
Tecnologías de Avanzada

A GENERIC AND MODULAR MODEL TO DEVELOP VIRTUAL LABORATORIES OVER INTERNET

F. VON BORSTEL J. L. GORDILLO B. PONCE

Center for Intelligent Systems
Instituto Tecnológico y de Estudios Superiores de Monterrey
Campus Monterrey, Monterrey, N.L., México
+52 (81) 8328-4379
fborstel@cia.mty.itesm.mx, JLGordillo@itesm.mx, bponce@cia.mty.itesm.mx

ABSTRACT

Until now, Virtual Laboratories have been implemented following empirical considerations and needs. Instead, we propose a modular and generic model to lead the Virtual Laboratories development. A complementary method to transform experimental applications to Virtual Laboratories is also proposed. The method starts with the experiment analysis recognizing inherent procedures to be distributed and encapsulated into model elements. Then a telecommunication platform is selected to interrelate the separated procedures. The complete layout is implemented and evaluated. The method loops into an implementation evaluation cycle for debugging. In this stage improvements and complementary systems can be added. An existing application on mobile robotics served as testbed to validate our approach. The implemented Virtual Laboratory was executed under hard conditions, e.g. allowing long distance interaction in an overloaded network, reporting successful results.
Keywords Computer Vision, Distance Learning, Robotics, Teleoperation, Virtual Laboratory. 1

INTRODUCTION

The Virtual Laboratory concept has been related to distancelearning over Internet. Several papers have shown the importance of span access to sophisticated and expensive equipment to increase students' practical skills, doing preformatted didactical experiments [1-3].MoreoAver, the Virtual Laboratory concept has been applied to research experimentation on robotic teleoperation over 1Internet. Several experimental implementations have been reported in the literature [4-6]. These research applications explored Internet as a communication platform and performed simulated and remote teleoperation. Distance learning and teleoperation create distributed environments to perform distance experimentation.Although multiple implementations have been built, it does not exist a model that defines how specific experiments have to be transformed into Virtual Laboratories. Proposing a generic model is not

straightforward, since the model has to consider the intrinsic procedures of heterogeneous experiments, and distribute them in local and remote sites. The distribution definition becomes a critical step when performance specifications must be maintained. Nowadays, a telecommunication platform model to access Virtual Laboratories can be found in [7], but the experiment procedures were omitted. We propose a generic and modular Virtual Laboratory model. A model-based method to define the optimal 1process distribution on a Virtual Laboratory is also proposed.

This paper describes the Virtual Laboratory types, introduces the proposed model, along with the model-based 1method. An example validates the model and its method, providing experimental performance results and a base for conclusions about the work.

Virtual Laboratory Types

A Virtual Laboratory over Internet always provides access to one or multiple users to operate, in a simulated or remote way, physical equipments to perform, to mount, and to execute experiments. This system provides the user, with capabilities to observe, to supervise, and to interact with an experiment. In general, two Virtual Laboratories types are found:

• *Simulated*: Through a computer connected to Internet, these laboratories give access to a simulated equipment model with its surrounding environment. The simulated equipment operation allows performing tests or complete experiments, giving unlimited access to users and protecting equipment from unskilled users.
• *Remote*: Physical equipment inside a laboratory facility is remotely accessed through a computer connected to Internet. Remote experiment development is possible by this laboratory type. The experiment observation is done using information from sensors connected to Internet and allocated in the equipment's environment. The user has interaction with physical equipment, giving her/him practical skills facing situations that could be difficult to model into a simulation. Unfortunately, these Virtual Laboratories have unpredictable ommunication delays, which affect the overall system performance. Remote Virtual Laboratories are complemented or combined with simulated ones to overtake this disadvantage.

A GENERIC AND MODULAR MODEL

The model represents the abstract description of the processes composing a Virtual Laboratory. The processes distribution was defined by the properties involved in the optimal laboratory performance. The processes as entities can be either i implemented or discarded depending on the application specifications. The model besides being genericis also modular. This modularity characteristic allows the model to be enhanced on any specific implementation. Figure 1 shows the proposed model.

