RISCBOT: AN EXPERIMENTAL TELEROBOTIC SYSTEM

SAROSH PATEL[1], RAJEEV SANYAL and TAREK SOBH
RISC[2] lab, University of Bridgeport, Bridgeport, CT 06601, U.S.A.

Abstract — This paper describes RISCBOT[3], an experimental 802.11b - enabled mobile autonomous robot built at the RISC Lab of the University of Bridgeport. RISCBOT localizes itself and successfully fulfills www - enabled online user requests and navigates to various rooms, employing a visual recognition algorithm. This article describes the mechanical design, hardware and software algorithms of the robot, and the web - based interface for communicating with the robot.

Keywords – RISCBOT, Telerobotics, Autonomous Control, Mobile robot, Web-based control.

I. INTRODUCTION

Telerobotics (controlling robotic devices from a distance) has enjoyed a rich history. It has led to many practical applications and to a broad vision of interacting with environments far removed from the user. With the advent of the Internet, telerobotics has received a major boost.

A number of mobile robots exist in the world today, catering to online requests to navigate to a desired location. Xavier [3] can accept commands to travel to different offices within a CMU building, broadcasting camera images as it travels. Minerva [1] is an interactive autonomous robot that moves daily through crowds at the Smithsonian's National Museum of American History. Rhino [2] has been deployed as a tour guide robot at Deutsches Museum in Bonn, Germany.

The experimental online robot we built, RISCBOT, utilizes room visual identification for localization. RISCBOT was built with the purpose of operating in a commercial office environment. RISCBOT is equipped with an onboard PC (personal computer), WLAN (Wireless Local Area Network) card, NM6403 based DSP (Digital Signal Processing) board, batteries, cameras and ultrasonic sensors. The robot has been designed modularly in order for researchers in the future to be able to add new features to the present system. Online users receive real time video feedback from the robot and can also view the robot position. Navigation is performed with the help of the cameras and ultrasonic sensors. The robot processes images from the camera to differentiate between doors, walls and obstacles. The robot can navigate the University of Bridgeport (UB) Engineering Technology building and successfully fulfills online user requests from the internet. It can run uninterrupted for about an hour, but has to be recharged hourly.

RISCBOT navigates the second floor of the University of Bridgeport (UB) Technology (TECH) building and can identify eleven rooms. A network camera (AXIS) is installed at the ceiling of the second floor of the TECH building, providing the online user with a view of the robot and the different rooms. The installed wireless access point has a range of 50 feet.

II. MECHANICAL CONSTRUCTION AND HARDWARE DESCRIPTION

The initial design of the robot was implemented using Pro Engineer (ProE). The robot frame is built with T slotted aluminum extrusion rods to allow for a modular structure. A differential drive mechanism has been implemented with two 4” wheels and a caster wheel for support. Two 12V dc servo-motors (Pittman) drive the wheels. An ATM 103 MCU, inverter (purchased off the shelf) and a 12V Panasonic SLA battery are mounted on a ¼” acrylic sheet. The PC cabinet housing the WLAN card and an NM 6403 DSP board is mounted on top of the base. Three ultrasonic sensors, two Logitech cameras and an NTSC camera are mounted on the PC cabinet. Figure 1 shows the mobile base design.

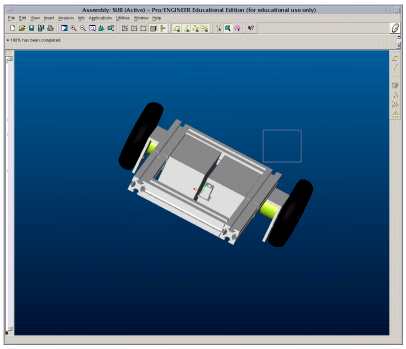

Figure 1: ProE mobile base design.

The ATM103 MCU controls the ultrasonic sensors and the two motors. The PC sends commands to the MCU through a serial port at 9600 bps baud rate. A MATLAB program that checks for doors runs on the PC continuously. The NM6403 DSP board performs a visual recognition algorithm when signaled by the PC. Figures 2a and 2b show different views of the mobile robot platform. Figure 3 shows a sample task performed by RISCBOT.

