IEEE ISIE 2005, June 20-23, 2005, Dubrovnik, Croatia

RISCBOT: A Mobile Surveillance and Identification Robot

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], a wirelessly enabled autonomous navigational and surveillance robot. The robot can be controlled through the Internet and can be commanded to move to any desired location. The robot navigates using a visual recognition algorithm, also providing a real-time video feedback from the operation site. This paper describes the hardware, software and the wireless interface modules of the RISCBOT.

I. INTRODUCTION

Autonomous mobile robots are one of the modern marvels of interdisciplinary scientific research and development. They provide a great opportunity of interacting with environments for removed from the user. With the advent of the Internet telerobotics (controlling robotic devices from a distance) has received a major boost.

Telerobotics has enjoyed a great past, with many popular mobile navigational robots operating in public places such as museums. For example, Xavier [1] can accept commands to travel to different offices within a CMU building, broadcasting camera images as it travels. Minerva [2] is an interactive autonomous robot that moves daily through crowds at the Smithsonian's National Museum of American History. Rhino [3] has been deployed as a tour guide robot at Deutsches Museum in Bonn, Germany.

The autonomous surveillance and navigational robot, RISCBOT, built at the RISC Lab, University of Bridgeport utilizes a room visual identification for localization. RISCBOT navigates the second floor of the University of Bridgeport Technology building catering to the online user requests. The robot can identify and navigate to eleven rooms on the second floor all while providing a real-time video feedback. A network camera installed on the ceiling provides the online users with view of the robot as it moves and the different rooms.

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 navigates using the onboard cameras and ultrasonic sensors. The robot processes images from the camera to differentiate between doors, walls and obstacles.

Designed to provide an efficient telerobotic system over the Internet, RISCBOT is modular in design. The region of operation is limited only by the coverage of the wireless access point and its onboard battery life, which is about an hour for uninterrupted operation.

II. HARDWARE DESCRIPTION

Pro Engineer (ProE) was used to design and visualize various configurations of the robot in the initial design phase. The best design was selected based on stability criterions, speed, floor clearance, turning radius and appearance. RISCBOT has been built with T-slotted aluminum extrusion rods as it provides the liberty of altering the design and making changes with ease. Figure 1 shows different views of the ProE design for the mobile base.

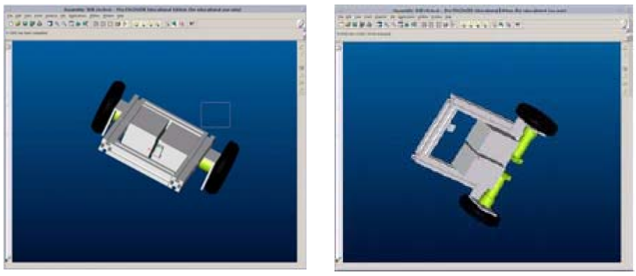

Figure 1: Different views of the ProE mobile base design.

RISCBOT navigates using a front end differential drive, implemented with two 4” wheels and a rear caster wheel for support. Two 12V dc servo-motors (Pittman) drive the wheels. An ATM103 MCU controls the ultrasonic sensors and the two motors. The PC sends commands to the MCU through the serial port. Atmel’s ATM 103 [4] 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.

Figures 2a and 2b show different views of the mobile robot platform. Figure 2c shows a sample task performed by RISCBOT.

[1] Contact author for correspondence: saroshp@bridgeport.edu
[2] RISC lab: Interdisciplinary Robotics, Intelligent Sensing, and Control laboratory at the University of Bridgeport.
[3] “RISCBOT” name has been coined from RISC lab and ‘bot’ from robot.

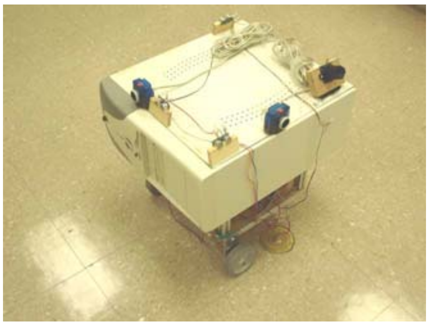

Figure 2a: RISCBOT Top view (front).