Virtual Laboratory Elements

The following elements are needed to perform experimentation as a Virtual Laboratory over Internet:
• *Guest*: It contains an interface that allows the user to observe, to mount and to execute experiments in a remote or simulated way.
• *Host:* It executes the task required by the user, providing access to software libraries and the equipment, depending on the Virtual Laboratory type (Simulated or Remote).
• *Media:* The computer digital network used to establish the information flow between *Guest* and *Host*. The *Media* in terms of the conceptual layer model OSI (Open System

Interconnection) includes from the low physical communication layer to higher transport communication layer protocols as the TCP/IP (Transport Communication Protocol/Internet Protocol) suite. Therefore, the model must be composed by, at least, the three elements mentioned before; nevertheless, several possible configurations can be envisioned: (a) a single *Host* to single *Guest* connection, (b) a single *Guest* connected to multiple *Hosts*, (c) multiple *Guests* connected to a single *Host.*

Guest Entities

The *Guest* element has several entities to enclose every procedure needed to get a better user performance. The *Input Devices* acquire the commands from the user. These commands are verified and interpreted by procedures enclosed in the *Input Processing* entity, hich interacts with other entities to perform the user requests. In contrast, the *Visual Display* entity encapsulates the procedures that receive information from *Host* and/or *Guest's* simulation procedures to display it via the *Monitor* devices. The model considers in the *Guest* side a *Simulation* entity to delimit the procedures that provide an immediate experiment feedback. In simulated applications, these procedures access the *Host* to get simulated models of the equipment and its environment. In remote applications, the simulation is a tool to foresee the effects of commands applied in the experiment. For enhanced reality feedback, the simulation is used to recreate immediately the physical equipment and its environment state. This format requires updated information from *Host* [8,9].

The *Force and Displacement* entity allows a complete perception of the experiment. These procedures perform the conversion from discrete information into continuous signals to activate electromechanical *Effectors*.

Unfortunately, this kind of feedback implies to solve the intrinsic communication medium problems [10,11]. Finally, keeping data to recreate an experiment performed is a need. The *Task and Settings* entity represents the procedures to establish a database in *Guest* in order to maintain tasks descriptions and configuration settings.

Host Entities The *Host* element is defined by entities that enclose procedures to get an optimized experiment performance.

The *Experiment Control* entity regulates the experiment behavior. This entity encapsulates the control algorithms that belong to a specific application. Knowledge and behaviors can be coded into these algorithms to add an autonomy level to *Host* [12]. The *Experiment Control* procedures work in an open loop mode using the commands sent by the user as reference input, or in a closed loop mode receiving information from *Data Sensed Processing* procedures as a control feedback. The *Data Sensed Processing* entity encloses procedures which receive raw data from *Sensors* devices to convert it into discrete data, and analyze it to get specific information. This information can be sent to *Guest*, if it is required. The *Sensor* devices are allocated in the *Remote Environment* acquiring data from the experiment. The experiments' data storing procedures are enclosed by the *Experiment Database* entity.

The *Controlling Devices* are considered as electromechanical devices that include a processing unit to communicate via a discrete interface. The *Commands Conversion* entity

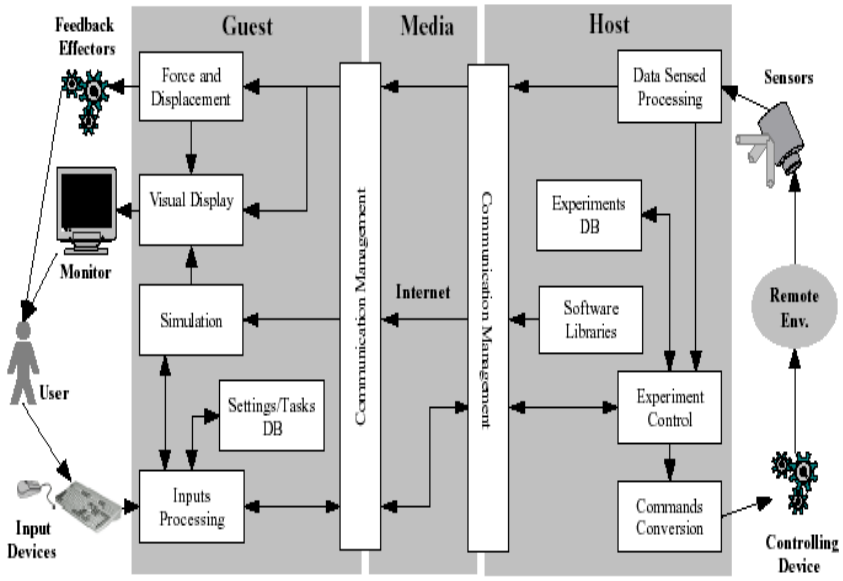

Fig. 1 Generic and modular model to lead Virtual Laboratories implementation, several abstracts entities compose every essential element, every entity can be discarded or implement depending of the experiment application specification.