[1] Contact author for correspondence: saroshp@bridgeport.edu
[2] RISC lab: Interdisciplinary **R**obotics, **I**ntelligent **S**ensing, and **C**ontrol laboratory at the University of Bridgeport.
[3] “Riscbot” name has been coined from RISC lab and ‘bot’ from robot.

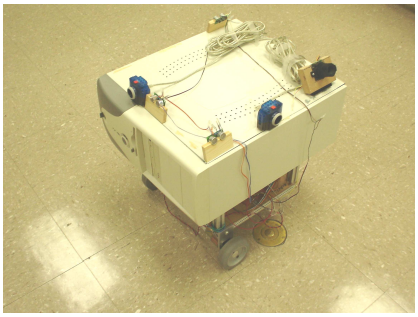

Figure 2a: RISCBOT Top view (front).

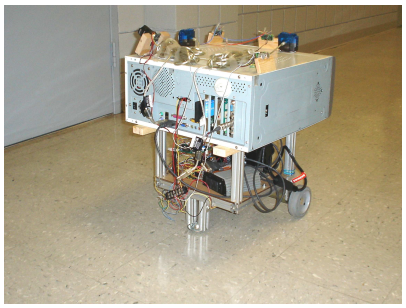

Figure 2b: RISCBOT Top view (Back).

Figure 3: RISCBOT Accomplishes its task.

NM6403 is a high performance dual core microprocessor with combination of VLIW/SIMD architectures. The architecture includes two main units: a 32-bit RISC Core and a 64-bit VECTOR co-processor to support vector operations with elements of variable bit length. Figure 3 shows the NM6403 based reconfigurable high-performance multi-DSP development set being used in RISCBOT for real-time video and image processing [5]. The development set consists of four NM6403 DSP processors, 16 MB async. SRAM, 16 MB SDRAM, 1 MB Flash, NTSC and PAL video decoder and PCI host interface. The PC and the NM6403 share 4 MB SRAM.

Atmel's ATM 103 is an 8 bit RISC MCU being used to control the dc motors and interface with the PC board (serially) and the ultrasonic sensors. The data from the ultrasonic sensors is monitored in an interrupt based embedded C code on the ATM 103. The main program running in the 8-bit CPU for controlling the motors depends on the instructions sent serially from the PC.

III. NAVIGATION MODULE

The robot waits till it receives information from the server. Once it receives a command from the server it starts searching for the requested room. Figure 4 shows the navigation module.

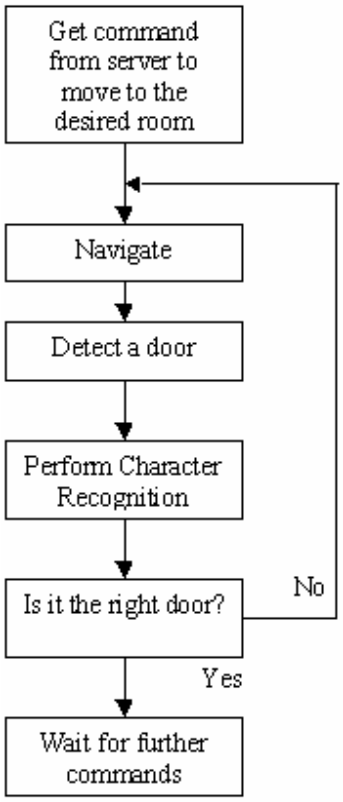

Figure 4: Navigation module.

The robot navigates along the wall to the left side of the corridor. With the help of the onboard ultrasonic sensors the robot maintains a safe distance of 45-50 cm from the wall. If the robot gets closer to the wall, it turns right, if it get further away it turns to the left and if the distance from the wall is within 45-50 cm the robot continues to move straight. If the robot encounters a wall right in front of it (example, at corners), it takes a right turn.

The image processing program checks for doors continuously. Once the program detects a door, it signals the NM6403 DSP board to check for the room ID. If the room ID matches the requested ID, the robot stops. If not, the robot continues moving till it finds the desired room.

IV. IMAGE PROCESSING

The door recognition algorithm is computationally fast, so that doors can be recognized in real time and appropriate commands can be sent to the Navigation module to stop the robot in front of the desired door.

Our algorithm employs edge detection to differentiate between the wall and the door. As the walls are rougher, the edges can be easily detected by selecting an appropriate order for the filter.