Figure 2b: RISCBOT Top view (Back).

Figure 2c: RISCBOT accomplishes its task.

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. The NM6403 DSP board performs a visual recognition algorithm when signaled by the PC. NM6403 is a high performance dual core microprocessor with combination of VLIW/SIMD architectures [5].

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. NM6403 based reconfigurable high-performance multi-DSP 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. Figure 3 shows the NM6403 based reconfigurable high-performance multi-DSP development set being used in RISCBOT for real-time video and image processing.

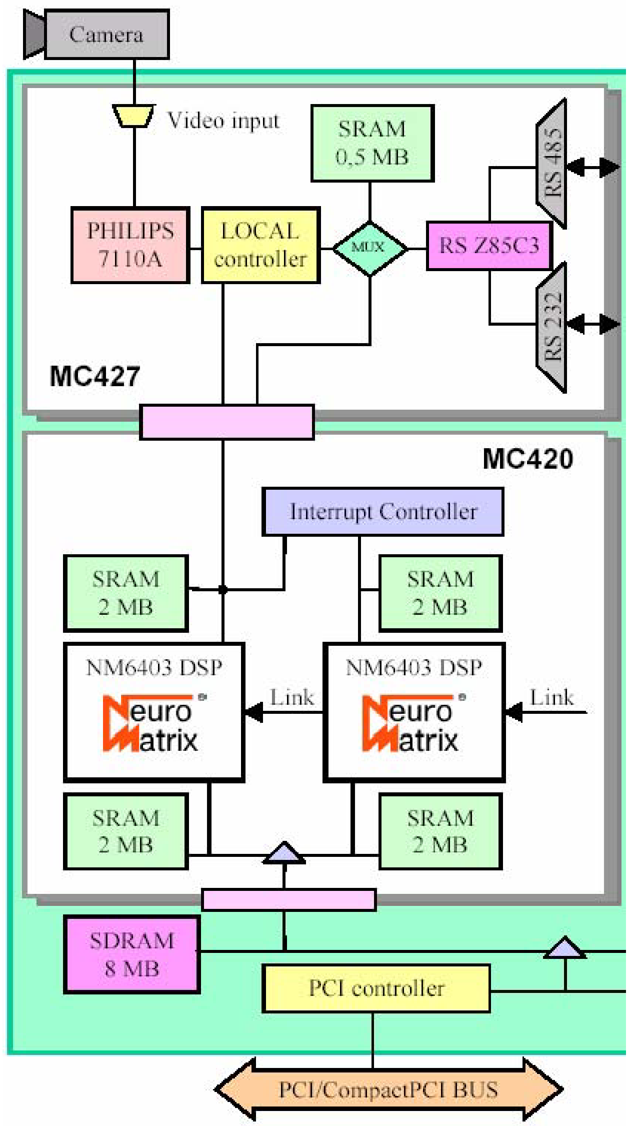

Figure 3: Architecture of the NM6403 based DSP board.

III. NAVIGATION MODULE

The robot remains at rest until it receives a command to move to a particular position from the server. Once it receives a command from the server it starts searching for the requested room. Figure 4 shows the control flow diagram for the RISCBOT.

The robot keeps to the left side of the corridor. The robot maintains a safe distance of 40-50 cm from the wall with the help of the onboard ultrasonic sensors. Depending on the input from the ultrasonic sensors the Atmel MCU takes appropriate action to control the motor speed and direction. A time lag of 10 ms was employed between the firing of two successive ultrasonic sensors to avoid crosstalk. If the robot gets closer to the wall, it turns right, if it get father 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. Figure 4 shows the control flow diagram for the navigation module.