represents the procedures for converting *Experiment Control* output to the required *Controlling Devices* input e.g. device communication protocol. In simulated experimentation, the *Guest* accesses to simulated models allocated in the *Host*, these models are represented by the *Host's Software Libraries* entity.

Media entities

The *Communication Management* entities include procedures for both managing and sharing information between entities through the *Media*, using one or several technologies to perform it.The *Guest* implementation can be considered as: *Browser Application* or *Stand-alone Program*. *Browser Applications* are embedded or called in a Web page, the browser and Web server manage the communication at first instance. These applications use three ways to perform the communication flow: using the Web server CGI (Common Gateway Interface) to send commands to the *Host* system; using browser's virtual machine to call programs e.g. Applets; using Plug-Ins to access virtual worlds e.g. VRML (Virtual Reality Model Language). *Stand-alone Programs* are executed on the *Guest* computer operating system. These applications manage their communications via Sockets, RMI (Remote Method Invocation), or using different technologies as middleware to overtake the involved management implementation problems when the *Host* procedures are distributed in several locations and/or are going to be accessed by multiple *Guests* e.g. CORBA (Common Object Request Broker Architecture).

Model-Based Method

The first step in the proposed method analyzes the experiment to provide, as output, the main

tasks performed, the intrinsic procedures needed to accomplish these tasks and their specific interrelationships. In the second step, the procedures are related and encapsulated into the model abstract entities. Procedures that have strong interaction with the user are related to *Guest*. Procedure that have strong interaction with the equipment and its environment are related to *Host*. This step also identifies possible weak interactions to establish the experiment break point. This is done reducing the data amount passing through, and the iteration frequency. The third step is choosing the adequate technology to establish the communication flow to interrelate, again, the procedures separated by the *Media*. This step provides a specific schema to lead in the system development. The fourth step is the *Guest* and *Host* implementation leaded by the schema. The fifth step is an evaluation which checks if the tasks and its specifications have been met. In this step, user interaction, security, autonomy, capacities, and performance lacks are described, performing a backtracking to the fourth step in order to improve the laboratory implementation. Finally, once the specifications have been met, a final implementation is generated. Figure 2 describes the method steps.

EXPERIMENT TRANSFORMATION

The proposed model and its method were validated by transforming an experimental application on mobile robotics [13]. This application was built as a didactic and research exercise based on mobile robot path planning and path following using computer vision techniques. The application controls a Khepera robot [14]. The experiment environment is set when the user indicates the physical obstacles vertex on a robot workspace top view image. These vertexes are used by the application to establish a *C-Space* according to the robot characteristics and its current environment. Once the *C-Space* is completed, the user indicates a point to be reached by the robot. The system draws an obstacle-avoiding trajectory on the image. If the user accepts it, the application makes robot to follow that path.

Experiment Analysis

The path planning is done using a potential field method, while the path following uses the planned path as reference and a visual feedback to control the robot movements.

Figure 3 describes the experiment chart flow. First, the image is acquired by a camera as compound video, and converted into discrete data using a video capture board. This process generates a matrix (*I*), with 320 x 240 integer elements on a 256 gray scale. This matrix is binarized and segmented to find every object) (*i O* in the image. After that, the objects are characterized using Hu invariant moments to identify a robot artificial mark. Two circles compose robot's mark, one circle with bigger area than the other; the bigger circle represents robot's front while the small one the robot's end. The robot position) , (*k k y x* is the big circle centroid position, and the robot orientation) is the angle defined by a line passing on both circles centroids and the *x*-axis. Meanwhile, the image is displayed to the user and then s/he interacts with the user interface to define a *C-Space* based on the workspace image. This definition is done using a PC mouse, the user points on the image the obstacles vertex) , (*v v y x* , and the system computes a vertex convex hull, drawing a convex polygon) (*i H* , which is expanded a half of robot dia meter anddisplayed again on the image to represent a *C-Obstacl)*

This procedure is carried out *m* times, once per each physical obstacle. If the user is satisfied with this *C-Space*, then the system converts the displayed Bitmap (*P*) to a matrix (*C*) with the same size than *I*, where the pixels inside every convex polygon's area are represented by elements with great decimal values e.g. 10,000. This *(C)* matrix models a three-dimensional *C-Space* grid, which is used in the path planning. Once the user indicates the destination point) , (*f f y x* on the image, the path planning process starts.