We used various filtering techniques for edge detection. Best results were obtained using a Gaussian – Laplacian filter [9] also commonly known as the Log filter, with an order of 1.7. The order of the filter should be carefully selected, since with the increase in order of the filter, undesired and very minute edges show up.

This module is programmed in MATLAB. Images from the camera are captured on the run using the vcapg2 utility [7], since MATLAB 6 does not have native support for the USB port. Images are captured at a resolution of 352 x 288 pixels. The auto gain for the camera is turned off so that all the images are captured with a constant gain.

Figures 5 – 9 show a set of images captured by the camera and Figures 10 – 14 show some results of the edge based door recognition algorithm. These images are converted to gray scale and then filtered to recover the edges.

Figure 5-9: Images captured by the Robot.

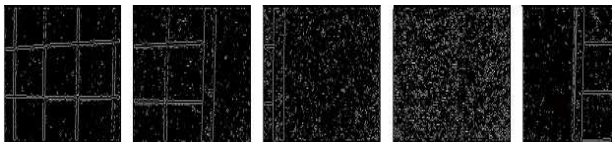

Figure 10-14: Images after Edge Detection.

As seen from figures 10 – 14, the edges are represented by zeros (white). As the robot encounters a door, there is usually a sharp drop in the number of zero - valued pixels in the image, since the door has no fewer edges than the wall. This can be clearly seen in the graph in Figure 15.

The graph depicted in Figure 15 was plotted as the robot moved across the corridor encountering doors. The graph shows the number of white pixels (or edges in each frame). Sharp drops indicate that the robot has encountered a door. This technique is not independent of varying light intensity. In regions where there is high light intensity, clearer edges can be seen.

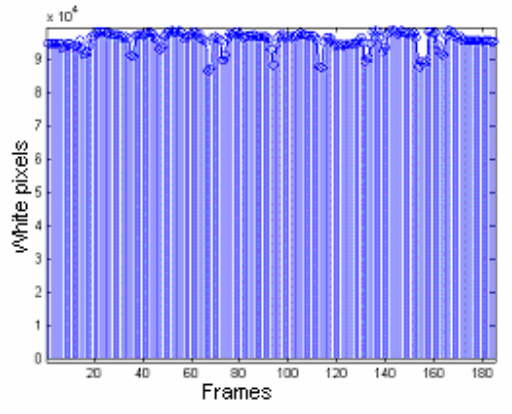

Figure 15: White Pixels (Edges) Vs Frames.

A clear example of how the intensity of light varies along the corridor can be seen from the stem plot depicted in Figure 16. The smooth curves show the varying light intensity and the sharp changes are the doors.

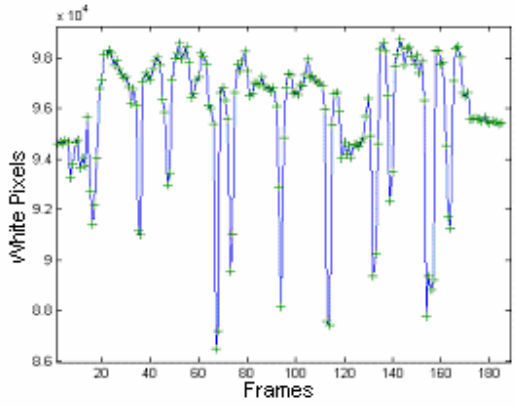

Figure 16: Varying light intensity.

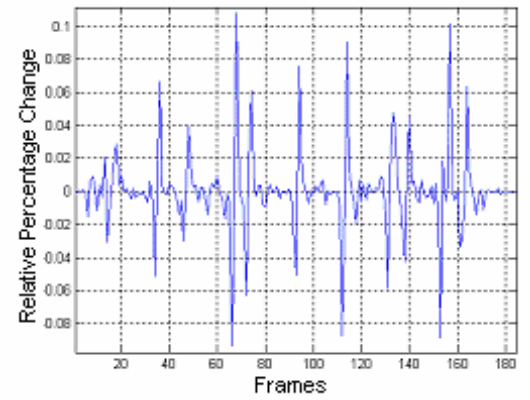

Figure 17: Relative Percentage Change.