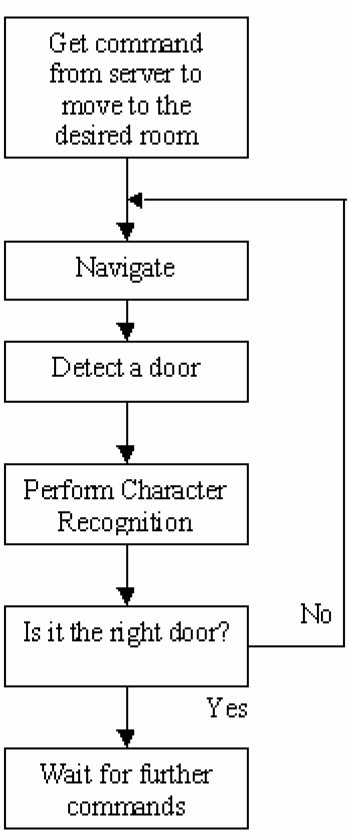

Figure 4: Navigation module.

When the image processing program encounters a door, it signals the DSP board to confirm the door ID. If the door ID matches the requested door ID the robot strops. If not the robot continues to move ahead until it finds the desired door.

IV. DOOR RECOGNITION MODULE

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 Laplacian of Gaussian filter [6] 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 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.

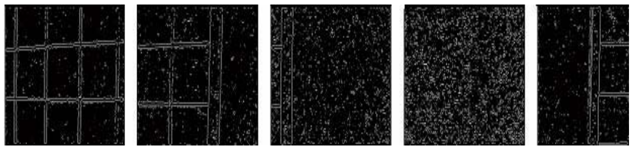

Figure 10-14: Images after Edge Detection.

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.

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.

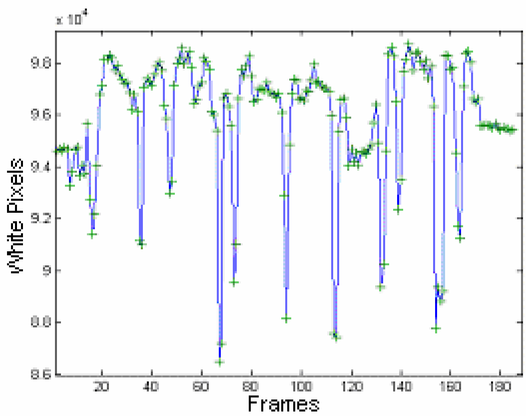

Figure 15: Varying light intensity.

A clear example of how the intensity of light varies along the corridor can be seen from the stem plot depicted in Figure 15. The smooth curves show the varying light intensity and the sharp changes are the 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 doors. Once a door is recognized, the control is passed to the Recognition Module in order to recognize the door.

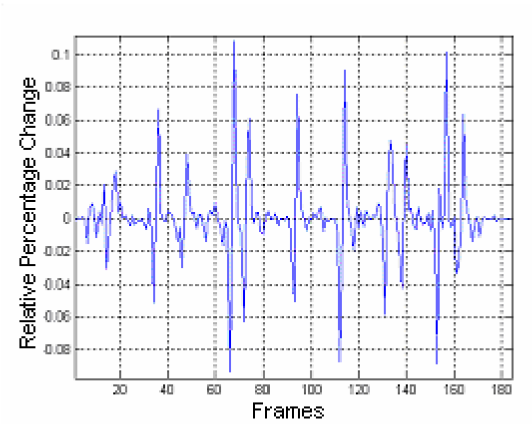

Figure 16: Relative Percentage Change.

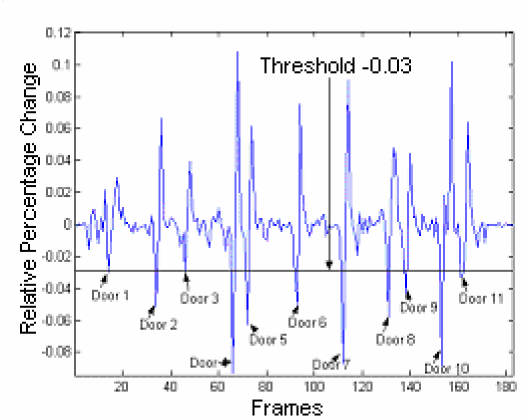

Figure 17: Plot showing the doors recognized.