This process is based on the wavefront expansion NF1 method, see Figure 4, then a best-first path search using a minimal Manhattan distance heuristic is performed.

A cubic spline interpolation procedure is used to smooth the continuous planned path)} , {(*y x* , which is analyzed to get the slope change points to keep them in an array of *n* sequential points .

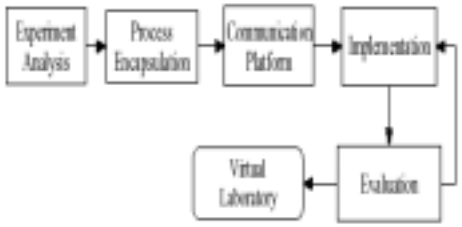

Fig. 2 Describes the proposed method to transform experimental applications to Virtual Laboratories.

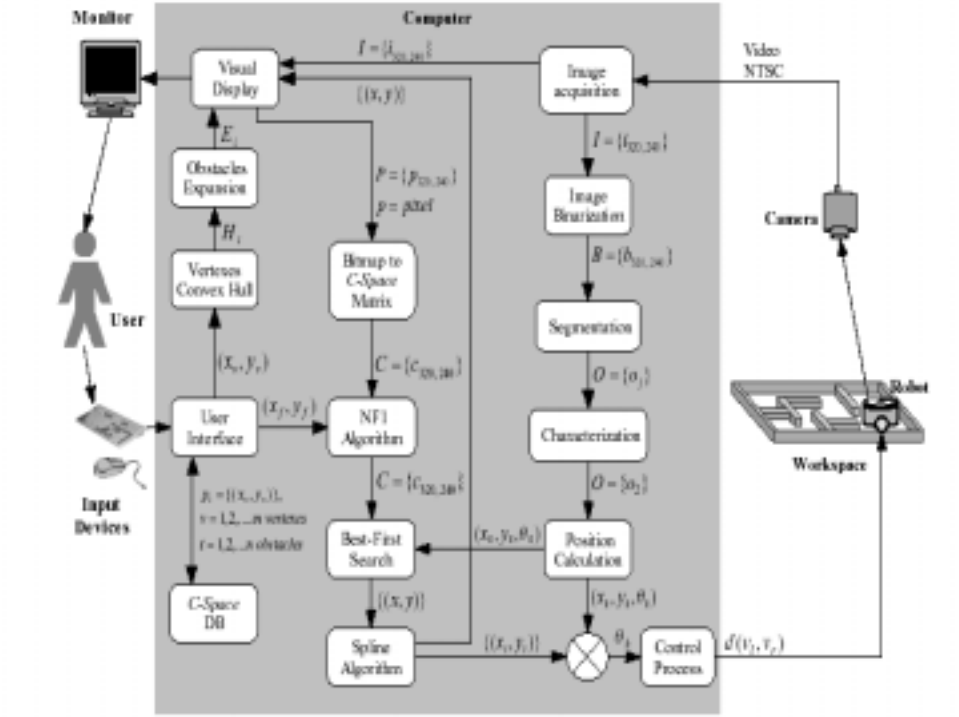

Fig. 3 Describes the experimental process chart flow. The computer-based experiment performs robot path planning and path following using a potential field planning method, and computer vision techniques to feedback the follower control.