A better strategy to recognize a door is to monitor the relative percentage change in the edges. When there is a

drop in the relative percentage, below a particular threshold, the robot is assumed to have encountered a door. Figure 17 shows a plot for the relative percentage verses the frames. Sharp negative peaks below a threshold of 0.03 indicate doors. Figure 18 shows a plot of recognized doors.

The program maintains an internal count for the doors encountered, to utilize if the door recognition algorithm fails. In addition, adequate measures have been incorporated so that when more than one image of the same door is captured; the robot does not treat them as two different

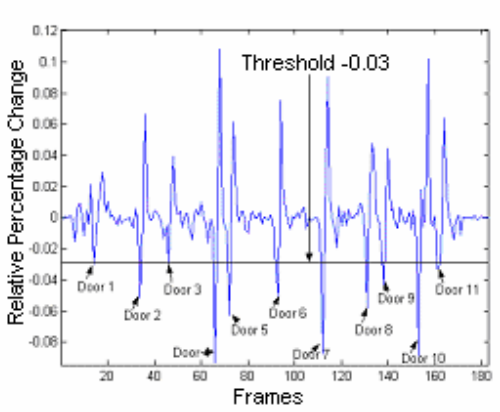

Figure 18: Plot showing the doors recognized.

doors. Once a door is recognized, the control is passed to the Recognition Module in order to recognize the door.

V. CHARACTER RECOGNITION

The door ID character recognition algorithm runs on the NM6403 DSP board.

5.1 Identifying the door plate:

The room numbers have been printed on a plate and stuck on the doors. The image -processing algorithm identifies the plate on the door. We have implemented this task by using the Hough transform for detecting lines first, and then checked the relative dimensions of the lines for detecting the plate. We implemented the ideas described in [8].

Figure 19 shows the steps involved in detecting the plate [8]. Thresholding and edge detection is performed to reduce the number of processed points. The resulting image is then fed to the Hough transform, which then returns a list of straight lines.

The algorithm for detecting a straight line (adopted from [8]) is as follows:

1. Select start pixel S (x/y).
2. Select end pixel E (x/y).
3. Follow line from S to E pixel-by-pixel and count the number of pixels on that path that are set in the binary image.
4. If the counted number of pixel is greater than the threshold value, a line SE is present in the picture and hence is labelled. Go back to step 1 and select two different pixels until the entire image is done.

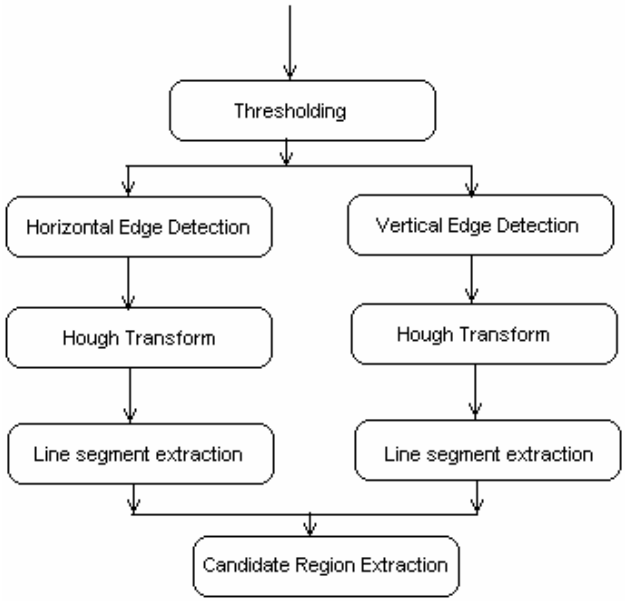

Figure 19: Steps for detecting the plate [8].

Finally, the list of straight lines is scanned to find pairs of straight lines that have the following attributes:

1. Both lines start and end at approximately the same position on the x-axis.
2. The relation between the length and distance of both lines should be equal to that of a door ID plate.

This algorithm is carried out on both horizontal and vertical lines. The resulting pairs from the list of horizontal and vertical lines are then compared and the end points not contained in both regions are discarded. The remaining regions are the final candidate regions, which contain the plate. A detailed explanation can be found in [8].