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 16 shows a plot for the relative percentage verses the frames. Sharp negative peaks below a threshold of 0.03 indicate doors. Figure 17 shows a plot of recognized doors.

V. CHARACTER RECOGNITION

The character recognition module is the last part in the control chain. The door ID character recognition algorithm programmed in C language runs on the NM6403 DSP board.

A. 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].

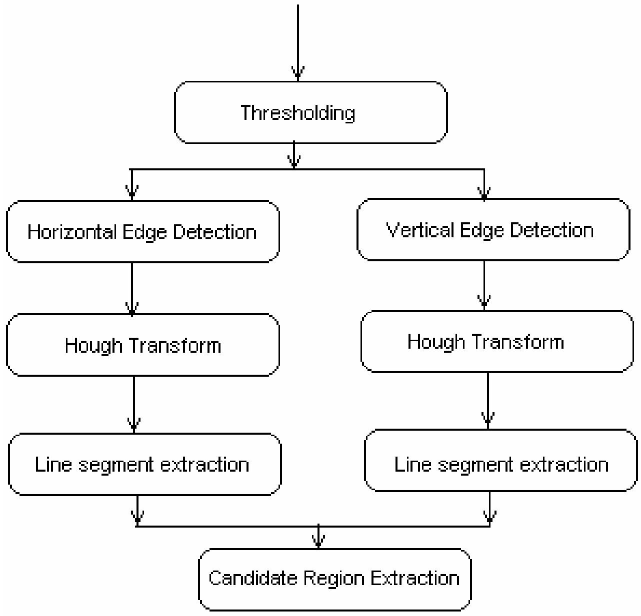

Figure 19: Steps for detecting the plate.

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 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.

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].

B. 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]:

$$X_k = (X_{k-1} \oplus B) \cap A \qquad k = 1,2,3\ldots.$$

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 $X_k = X_{k-1}$.

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. The complete character extraction process is explained in great detail in [8].

C. 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 [6].

VI. THE WEB INTERFACE

The web interface is an integral part of the telerobotic system. The mobile robot is connected to the Internet through an onboard WLAN 802.11b card. The web interface is as simple as selecting the desired room number and clicking the "Move" button. The robot can be controlled and viewed from the internet, through its website: www.bridgeport.edu/sed/risc/html/proj/RISCBOT/index.htm.

The web interface for the robot is simple, consisting of three windows: the control window, top view window and the camera view window. Figure 20 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.

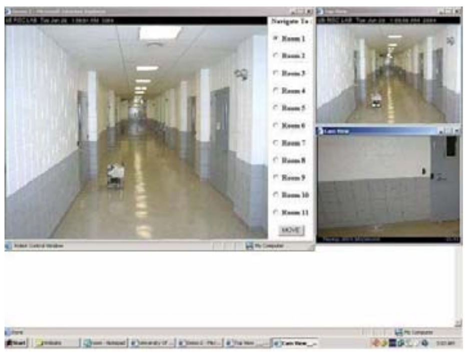

Figure 20: RISCBOT website.

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. 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.

Updates on the web services and server availability information will be posted on the website. Users can also download videos and pictures of sample navigation and recognition tasks performed by the robot.

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. The time is dependent 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 MATLAB program overcame failure of the character recognition algorithm.

VIII. FUTURE WORK

Potential future enhancements to this project include:

1. Designing and building pan – tilt units for the cameras.
2. Implementing a dc – dc (ATX power supply) converter circuit that will increase the power conversion efficiency and thereby the operational time for the robot.
3. Permitting the robot to recharge itself by plugging into wall outlets.
4. 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] 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.

[2] 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.

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

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

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

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

[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.

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