Mobile Robotics Virtual Laboratory Over the Internet

Fernando D. Von Borstel, Brian A. Ponce and José L. Gordillo
Center for Intelligent Systems, ITESM Campus Monterrey, Monterrey, N.L., México.
fborstel@cia.mty.itesm.mx, bponce@cia.mty.itesm.mx, jlgordillo@itesm.mx

Abstract

In this paper we describe a mobile robotics virtual laboratory, that allows the user to define and perform remote experiments over the Internet using a mobile minirobot. The user interacts with a user interface, which receives a workspace top view from a remote laboratory facility. She defines and draws a C-Space on the image, and provides a destination point. Once C-Space and destination are defined, a potential field based planner draws a path. This path is sent to the laboratory facility and a vision-based computer system makes the robot follow that path. Several capabilities to enhance real-time user's interaction are provided: grab and drop path modification, virtual obstacles insertion, changeable experiment settings, and a complementary observer system used to get a detailed view of the robot and its workspace.

1. Introduction

The *Virtual Laboratory* (VL) concept has been related to distance learning over the Internet. Several studies have shown the importance of increasing access to sophisticated and expensive equipment to improve students' practical skills, by doing preformatted didactical experiments [1-3].

Moreover, the VL concept has been applied to robotics research over the Internet. Several experimental implementations have been reported in the literature [4-6]. These research applications explored the Internet as a communications platform and performed simulated and remote experiments.

Distance learning and research applications enable remote human collaboration and create distributed environments to perform experimentation.

This paper describes the current development status of a VL implementation to give remote access via the Internet, to a mobile robotics learning and research laboratory. A robotic navigation application was modified to be accessed over the Internet, and was improved to offer several capabilities to enhance real-time user interaction.

The article is organized as follows: Section 2 gives a detailed description of the modified application and its improvements. Section 3 presents performance results under stress conditions, e.g. allowing long distance interaction in a multiple jump network route. Section 4 provides conclusions and further work.

1.1. Virtual Laboratory Architecture

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

The Simulated VL, give access to a simulated equipment model with its surrounding environment. Remote VL give access to physical equipment inside a laboratory facility. Experiment observation is done using information from sensors connected to the Internet .

We defined a VL architecture based on two elements. The first element which we called *Guest* is the computer containing user interface(s) that allows the user to observe, mount and execute experiments in a remote or simulated way. In contrast, we called *Host* to the computer containing an application which executes the task requested by the user, providing access to software libraries and/or physical equipment, depending on the VL type (Simulated or Remote).

This architecture must be composed of at least, a single *Guest* connected to a single *Host*; nevertheless, other possible configurations can be envisioned: a single *Guest* connected to multiple *Hosts*, multiple *Guests* connected to a single *Host.*

We describe a Remote VL which is composed by a *Guest* connected to two *Hosts* controlling a mobile robot and a mobile camera.

Proceedings of the Fourth Mexican International Conference on Computer Science (ENC'03)
0-7695-1915-6/03 $17.00 © 2003 IEEE

2. Mobile Robotics Experiment

The experiment was based on a mobile robotics application [7] using robotics navigation and computer vision techniques to control a Khepera minirobot [8].

The experiment environment is set when the user indicates the physical obstacle vertices on a robot workspace top view image. These vertices are used by the system 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 user accepts, the application makes robot follow that path.

2.1. Experiment Analysis

The path planner uses a potential field method, while the path follower uses planned path as reference and a visual feedback to control the robot movements.

Figure 1 describes the experiment chart flow. First, the image is acquired by a camera as compound video, and converted into discrete data using a video grabber board. This process generates a matrix (*I*), with 320 x 240 integer elements on a 256 level gray scale. This *I* matrix is sent to *Guest* to be displayed on user interface.

Meanwhile, in *Host* the *I* matrix is binarized and segmented to find every object (O_i) in the image. After that, the objects are characterized using Hu invariant moments to identify a robot artificial mark. Two circles compose the robot's mark, one circle bigger than the other and representing robot's front while the small one indicates the rear. The robot position (x_k, y_k) is the big circle centroid, and the robot orientation (θ_k) is the angle defined by a line passing through both circle centroids and the *x*-axis.

