E-Learning: Case Studies in Web-Controlled Devices and Remote Manipulation

TAREK SOBH, RAUL MIHALI, PUNEET BATRA, AMIT SINGH, SUDIP PATHAK, TOMAS VITULSKIS, ANDREW ROSCA

School of Engineering and Design, 169 University Avenue, Bridgeport, CT 06601, U.S.A.
Phone: (203) 576-4116, Fax: (203)576-4766
http://www.bridgeport.edu/~sobh

Abstract - Chances are that distance learning will transparently extend colleges and institutes of education and could plausibly overtake and turn into a preferred choice of higher education, especially for adult and working students. The main idea in e-learning is to build adequate solutions that can assure educational training over the Internet, without requiring a personal presence at the degree offering institution.
The advantages are immediate and of unique importance, to enumerate a few: Education costs can be reduced dramatically, both from a student's perspective and the institution's (no need for room and board, for example); The tedious immigration and naturalization issues common with international students are eliminated; The limited campus facilities, faculty members and course schedules an institution can offer are no longer a boundary; Working adults can consider upgrading skills without changing their lifestyles
We are presenting through this material a sequence of projects developed at University of Bridgeport and than can serve well in distance learning education ranging from simple "hobby" style training to professional guidance material. The projects have an engineering / laboratory flavor and are being presented in an arbitrary order, topics ranging from vision and sensing to engineering design, scheduling, remote control and operation.

Keywords: distance learning, remote teleoperation, remote manipulation, web control, e-learning

I. INTRODUCTION

In a supportive effort towards faster distance learning implementation, we present through this work a sequence of projects that have been developed and can serve the process of distance learning education ranging from simple "hobby" style training to professional guidance material.
The projects have an engineering / laboratory flavor and are part of ongoing work of the faculty and students of the Computer Science and Engineering Department of the University of Bridgeport, and are being presented in an arbitrary order, topics ranging from vision and sensing to engineering design, remote control and operation. The inclination is towards being able to set up physical labs online allowing students to use equipment / machines (microscopes, manipulators, for example to handle substances or various objects), hence the goal of creating tools to allow engineering lab-based courses to be offered via distance learning.

II. MOBILE ROBOT CONTROLLED BY A PHONE

In an attempt to add to the many possible ways of automating and implementing remote engineering, this project presents a complete, in depth, cost-effective solution for controlling a robot through phone calls. Various extension possibilities are being discussed as well (instructing a robot for vacuum cleaning, changing switches, moving objects, surveillance, etc).
Mobile Robots have numerous applications: unmanned exploration, land mine removal, energy plants and manufacturing factories.
We introduce a cost-effective robot. With the introduction of video cell phones it will be possible for the user to see the robotic movement in real-time and possibly perform educational exercises using a simple interface at a distance. Examples include “calling” a robot on the way home from work and have it do various jobs like vacuuming the home and sprinkling the garden. This could also be done by logging on to a web server via an internet connected device and sending signals to the robot [1], or directly sending specific commands to the robot through cellular phones to a wireless (or cellular) server at home (with or without a web connection) (figure 1).

Figure 1: An application of a Robot with a phonechip

We focus on communication with the robot. The main objective was to make the robots more user friendly and to be able to communicate with them. Thus, we decided to design a simple prototype of a robot that can be controlled via phone - "Phonebot". We considered a design based on a phone chip

Hardware/Software and Test Equipment Used

Hardware Requirements:

Flex 10K70 Chipset
Talrik II Robot (including the Servo motors and Sensors)
TelePhone Set
Teltone's M-8870-01 Chip
SPST 90-2323, 5V relay switch from RadioShack
3.58MHz Crystal Oscillators (Part #)

Variable Power Supply
Resistors, Capacitors and Diodes
Software Requirements:
Altera's MaxPlus II, OrCAD, HTML, CGI/ASP, Java, Matlab
Test Equipment:
Oscilloscope, Logic Analyzer, Multimeters, Nerd-Kit

Implementation
A call is placed to the Phonebot using either Plain Old Telephony System (POTS) or a Personal Communication System (PCS), or by using the telephony server via the Internet. Once the Phonebot receives the ring, a ring detector circuit detects it and the call is completed by establishing a connection between the phone chip and the FPGA chip (figure 2).
The project was implemented using a top-down process:
- sending a signal through the phone line
  a) Ring Detect and connect phone line
  b) DTMF decoder
- robot control with the FPGA device
  c) Clock Division (VHDL program)
  d) Ring Detect (VHDL program)
  e) Motor Control (VHDL program)
  f) Robot Control (VHDL program)

The final part of the project involved assembling various parts of Phonebot and combining the different VHDL modules.