The follower procedure uses this array as reference. The path following is done comparing each current robot position with the next path point to reach. an advance command is sent to the robot if a specified threshold is smaller than the angles difference between and the angle defined by a line passing on and, and the *x*-axis. An orientation command is sent if the threshold is bigger than

Processes Encapsulation

In this step, we relate and encapsulate the experiment inherent processes into the model entities. Processes that establish interactions with the user define the *Guest* side. In this case, the user has strong interaction with the planning processes. Four entities enclose the planning procedures. First, the *Input processing* entity encapsulates the processes to establish the user interface behavior. The *Visual Display* entity encloses the processes to display the workspace images, and the *C-Space* Bitmap creation. The *Simulation* entity delimits the *C-Space* Bitmap conversion process to generate a discrete potential field representation, and a path creation process based on a potential field gradient descent, combined with an interpolation algorithm to smooth the path. Though, it has not been mentioned before, there exist a process to save and load *C-Space* definitions on a storage device. This process is related to the *Task and Settings Database* entity. Processes that interact with the experiment and its environment define the *Host* side. The path following procedures receive processed information from a visual *Sensor* to regulate the *Controlling Device* behavior, in order to perform the experiment goal using the planned

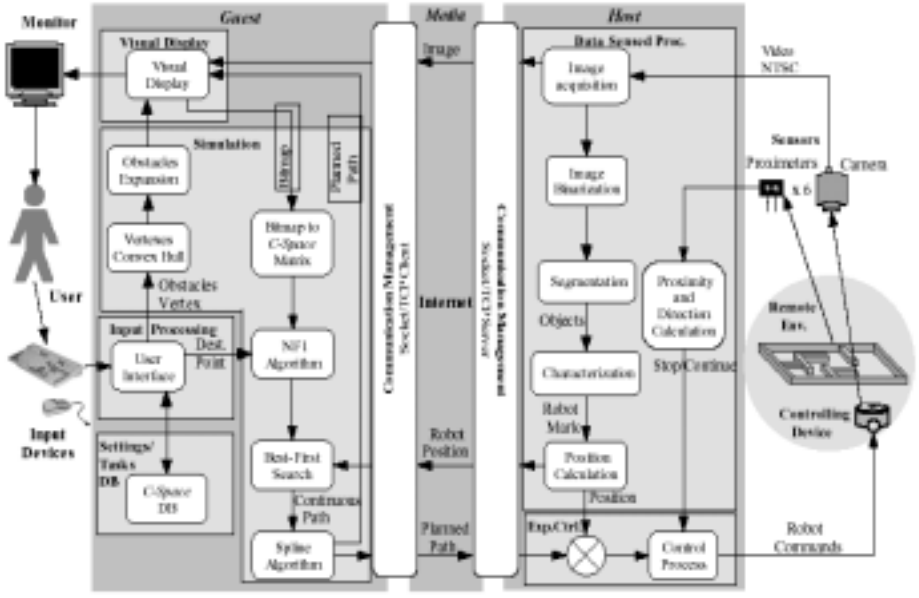

Fig. 5 Final abstract template based on the model to implement a path planning and following Remote Virtual Laboratory. The Virtual Laboratory is complemented by the observation system to get a better experiment visual feedback. Some links are not presented to keep the schema simplicity.

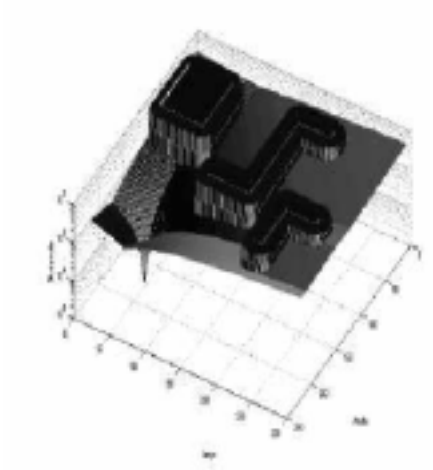

Fig. 4 Potential field matrix created by the NF1 procedure, the graphic uses a logarithm scale to show the cone shape on the finish point.

path as reference. These procedures are related to two entities. The *Data Sensed Processing* entity delimits the image information extracting procedures. The *Experiment Control* entity encapsulates the path and robot position comparison processes for controlling the robot movements.Three possible weak links are identified among the *Guest* and *Host* systems:

- The images sent by the image acquisition process to the visual display procedure. This is a strong link due to the amount of data per image (153.6 Kbytes) to reduce it, a less number of images should be send.
- The robot position sent by the position calculation process to the path search procedure is a weak link, since the robot position has 6 bytes of data and it is required once when the path search starts.
- The planned path sent by the spline algorithm to the robot control process is a weak link, because the planned path usually takes less than 70 bytes and it is sent once each following task.