On the other hand, the image is displayed on user interface which the user interacts to define a *C-Space* based on the workspace image. This definition is done using a PC mouse; the user marks on the image the obstacle's vertices (x_v, y_v), and the system computes a vertex convex hull, drawing a convex polygon (H_i), which is expanded a half of robot diameter and displayed again on the image to represent a *C-Obstacle* (E_i). 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 high 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 (x_f, y_f) on the image, the path

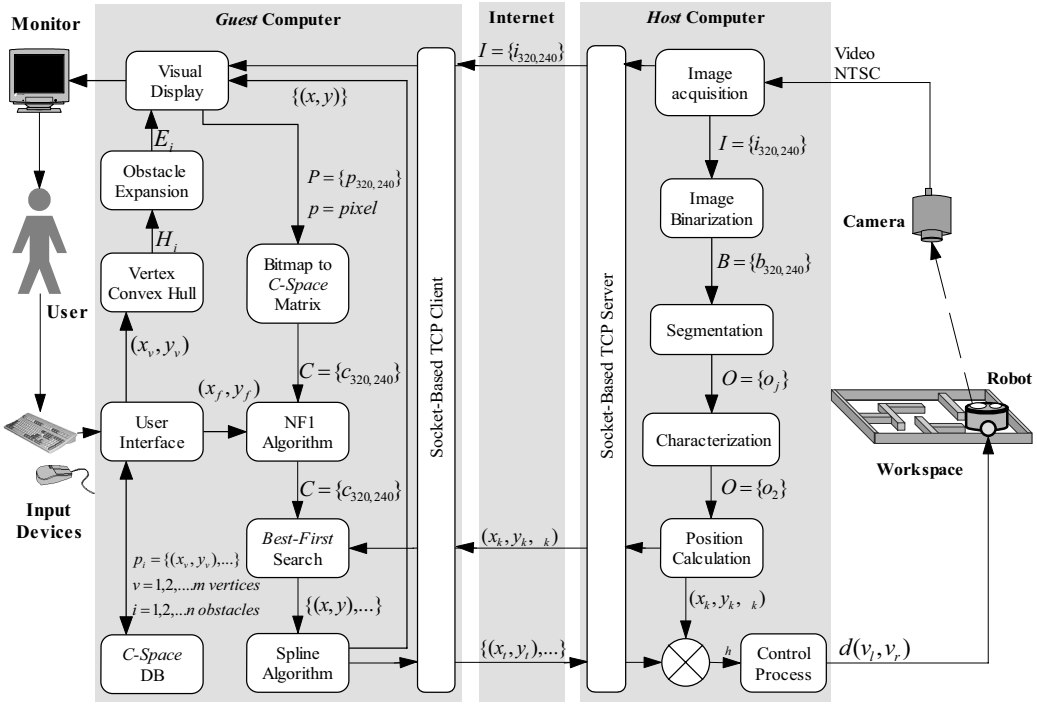

Figure 1. Describes the experiment process chart flow. The experiment performs robot path planning and following, using a potential field planning method and computer vision techniques.

planning process starts. This process is based on the wavefront expansion NF1 method, --see Figure 2-- 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 $\{(x,y),...\}$, which is analyzed to get the slope change points to keep them in an array of *n* sequential points $\{(x_t,y_t),...\}$.

In *Host*, the follower procedure uses this array as reference. The path following is done comparing each current robot position (x_k,y_k,θ_k) with the next path point to reach (x_t,y_t) e.g. an advance command $d(+v_l,+v_r)$ is sent to the robot if a specified threshold (δ) is smaller than the angular difference (θ_h) between θ_k and the angle defined by a line passing through (x_k,y_k) and (x_t,y_t), and the *x*-axis. An orientation command $d(\pm v_l,\ v_r)$ is sent if the threshold is greater than θ_h.

2.2. Telecommunications Platform

The implementation was written in Java, and the telecommunications platform chosen was socket-based TCP (Transport Control Protocol). The *Guest* behaves as client requesting data and sending commands to *Host*. The *Host* behaves as a server receiving and filling *Guest's* requirements. Until now, the connection is one to one using a scheduling strategy to perform the experiments.

The uncompressed images sent by *Host* to *Guest* are 76.8 Kbytes long. The robot position sent by the robot position calculation process to the search procedure has 6 bytes of data. The planned paths sent to *Host* usually take less than 320 bytes and are required once per path.

The communication flow was based on a synchronized protocol. 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 *n Host* cycles that can be performed in an image delay interval. 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*. Robot movements are held a specific time frame to avoid 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.