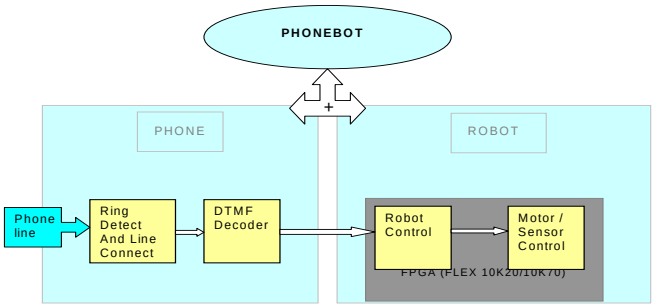

Figure 2: PHONEBOT – Basic Block Diagram

a) Ring Detect and Phone Line Connection
When the phone rings, the telephone company is sending a ringing signal, which is an AC waveform. The common frequency used in the United States is 20 HZ and in Europe it is typically 25 Hz and it can be any frequency between 15 and 68 Hz. Most of the world uses frequencies between 20 and 40 Hz. The voltage at the subscribers end depends upon the loop length and number of ringers attached to the line; it could be between 40 and 150 Volts.
The telephone line has only DC (-48V) and/or small signal AC (audio). In the circuit shown in Figure 3, capacitor C1 blocks the DC and the voltage divider circuit obtained from the R3 and R2 resistors prevent the low level AC from having any effect on the circuit.

Figure 3: Phone detect circuit

Specifications
C1 = 1 uf, CR1, CR2, CR3 = 1N914, C2 = 10 uF, R1, R3 = 100K, R2 = 10K
When the telephone rings, it brings about 90V RMS of AC at 20Hz. When the telephone rings, the capacitor C2 is charged. Diodes CR1 and CR2 guarantee that the output (ring detect logic) does not exceed the power supply levels and prevent any damages to other circuits driven from its output. Since C2 and R1 have the time constants of 1s, the output goes low for 1second after the ring stops. This pulse is to be detected and used for connecting the telephone circuit by the FPGA chip [2].

b) DTMF Decoder
Dual Tone Multi Frequency (DTMF) signals are used for speed dialing and replace the conventional rotary dialing system. These signals correspond to the digits on the dial pad of any modern touch-tone phone. Each of those touch-tones is constructed with the combination of two different frequencies. This information can be utilized to find out which button was pressed on the keypad and can be used for various applications. The construction of the signals according to different frequencies is shown in table 1:

Table 1: Signals related to different frequencis

	High Frequency Values (Hz)				
LowFrequency Values (Hz)		1209	1336	1477	1633
	697	1	2	3	A
	770	4	5	6	B
	852	7	8	9	C
	941	*	0	#	D

These frequencies can be decoded using precise filters and then we can decode which digit was pressed depending upon these decoded frequencies. We used the M-8870-01 DTMF decoder chip to decode this information. The chip uses a series of low pass and high pass filters to decode the frequencies. Then it uses a digital detection algorithm and a code converter to provide its output in the form of four binary output data [3]. This data is supplied to the FPGA chip for further use.