5.2 Character Extraction

Characters are extracted using a region - growing method [8]. The extracted space (from the top) is searched for a black pixel. When a black pixel is found it is assumed

that it is part of a character. An iterative process is then carried out using the following equation described in [8]:

$$Xk = (Xk-1 \oplus B) \cap A \qquad k = 1,2,3\ldots.$$

Where Xk represents the extracted component, A is the source image and B is a structuring element of size 3x3 indicating 8-connectivity neighboring. X0 is the first black pixel from the location where the iteration starts.

This iterative algorithm creates a new image X_0, which only contains the first black pixel. X_0 is dilated (i.e. expanded) such that it contains all its neighboring pixels. Any of the neighboring pixels that are also black are considered to be a part of the component and are put into a new image X_1 along with the original black pixel to complete the first iteration of the algorithm. X_1 is dilated in the next iteration. This process continues until $X_k = X_{k-1}$.

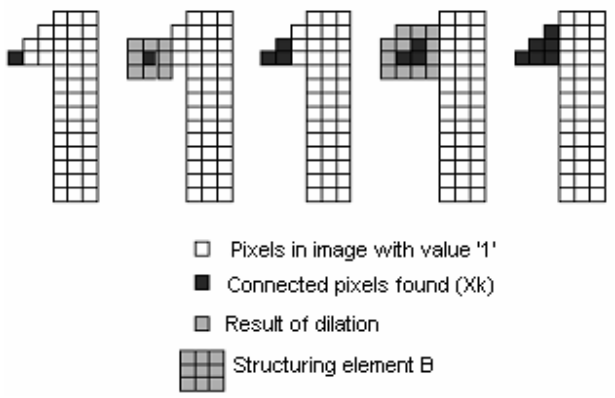

Figure 20: The first two iterations in the process [8].

When a connected component is found it is then tested to see if it meets the requirements of a character (e.g. size). This whole process is carried out iteratively until all the characters in the page have been extracted. Each character is labeled. Figure 20 illustrated the first two iterations in the extraction process [8].

5.3 Symbol Recognition Technique

We have implemented the symbol recognition technique described in [9]. In [4], Sridhar et al describe a collection of topological features that can be used to classify numerals. Most of these features are properties of the outline or *profile*, of the numeral.

After the digit is isolated and thresholded, the number of background pixels between the left side of the character's bounding box and the first black pixel is counted and saved for each row in the bounding box. This provides a sampled version of the left profile (LP), which is then scaled to a standard size (52 pixels). A similar process produces the right profile (RP); the difference between the processes is that the last black pixel on each row is saved. Having computed the profiles, the properties of the profiles are measured, as described in [9], as follows:

1. Location of extremas:
   a) Lmin: location of minimum value on the left profile.
   b) Lmax: location of maximum value of the left profile.
   c) Rmin: location of minimum value on the right profile.
   d) Rmax: location of maximum value of the right profile.
2. W(k) = Width at position k = RP(k) – LP(k).
3. Wmax, the maximum width of the digit; this is W(k) at some point k where RP(k) – LP(k) is a maximum.
4. R, the ratio of height to maximum width.
5. Based on first differences:
   a) LDIF(k) = LP(k) – LP(k-1).
   b) RDIF(k) = RP(k) – RP(k-1).

Based on the above properties, 48 features are computed for each sample numeral. The features are described in [4].

VI. THE WEB INTERFACE

The web interface is an integral part of the mobile navigation and identification process. The mobile robot is connected to the Internet through an onboard WLAN 802.11b card. The robot can be controlled and viewed from the internet, through its website: www.bridgeport.edu/sed/risc/html/proj/RISCBOT/index.htm.

Updates on the web services and server availability information are posted on the website. Users can also download videos and pictures of sample navigation and recognition tasks performed by the robot. The web interface for the robot is simple, consisting of three windows: the control window, top view window and the camera view window.

Figure 21 shows a view of the web interface while the robot is navigating. Once logged on, any user can send a request to move the robot to a particular door by selecting the appropriate door ID on the control window. A real time video feedback is provided as the robot broadcasts video while moving. The feedback is implemented using Microsoft Media Encoder. This video can be seen on the Top view window and the camera view window.

The top view window shows the video feedback from a network camera mounted on the ceiling. This helps the

remote user visualize the exact position of the robot and issue any further commands if needed.