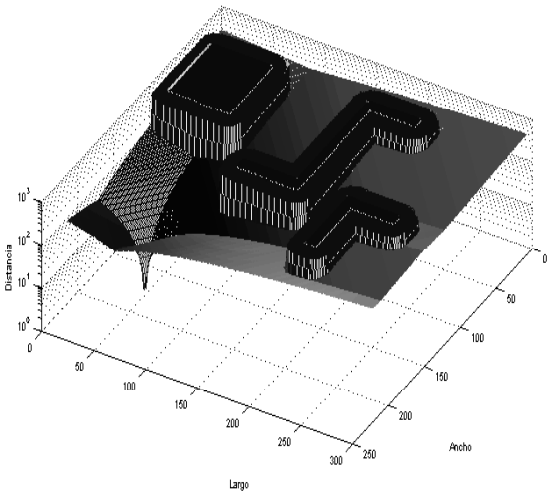

Figure 2. Potential field matrix created by the NF1 procedure, the graphic uses a logarithmic scale to show the cone shape on the finish point.

2.3. Laboratory Improvements

Based on experimentation, we got system deficiencies: poor user interaction with the experiment, and reduced experiment observation capabilities.

We decided to make the system more interactive with the user. Several modifications were done to the implementation: The six infrared proximeters installed on robot's front were used. These sensors increased system autonomy and security, just adding one process in *Host*. These sensors allowed a dynamic workspace, since they can detect obstacles not defined in the *C-Space*. The user interface was also modified, allowing insert pre-defined virtual obstacles on the workspace image and in the discrete *C-Space*. To avoid "collisions" with the virtual obstacles, updated robot positions were sent to *Guest* to compare them with the discrete *C-Space.* If a "collision" is detected on the planned path the *Guest* warns the user and before robot "collides", it sends a stop command to *Host*. Then, the user can perform path modifications in a grab and drop mode. Furthermore, the user can get robot remote control, and change experiment and sensor settings. These improvements added user interaction with the experiment development.

On the other hand, the reduced number of images displayed in the *Guest*, decreased the user ability to follow the experiment development. Therefore, we decided to build a complementary observation system. The observer system was developed using the same

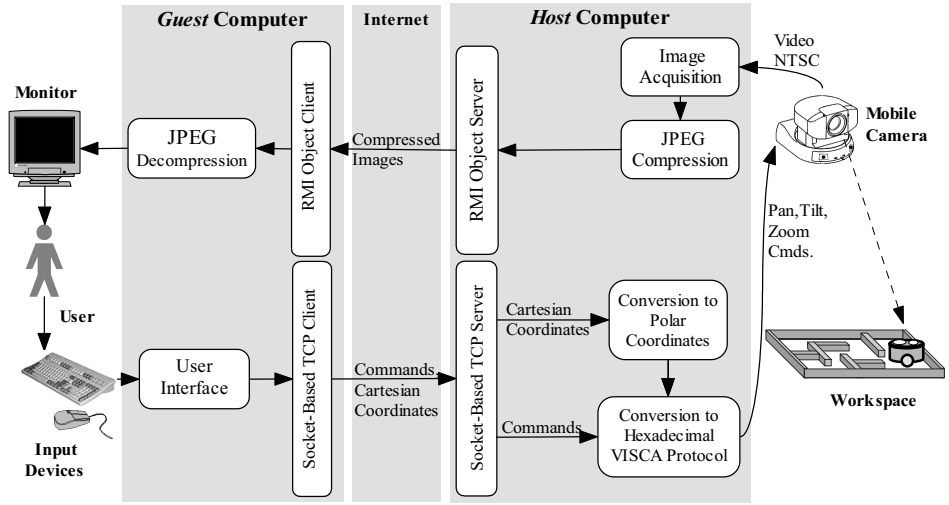

Figure 3. The complementary observer system uses RMI and Socket/TCP telecommunication procedures.

architecture, and is able to observe the entire experiment scenario with more detail and frequency. The observer user interfaces were implemented in Java as applet applications.

2.4. Observer System

This complementary system controls a second mobile camera in open control loop mode. Figure 3 describes the complete observer system. The camera is a Sony EVI-D30 capable of making pan tilt and zoom movements. Images acquired by a video grabber board are compressed in a JPEG (Joint Photo graphics Expert Group) format, using methods written in C and converted to JNI (Java Native Interfaces). Compressed images are sent to *Guest* via the Internet by means of a Java RMI (Remote Method Invocation) object server. The *Guest* receives these images using a RMI client procedure. This images received are decompressed and displayed by a video interface.

On the other hand, commands and Cartesian coordinates needed to control camera movements are introduced to another user interface. Commands and coordinates are sent to *Host* via a socket-based TCP client procedure, and received by a listening socket-based TCP server procedure. Cartesian coordinates are transformed to polar coordinates. Both commands and positions are converted to the camera serial protocol.

The performed tests indicated that there is a trade off between the JPEG image compression quality and the amount of data sent. We got less than 8Kbytes, when we compressed the image with a JPEG compression quality of 50%.