FPGA Control
For the controller we used the Altera's Flex 10K70 chip. The FPGAs contain arrays of logic cells, and enable designing real systems to operate at increasingly higher frequencies [4]. They have the ability to increase integration, to place more and more electronics in a chip and use all available gates within the FPGA, thereby providing cost-effective solutions. The most important

factor for selecting this chip is that we can program and reprogram the device while it is in a system. This provides us with great flexibility in using the Phonebot. The Phonebot can be programmed within a few minutes to do many tasks. Once programmed the robot will have the "intelligence" to complete the requested task.

c) Clock Division Module

This module divides the 25MHz clock of the Altera's University Board into slower clocks so that we can provide the precise timing for the servo-motors because these motors work at pulses in the order of milliseconds. The 25MHz clock is divided into 1Mhz, 100Khz, 10KHz, 10Hz and 1Hz.

d) Ring Detect Module

This module takes its input from the ring detector circuit and the DTMF decoder, provides output to the relay for closing or opening the telephone circuit. As soon as a ring is detected this module provides a 0V signal to the relay and thus closes the circuit. When it senses a DTMF code for digit 0, it opens the switch again. We had a small problem while disconnecting the phone. The FPGA device stored the code for digit 0 even after it was disconnected and for the same reason next time we dialed up, it would connect and disconnect on its own. This problem was resolved by resetting the signals in the FPGA device on the falling edge of the SB signal (SB signal goes high when there is a valid code detected by the DTMF decoder).

e) Motor Control Module

The Phonebot has two servo motors to aid its movement. First we had to hack these motors to create a DC gearhead motor. The information on hacking the servos was obtained from Mekatronix manual for the Talrik II robot. The hacked servos work on Pulse Width Modulation (PWM). This module provides the robot with pulses at specific times. If we provide a pulse for 1 to 1.5ms, then the motor spins in one direction and if we provide a pulse for 1.5 to 2ms, then the motor spins in another direction. The pulses have to be issued in an interval of about 20ms. This module implements the generation of this timing.

f) Robot Control Module

This module programs the movement of the robot according to the detected DTMF signals. It determines the path of the robot and provides a signal for the servo motor control module controlling the direction and the duration of running the motors. For example, if we press 7, it moves the robot in the forward direction for 4 seconds; if we press 8, the Phonebot moves in the reverse direction for 4 seconds; and if we press #, Phonebot comes to a complete stop. We also added few bump switches to the Phonebot. These bump switches send a high signal to the FPGA when they are bumped [5]. Depending upon which bumper detects switch closures, the FPGA device determines where the collision occurred and issues a signal to the Phonebot to change the direction of its path.

The final task in the project involved assembling the different modules together. We had to make a common ground for the FPGA device, the circuit we designed for the ring detection and the DTMF decoder [6]. The Altera board was not detecting the decoded signals produced by the DTMF decoder until a common ground was implemented. The main controller program was implemented in structural format combining the various components.

Instead of the DTMF decoder, a voice encoder/decoder can be used. Similar to a DTMF decoder, the voice decoder would recognize certain frequencies and depending upon those frequencies, the Phonebot can be programmed to do certain tasks. This feature could also be used for security purposes. Using more sensors and a visual feedback would make the Phonebot more animated and will be able to carry out more tasks. Integrating with the Internet would also add more access to the Phonebot. A movie showing the Phonebot can be seen at http://www.bridgeport.edu/cse/projects/phonebot/index.html, as well as the controlling software of the robot

III. INTERNET BASED SOFTWARE LIBRARY FOR THE SIR-1 SERIAL PORT CONTROLLED ROBOT

The idea of web based control has been always envisioned from the first days of networked computing. Being able to execute operations from remote locations [7], with feedback of some sort, it is an active and desired choice in many fields, such as robotic manipulation. This project presents a complete web based control solution for a manipulator, thus exemplifying one of the possibilities that remote automation encompasses.

A complex API for the control of the SIR-1 robot (figure 4) has been developed. Available functions include direct/inverse kinematics computation, serial port communication interfacing, and link speed control. The API can support an indefinite number of port connections and thus control a theoretically indefinite number of SIR-1 robots [8]. For the purpose of this project, two robots with their respective controllers have been used.

Due to the high availability of serial ports on standard PCs, this API can be deployed virtually anywhere and in any environment, including the Internet, on any applications.