Choosing a Communication Platform

The original implementation was made as a Java application; we decided to implement *Guest* communication as *stand-alone program*, choosing Sockets based on TCP (Transport Control Protocol). The *Guest's Communication Management* entity will behave as client requiring data and sending commands to *Host*. The *Host's communication Management* will behave as a server receiving and filling the *Guest* requirements. These last definitions complete the processes distribution on the proposed model. Until now, the connection will be one to one using a scheduling strategy to perform the experiments.

Virtual Laboratory Implementation

The *Guest* and *Host* implementations were done separating the experiment processes based on the generated schema. This model-based configuration gives the optimal process arrangement for a good laboratory performance. The separated processes that need to be interr elated between the path planning and the path following procedures are reconnected using the defined communication platform. The communication management was based on a synchronized communication flow. To avoid great delay times by continuous image requests by *Guest*, when it starts performs 10 image requests and calculates their average time delay. The *Guest* divides the average image delay by the average *Host* execution time (131.145ms). This action gives the *n Host* cycles that can be performed in an image delay lapse. The *Guest* starting its eleventh cycle sends a 2-byte length command. The *Host* reads this command and performs the path following task, without sending any response to *Guest*. The robot movements are held a specific time frame to avoid the instability introduced by the unpredictable communication delays. This is

done *n*-1 additional times more, then the *Guest* sends an image request, and the 2-byte command cycle starts again.

System Evaluation

Based on experimentation, we get the system lacks: poor user interaction with the experiment, and reduced experiment observation capacities. We decided to make the system more interactive within the user and the experiment. On the other hand, the reduced number of images displayed in the *Guest*, decrease the user ability to follow the experiment development. Then, we decided to b uild a complementary observation system.

Laboratory Improvements

Several system modifications were done based on the evaluation step. The six infrared proximeters that are installed on robot's front were used. These sensors increased the system autonomy and security, just adding 1one process in the *Data Sensed Processing* entity. These sensors allow a dynamic workspace, since these sensors can detect and avoid collisions with obstacles which were not described in the defined *C-Space*.

The simulation and the path following procedures were also modified. These modifications allow: inserting virtual obstacles on the *C-Space*, avoiding "collisions" with them, performing path modifications, getting robot remote control, and changing experiment settings. These improvements added user interaction with the experiment development. Figure 5 shows the final schema used to implement the remote Virtual Laboratory.

Finally, the complementary system was set on an independent application. This system was developed using the model and its method, and it is able to observe the entire experiment scenario with more detail and frequency.

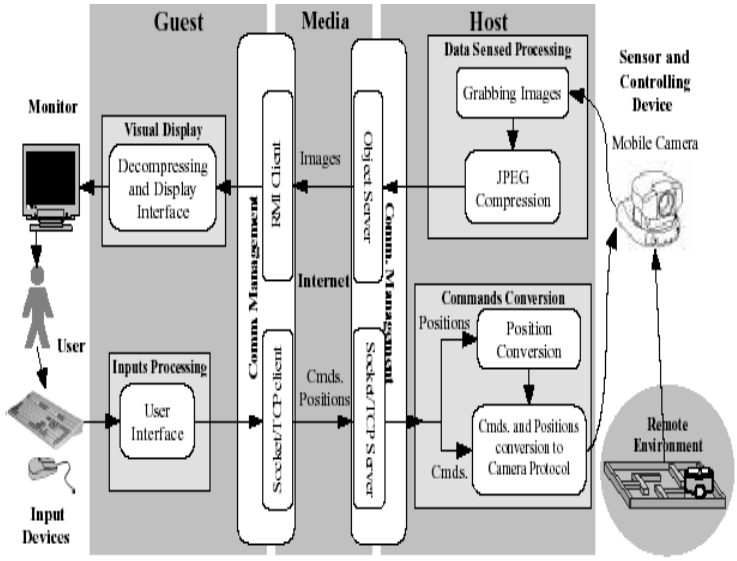

Fig. 6 Complementary observation system abstract template, this system acts as observer with two communication management formats implemented.