3. Experimental Results

Remote experiments were done using the path planner and follower laboratory in conjunction with the observer system. All user interfaces were run in a *Guest* computer, and the server applications were executed in two *Host* computers: one computer was the path follower *Host*, and the other was the observer *Host*. The *Guest* computer was geographically separated from *Host c*omputers. The results were acquired using a route with 23 jumps, based on the *traceroute* analysis. This route had variable bandwidth and large delay times. Figure 4 describes the experimental configuration.

Computers used to implement the Remote VL have the following characteristics: the *Guest* Computer is a 1.1 GHz. Pentium III Compaq laptop with 128 MB in RAM, and MS Windows 2000. The path follower *Host* is a 333 MHz Ultra 10 Sun workstation, with 320 MB in RAM and Sun OS 5.7. The observer *Host* is a 270 MHz Ultra 5 Sun workstation, with 256 MB in RAM and Sun OS 5.7. Both *Host* computers are equipped with Sunvideo Plus video grabber boards.

In the follower *Host*, the robot serial communication was set at 9600 bauds. Robot commands were speed commands, which set each wheel motor speed through a PWM (Pulse Width Modulation) control. Advance commands were done using +13 pulses/10 ms settings.

The orientation commands were done using ± 4 pulses/10 ms settings. These settings gave an average forward speed of 2.3383 cm/s on straight paths. The threshold δ was set to 7 degrees in the follower system.

Figure 5 shows all *Guest* interfaces, the path planner and follower *Guest* interface has drawn the *C-Space* used in the experiments.

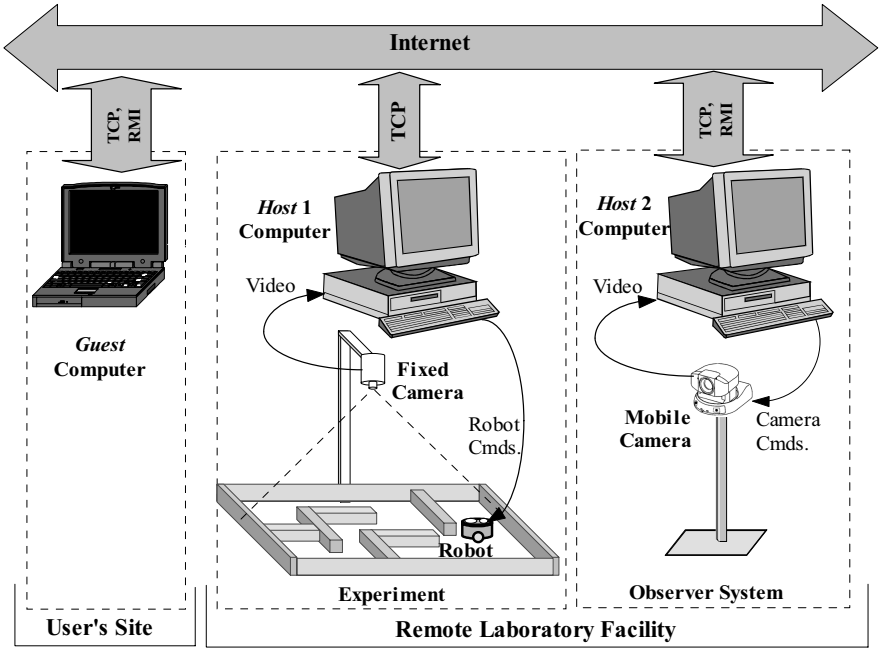

Figure 4. Virtual Laboratory configuration to perform remote experiments. A single *Guest* computer was connected to two *Host* computers to get a better interaction with the experiment development.

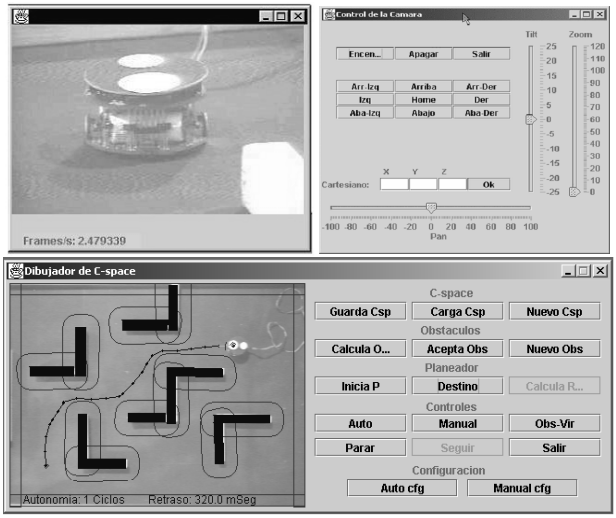

Figure 5. Shows *Guest* interfaces: visual and control observer interfaces on top, and the path planner and follower interface at the bottom.