Figure 4: The SIR-1 Robot

Programming Generalities

The SIR-1 controller accepts commands from the serial interface as CRLF-terminated ASCII strings. Prior to issuing any command, a handshaking sequence has to be performed. Once the character ESC has been sent, the controller will respond with an 8-bit integer. The respective bits indicate the status of each link (1 for active and 0 for

inactive). This is the only status update the controller will send throughout the process.
Commands are in the form:
<command char> <steps for link 1>, <steps for link 2>, … <steps for link 6>
For example, in order to move the first link 100 steps, the following string would be sent to the serial port:
M 100, 0, 0, 0, 0, 0 <CR><LF>
In this case, the trailing zeros can be left out.
Programming the controller via the serial port is thus reduced to assembling a command string in a buffer and sending it to the port character by character.

Web Usable Enhancements Library Set
Steps-degrees conversion: In order to control link movement by degrees rather than steps, a simple proportional correspondence between the number of steps each link can move and the angle it covers has been implemented [9].
Inverse Kinematics: Inverse kinematics functionality has been implemented. Link movement is no longer controlled only by angle, but also by absolute rectangular coordinates.
Programming Platform: A Visual Basic port of the library is implemented. This allows for web deployment and robot control over the Internet.
Function Library Abstract: Below is a summary of all functions present in the library.
int SIR1_Handshake();
Initiates communication with robot. This function must be called before any other command can be issued.
int SIR1_MoveLink(link L, int degrees);
Moves one link a specified number of degrees. If L = G (the gripper is moved), the integer degrees specifies millimeters of gripper opening.
int SIR1_MoveRobot(int AlphaBase, int AlphaShoulder, int AlphaElbow, int AlphaPitch, int AlphaRoll, int PercentGripper);
Moves all links a specified number of degrees.
int SIR1_GotoXYZ(int X, int Y, int Z, int PITCH, int ROLL);
Moves the center of the gripper fingers to coordinates X, Y, Z, with a specified roll and pitch of the gripper segment.
void SIR1_SetSpeedLinks(SPEED B, SPEED S, SPEED E, SPEED P, SPEED R, SPEED G);
Sets the speed of each link to a specified value (valid values are 0 to 7, of type SPEED defined at the top of **sir.h**).
int SIR_SetSpeedRobot(SPEED S);
Sets the speeds of all links to the specified value.
void SIR1_Pause(int TIME);
Pauses robot for 1/100 * TIME seconds. No return value
int SIR1_LinkPosition(link L);
Returns the position of each link (in degrees) relative to the absolute system of coordinates defined for the robot. For the gripper the percentage of opening is returned.
int SIR1_IsActive(link L);
Checks to see if link L is active

Although the SIR-1 is a relatively simple robot, it can accomplish complex tasks, due to its high repeatability (0.6 mm according to specifications).
The project clearly demonstrates the endless possibilities of using such a robot as SIR-1 for web/internet based remote automation through the implemented API, from pressing buttons, flipping switches or remotely controlling any other similar interfaces, to distance learning applications.

IV. INTERNET BASED COMPUTER VISION FRAMEWORK FOR SECURITY, SURVEILLANCE AND TRACKING APPLICATIONS

This project presents a possible framework to easily design and implement vision systems that perform useful real-time tasks using only off-the-shelf hardware, typically available through general consumer stores.

1) The Problem Domain
The general goal of any computer system is detection and identification of object models in an input image or stream of images [10]. Such general tasks are typically difficult to achieve with satisfactory speed and accuracy. Depending on the application, certain assumptions can be made about input images. For example, these assumptions could include: *Lighting condition in input images is known and/or constant; Object orientation in the input image is known; It is known that objects are not rotated out of the vision plane; It is known that objects are not rotated in the vision plane; Object is rigid; Object has constant shape; Object scale is known; Background of image scene is known; Sequence of images is available; (And /or possibly several others).* Such assumptions make it possible to build vision systems that operate in real time and give acceptable accuracy [11].