Observer System

This complementary system controls a second mobilecamera as a *Controlling Device* in an open control loop mode. This camera is able to make pan, tilt and zoom movements. The system tasks are: to observe and to follow the experiment development. The observation task is performed processing the acquired images. These images are also compressed in a JPEG (Joint Photo graphics Expert Group) format. The image processing done is related to the *Data Sensed Processing* entity. The compressed images are sent to *Guest* via a RMI object server procedure which isenclosed in the *host's Communication Management* entity. The *Guest* receives these images through a RMI client procedure encapsulated by the *Guest's Communication Management* entity. The images received are decompressed and displayed into a visual interface. The processes which integrate this visual interface are encapsulated in the *Visual Display* entity. Meanwhile, the commands and absolute positions needed to control the camera movements are introduced via *Input Devices* to a user interface. The user interface process is encapsulated in the *Input Process* entity.

The commands and absolute positions are sent to *Host* via a client Socket/TCP procedure, and are received by a server Socket/TCP procedure. These procedures are encapsulated in both *Communication Management* entities. The absolute positions are transformed from cartesian to polar coordinates. Commands and positions are converted to the camera serial protocol. These procedures are system was implemented in Java language. The performed tests indicated that there is a trade off betwe en image quality and the image data amount sent (less than 8Kbytes), if we compress the image with a JPEG compression quality of 50.

CONCLUSIONS

In this paper, we present a modular and generic model to develop Virtual Laboratories from specific ex periments. A complete model abstract entities description, and a transformation method were presented. The model and its method were validated using them to transform a mobile robotic experiment application. The experiment application was complemented to improve user observation capacities. The experimental results of the implemented Virtual Laboratory were also presented. The path analysis showed the system accuracy to follow remotely planned paths. The time analysis validates the processes distribution. Frequent iterations of the following task procedures are needed to get an optimal experiment performance.

REFERENCES

1. M. Joler, and C. G. Christodoulou. Virtual Laboratory, Instruments and Simulations Remotely Controlled via the Internet, *Antennas and Propagation Society* 2001, IEEE, Intl. Symposium Vol.1 2001, pp. 388-391

2. B. Wagner. From Computer-Based Teaching to Virtual Laboratories in Automatic Control. 29thASEE/IEEE *Frontiers in Education Conference* 1999, November 10- 13, San Juan Puerto, Rico, session 13d6.

3. C. Ko, B. M. Chen, S. H. Chen, V. Ramakrishnan, R. Chen, S. Y. Hu,, and Y. Zhuang. A Large-Scale Web- Based Virtual Oscilloscope Laboratory Experiment. *Eng. Science and Education Journal*, April 2000.

4. H. Hirukawa and I. Hara. Web-Top Robotics, Using World Wide Web as a Platform for Building Robotic Systems. IEEE *Robotics & Automation Magazine*, June 2000.
5. T. Hori, H. Hirukawa, T. Suehiro and S. Hirai.
Networked Robots as Distributed Objects. Proc. 1999 IEEE/ASME, *Intl. Conf. on Advanced Intelligent Mechatronics*. Sept. 19-23 1999, Atlanta, USA.

6. H. Hirukawa, T. Matsui, S. Hirai. A Prototype of Standard Teleoperation Sytems on an Enhanced VRML.Proc. IROS/IEEE '97, Vol. 1, pp. 1801-1806.

7. M. Kassouf, S. Pierre, C. Levert, and J. Conan. Modeling a Telecommunication Platform for Remote Access to Virtual Laboratories. Proc. IEEE *Canadian Conf. on Electrical and Computer Eng.*, pp. 127-132, Alberta, Canada May 9-12, 1999.

8. H. Friz, P. Elzer, B. Dalton, and K. Taylor. Augmented Reality In Internet Telerobotics Using Multiple Monoscopic Views. IEEE Intl. *Conf. on Systems, Man, and Cybernetics* 1998,Vol. 1, pp. 354-359, 1998.

9. D. Shulz, W. Burgard, and A. B. Cremers. Robust Visualization of Navigation Experiments with Mobile Robots over the Internet. Proc. 1999 IEEE/RSJ Intl. *Conf. on Intelligent Robots and Systems*, pp. 942-947.

10. J. Kikuchi, K. Takeo, and K. Kosuge. Teleoperation System via Computer Network for Dynamic
Environment. Proc. 1998 IEEE, *Intl. Conf. on robotics & Automation*, Leuven, Belgium, May 1998. 1