3.1. Path Analysis

We did several path planning and following tasks from *Guest* to analyze errors in the follower system. Results were calculated based on a workspace of 73.5 cm width and 98 cm length, this workspace was captured in a 320 x 240 pixels image, which gave a 0.30625 cm/pixel resolution.

Table 1. Path analysis results

Path	Control points	Total distance (cm)	Quadratic error sum (cm^2)	Avg. error (cm)
1	14	49.160	18.536	0.219
2	15	44.599	20.011	0.243
3	22	66.092	47.208	0.300
4	10	30.588	9.670	0.195
5	11	25.492	5.009	0.060
6	14	47.063	12.631	0.191
7	30	97.360	56.955	0.299

Table 1. Shows the path analysis results on each path following tasks.

Table 1 shows the path analysis results for each path done. Path control points give the amount of points to define the trajectories. Path distances are the sum of distances between control points. Quadratic error sum and

average error calculation are calculated based on the perpendicular distance between sensed points and its belonging path segment. Quadratic errors represent the area difference between both planned path and followed path. Errors depend basically on the path complexity, and they were proportional to how many curves exist in the paths, and how many control points were used to describe each curve. Straight paths can be followed with minimal errors, even if few control points describe them.

3.2. Time Analysis

The acquired data time analysis gave the results shown in Table 2, e.g. in path 1, 132.994 seconds were needed to accomplish it, an image was sent from the follower *Host* after 37 2-byte commands received, in total 12 images were sent to *Guest*, and the follower procedure was executed 478 times. Each image adds an average delay time of 4.744 seconds.

Table 2. Time analysis results

Path	Total Time (s)	Images sent	2-byte cmds. sent	*Host* cycles	Avg. delay per image (s)
1	132.994	12	37	478	4.744
2	402.734	25	37	951	9.210
3	603.861	50	37	1899	5.864
4	94.842	12	26	322	3.752
5	68.638	10	26	252	3.185
6	130.988	18	26	494	3.068
7	386.310	56	27	1439	2.927

Table 2. Shows the time analysis results acquired using a multiple jump network route, with variable bandwidth and delays.

The average time delay added per image shows longer and unpredictable communication time delays. These delays impact the following task total time, and the number of images sent to *Guest*. The follower system iteration frequency in each path is represented by the *Host* cycles showed in Table 2.

4. Conclusions and Further Work

In this paper, we presented a Remote Virtual Laboratory implementation. The VL allows the user develop experimentation on mobile robotics over the Internet.

The robotic navigation experiment and its improvements to get real-time user interaction were described in detail. A complementary system to improve user observation capabilities was also described.

Experimental results of the implemented VL were presented. The path analysis showed the system accuracy to follow remotely planned paths. The time analysis showed unpredictable delays in the Internet connection.

The implementation can be improved using multithreading and task queues to provide access to several users. Currently, we are developing another complementary system to define, mount and execute experiments over the Internet on several mobile robotics issues, e.g. sensor-based reactive behaviors, robot teleprogramming, and robot navigation using haptic force feedback interaction.

5. 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] O. Michel, P. Saucy and F. Mondada. KhepOnTheWeb: An Experimental Demonstrator in Telerobotics and Virtual Reality. IEEE Robotics and Automation Magazine, Vol. 7, Issue 1, pp. 41-47, March 2000.

[6] H. Gonzalez-Baños, J. L. Gordillo, D. Lin, J. C. Latombe, A. Sarmiento and C. Tomasi. The Autonomous Observer: A tool for Remote Experimentation in Robotics, 1999 SPIE *Int. Symp. on Intelligent Systems and Advanced Manufacturing*, Boston, U.S.A., SPIE Proc. 3840, November 1999.

[7] B. Ponce. Planeación y Seguimiento Visual de Trayectorias para Guiar un Vehículo Autónomo, Master's Thesis, Instituto Tecnológico de Estudios Superiores de Monterrey, Monterrey NL, México, 2002.

[8] F. Mondada, E. Franzi, and P. Ienne. Mobile Robot Miniaturization: A Tool for Investigation in Control Algorithms.

Proceedings of the Third Intl. Symposium on Experimental Robotics, Kyoto, Japan, 1993.

Proceedings of the Fourth Mexican International Conference on Computer Science (ENC'03)
0-7695-1915-6/03 $17.00 © 2003 IEEE