2) Architecture
Each algorithm dealing with a well-defined and solvable problem is implemented as a building block. The system can be built out of blocks connected using a plain pipe architecture where the output of a block is fed to the input of other blocks. Each system would have image acquisition and output analysis with vision processing implemented in between. This allows for a quick system implementation, without modifying underlying components. Components can be enhanced or replaced without affecting other parts of the system taken into consideration [12].
Generally, each system would consist of three large components: *Early processing, Feature extraction and Feature matching*
Early processing is the stage in which images are enhanced without trying to interpret them. Components that can be used in this stage include: *Gaussian filters, Histogram normalization and Color filtering*
Feature extraction selects certain feature points in the image that are relevant. Sample components within this module include: *Edge detection, Line or ellipse detection,*

Region growing, Region splitting and Minmax point extraction
Feature matching is the task of determining if model features are present in the feature set extracted from the image and how well they match.

3) Sample System Requirements
A system has to detect a face in the scene and follow it. Scaling, rotation in the vision plane are allowed. Small to medium variation in lighting is tolerated. Background is arbitrary. Rotation out of the vision plane and significant lighting variations are not allowed. The diagram of consecutive steps can be formulated as follows:
Acquisition -> Color filtering -> Conversion to Monochrome -> Gaussian blur -> Thresholding -> MinMax feature extractor-> Heuristic significant feature detection -> Feature matcher -> Match result
The steps from Color Filtering to Thresholding constitute the early vision processing. They only enhance the image for later processing without extracting any higher-level feature data.
The color filter removes regions that have colors considered not possible in the model. For example, the sky and forest background are removed in this step. Conversion to monochrome makes the image suitable for processing with other algorithms. Gaussian blur removes noise and many small, insignificant image features. It significantly lessens the number of features returned by the feature matcher. *Thresholding* removes parts of the image that are too dark, and therefore cannot contain objects for detection. Values for color filter, threshold and Gaussian blur can be specified at design time or extracted from the model.
The last four steps constitute high-level vision processing. *Feature extractor* scans the image and finds parts of the image that are most significant for detection. In this sample system, MinMax search is chosen because of its speed and simplicity. Local minimum and maximum points of image are selected as the most important points and used in the feature-matching algorithm. Applying a heuristic filter can filter features that are not relevant for detection. Relevance of a feature in this filter is defined as the energy carried by the min or max point. Energy is the summation of pixel values that are affected by that minmax point. *The feature matcher* performs optimized exhaustive search between features detected in the image and in the model and assigns each pair of combinations a match value. Matching is based on evaluating angles between features, proportions of distances and proportions of amplitudes. In this way, the matching function remains effective if rotations in the vision plane, scaling and uniform lighting variations are present. The combination with the highest match value is chosen as a possible detection candidate. If the match value is higher than threshold value, detection is successful; otherwise, the model is assumed not present in the image.

5) System Test Results
The original image (figure 5) has been passed through color filtering, blurring, thresholding, minmax feature extraction and heuristics to remove less significant features (figure 6).

Figure 5: Original Image

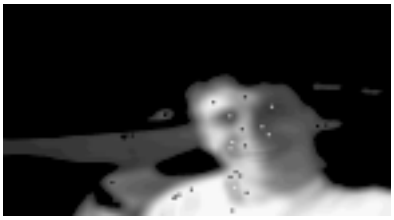

Figure 6: Heuristics to remove less significant features.

The importance of a feature is measured in terms of the energy it carries. After filtering, the remaining features are significant and the combinations of all of them can be examined quickly. Detections happen in less than one second using a Pentium PC and VGA resolution color images when detecting one model. Lower resolution images are suficient for many applications, therefore the system could be used in real time for the detection of more than one model [13].

6) Possible extensions
The modular design of the framework allows easy modifications and extensions to the system, such as using the system as a web based robotic eye that can follow faces. Sample modifications to improve the quality of detection include Fourier filtering and edge enhancement in early processing. In late processing, different feature extractors can be used – such as Hough transforms based methods, line following, snakes, region splitting and region growing. The matcher can be efficiently replaced with neural networks or other heuristics instead of exhaustive search. Background subtraction and light elimination could also improve detection rates. In order to further enhance functionality, the system could be extended with multi model matching, which would make possible the detection of objects rotated out of the vision plane.