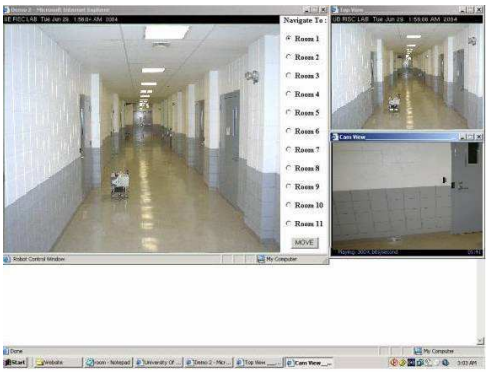

Figure 21: RISCBOT website.

The camera view window provides the user with virtual presence in the corridor. The robot encodes the images captured and broadcasts it on the web.

VII. COMPUTATIONAL TIME AND EFFICIENCY

The speed of the robot varies from 50 cm/s – 1 m/s depending on the battery condition. The wall detection time using the ultrasonic sensors varies from 75 ms to 100 ms/ sensor, depending on the distance of the robot from the wall. A time lag of 10 ms was employed between the firing of two successive ultrasonic sensors to avoid crosstalk. Experimental results with the character recognition algorithm running in isolation under ideal conditions showed an efficiency of 99%. Experimental results with the character recognition algorithm running on the robot showed an efficiency of 80%. The failures (no decision on a particular character) were due to partial or no acquisition of door plate, due to the proximity of the robot to the door. However, while the robot catered to the requests of online users, an efficiency of 95% was recorded. An internal door count algorithm running in a MATLAB program overcame failure of the character recognition algorithm.

VIII. FUTURE WORK

Potential future enhancements to this project include:

1. Implementing data transfer algorithms for faster transfer of high priority data packet.
2. Developing fast computing algorithms for the image-processing module.
3. Exploring ways to enhance RISCBOT's cognitive and sensing capabilities.
4. Implementing a dc – dc (ATX power supply) converter circuit that will increase the power conversion efficiency and thereby the operational time for the robot.
5. Permitting the robot to recharge itself by plugging into wall outlets.
6. Mounting a manipulator on the mobile platform for implementing mobile manipulation tasks.

ACKNOWLEDGMENT

We would like to thank Prof. Craig Dubois of the Industrial Design Department at UB who helped us with the mechanical construction, assembly and design. We are thankful to Dr. Khaled Elleithy for providing us with the WLAN access points and access card. Special thanks to Mr. Dmitry Fomin and Mr. Alex Ruzavin of Module Research Center, Russia, who helped us with the frame grabber code in the NM6403 DSP board.

REFERENCES

[1] S. Thrun et al "MINERVA: A second-generation museum tour-guide robot." In *Proceedings: IEEE International Conference on Robotics and Automation (ICRA '99)*, Detroit, Michigan, USA, May 1999.

[2] W. Burgard et al "The interactive museum tour-guide robot", American Association for Artificial Intelligence (AAAI-98), Madison, Wisconsin, July 1998.

[3] R. Goodwin, K. Haigh, S. Koenig, and J. Sullivan "A Layered Architecture for Office Delivery Robots" *First International Conference on Autonomous Agents*, Marina Del Ray, California, February 1997.

[4] Kimura, F. and M. Shridhar, *Handwritten numerical recognition based on multiple algorithms*, Pattern Recognition, Vol. 24, 1991, No. 10, pp. 969-983.

[5] NeuroMatrix Board Support Kit, "Multiprocessor module assembly NM42, Programmer's Manual", Rev. 01 95 02A, Module Research Centre, Russia, 2003.

[6] Atmel ATM103 Datasheet, Rev. 0945SE – 01/00/xM, Atmel Corporation, USA, 2000.

[7] Kazuyuki Kobayashi, "MATLAB Utilization Book", Shuwa System Co, Ltd., 2001, ISBN 4-7980-0141-4.

[8] Henrik Hansen et al "Automatic recognition of license plates" Institute of Electronic Systems, Aalborg University, Denmark, May 26, 2002. http://www.vision.auc.dk/~tbm/Student_projects/02-automatic.pdf.

[9] J.R. Parker "Algorithms for image processing and machine vision", ISBN 0-471-14056-2, Wiley Computer Publishing, 1997.