V. INTERNET CONTROLLED SATELLITE TRANSPONDER (USING A REMOTELY CONTROLLED ROBOTIC MANIPULATOR)

a) Objective
To be able to control a satellite transponder from a remote location by using a robotic manipulator to mechanically change controls on the receiver or a UHF/infrared remote control of the receiver.
The reasons a robot was used to mechanically use the controller unit instead of hard-wiring the controller unit (or

the UHF remote control) to the server were: (1) The process of hard-wiring can permanently damage the expensive controller unit and subsequently render the transponder useless [14,15]; (2) The robot can be easily reprogrammed to adapt to different controller equipment, such as working with a remote controller.

To establish communication between the remote client and the robot via the server and enable the robot to carry out the desired commands without exceeding its limited workspace or without running into obstacles, the following tasks needed to be implemented:

- Interfacing between robot and server (RS-232)
- Interfacing between receiver and server
- Interfacing between server and internet
- Teaching the robot (trajectory planning and generation)

A flowchart of the system is shown in figure 7.

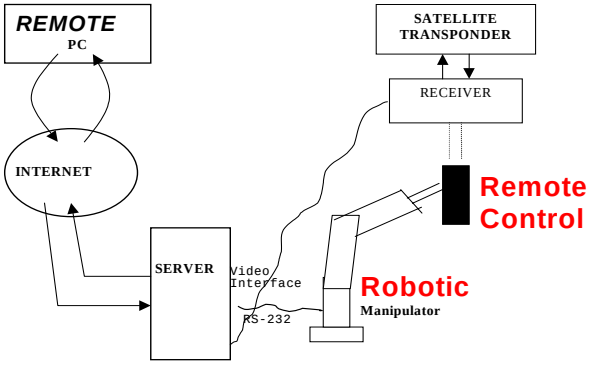

Figure 7: System Schematics

The interfacing between the robot and the server is carried out via the RS-232 serial port. The robot's controller box simply receives commands as ASCII strings through the RS-232 connection. An important issue to notice is the one-way communication (i.e. once the computer issues commands to the robot, the controller does not send back any signal to indicate the execution status of a command) [16]. To overcome this, it is important that a flawless trajectory be planned during the initialization of the robot, implemented so that obstacles do not pose a problem.

Sample commands that can be issued to the robot: **GC:** Closes the gripper, **GO:** Opens the gripper, **NT:** Nest the robot

The interfacing between the receiver and the server is done through a special interface card that takes the analog feed from the transponder's receiver and converts it into a digital format ready to be broadcast over the Internet. The PC card receives a feed through a standard coaxial cable. The transponder can point at different 'look-angles' and at each look-angle it can receive feeds at numerous frequencies.

The interfacing between the server and the Internet is done through a simple secure web server to carry HTTP requests from clients. It is important to ensure that only one individual is controlling the robot at the same time.

Lastly, the robot must be 'taught' about its surroundings and the environment that it will be operating in. This can be done by fixing the position of the remote control unit and defining its position relative to that of the robot. This allows the robot to 'know' where the controller unit is and where particular controls on the unit are.

b) Constraints and limitations:

- Limited set of commands that the robot's controller box can understand
- One-way communication between the server and the robot's controller
- The web server's clients have no access to the serial port, which is the only way of talking to the robot
- Time constraints
- Compatibility issues force use of a certain operating system

How were the above limitations overcome to achieve the objective?

The following tasks had to be carried out:

Configure the server-to-server HTTP requests

- Write software allowing HTTP clients to issue commands to the robot via the RS-232 port
- Write software to initialize the robot and define the work area

Server configuration was not complicated, resulting in setting up a Windows NT server with IIS, for processing requests and enabling security authentication and control.

An *ActiveX control* needed to be written to allow Active Server Pages to run executables on the server itself.

An *executable* 'writetoport.exe' did the simple task of taking command-line arguments and sending these arguments to the robot as commands via the serial port. The application was written using Visual Basic.

c) Robot initialization

As mentioned previously, robot-initialization is vital to ensure error-free operation, especially since the robot's controller unit sends back no signals to the server itself. The position of the remote control needs to be fixed and positions within the robot's work area need to be defined.

There are two basic types of movements: *Point to Point (PTP) movement, XYZ movement*

PTP movement involves specifying the joint variables (the joint angles in this case) and changing them according to the desired position of the robot.

XYZ movement involves moving the end-effector in an X, Y, or Z plane. The joint variables are calculated based upon position and desired motion and accordingly changed and recalculated at every point during the trajectory [17].

XYZ movement is much slower and inaccurate than PTP movement since the robot needs to perform repeated calculations to determine the desired position. Hence PTP movement was used to minimize error between the desired trajectory and the actual trajectory. In order to use PTP movement the trajectories need to be well-defined; each of the joint angles need to be specified and effected simultaneously to allow smooth trajectories.

List of resources (not exhaustive):

MoveMaster EX robot, Satellite transponder/receiver, Interface card for receiver, PC or compatible, Windows NT

4.0 (server), Internet Information Server 4.0, C++, Visual C++, Visual Basic, VBscript (for ASP)

d) Advantages and potential uses:
- Remote satellite feed for Internet users
- Remote communication
- Distance learning

e) Conclusion
- Robot was set up in the actual work environment
- Communications were established
- Rigorous testing was done to avoid unnecessary damage

The project demonstrates the *synergy* created by combining *robotic* and *computing power*. On a larger scale this concept can be ported to many more pragmatic applications using robots.

Bibliography

[1] Tanenbaum, Andrew S. Computer Networks, New Jersey, Prentice Hall.
[2] Meyers, Robert A. Encyclopedia of Telecommunications, Academic Press Inc.
[3] Skahill, Kevin. VHDL for Programmable Logic, Addition-Wesley, 1996.
[4] Yalamanchili, Sudhakar. VHDL Starter's Guide, Prentice Hall, 1998.
[5] Doty, Keith L. TALRIK^{II} Assembly Manual, Mekatronix^{TM}, 1999.
[6] Internet: ftp://ftp.teltone.com/pub/8870.pdf
[7] Ferrell W. R., Sheridan T. B.; Supervisory control of remote manipulation; IEEE Spectrum, October; 81-88; 1967
[8] Rastogi A., Milgram P., Drascic D., and Grodski J. J. [1993] Virtual Telerobotic Control Proceedings of the Knowledge-Based Systems & Robotics Workshop, Nov. 14-17, 1993; 261-269
[9] Rastogi A. [1995] Design of an interface for telerobotic in unstructured environments using augmented reality, M.A.Sc. Thesis, Department of Industrial Engineering, University of Toronto.
[10] A. Kosaka, M. Meng, and A. C. Kak, *Vision-guided mobile robot navigation using retroactive updating of position uncertainty*,' Proceedings of 1993 IEEE International Conference on Robotics and Automation, Vol.2, pp.1-7, 1993.
[11] A. Kosaka and A. C. Kak, *Fast vision-guided mobile robot navigation using model-based reasoning and prediction of uncertainties*,' CVGIP--Image Understanding, Vol. 56, No. 3, November, pp.271-329, 1992.
[12] A. C. Kak, K. M. Andress, C. Lopez-Abadia, M.S. Carroll, and J. L. Lewis, *Hierarchical evidence accumulation in the PSEIKI system and experiments in model-driven mobile robot navigation*, in Uncertainty in Artificial Intelligence (M. Henrion, R. Shachter, L. N. Kanal, and J. Lemmer, Eds), pp.353-369, Elsevier, North-Holland, 1990.
[13] P. Isto, *Path Planning by Multiheuristic Search via Subgoals*, Proceedings of the 27th International Symposium on Industrial Robots, CEU, Milan, 1996, 712-726.
[14] A. Li and G. Crebbin. *Octree encoding of objects from range images. Pattern Recognition*, 27(5):727-739, May 1994.
[15] W.E. Lorensen and H. E. Cline. *Marching cubes: A high resolution 3D surface construction algorithm*. In Computer Graphics (SIGGRAPH '87 Proceedings), volume 21, pages 163-169, July 1987.
[16] S-F Chang, J.R Smith, H.J. Meng, H. Wang, and D. Zhong. *Finding images/video in large archives*. D-Lib Magazine, February 1997.
[17] Stavros Christodoulakis and Peter Triantafillou. *Research and development issues for large-scale multimedia information systems*. ACM Computing Surveys, Dec 1995.