DESIGN OF AN AUTONOMOUS ROBOT FOR INDOOR NAVIGATION

Michael McConnell, Daniel Chionuma, Jordan Wright, Jordan Brandt, Liu Zhe (Students), Dr. Kurt Kosbar (Advisor)
Telemetry Learning Center
Department of Electrical and Computer Engineering
Missouri University of Science and Technology

ABSTRACT

This paper describes the design and implementation of an autonomous robot to navigate indoors to a specified target using an inexpensive commercial off the shelf USB camera and processor running an imbedded Linux system. The robot identifies waypoints to aid in navigation, which in our case consists of a series of quick response (QR) codes. Using a 1080p USB camera, the robot could successfully identify waypoints at a distance of over 4 meters, and navigate at a rate of 50 cm/sec.

Keywords: robot, indoor navigation, image processing, imbedded Linux

INTRODUCTION

This paper describes the system level design of a small robotic platform that has a sensor package and on-board data analysis processor which will allow it to autonomously navigate in an indoor environment. The objective of the project was to design a system which can be used for reconnaissance in an environment where the terrain is reasonably flat, such as in a residential or commercial structure. We assumed the design of the structure would be unknown in advance, either because the information was not available, or because it was in a format that was not convenient to the robotic system. The robot would execute a program that would allow it to systematically explore and map the environment.

To successfully map an area, the system needs a way to both sense its surroundings, and determine its location [1]. Since the intended application is for indoor environments, we assumed services such as the Global Positioning System would not be available because the structure could attenuate or reflect the signals to the point that they would not be useful. As an alternative, inertial navigation and distance measuring by using either an odometer (counting
revolutions of a wheel) or non-contact distance sensing using optical techniques can be used. These techniques, taken as a group, will be referred to in this paper as inertial position sensing.

One of the difficulties in using inertial position sensing is that position and heading errors accumulate in the system, and it can be difficult to detect and remove. This leads to maps which can have significant distortion. This problem is particularly severe when the robot rotates, or changes its heading. A very small error in the estimation of the angle of rotation will cause the internally generated map to be skewed. This problem becomes increasingly severe as the robot moves, or executes multiple turns. As a result the robot can eventually find itself back in a room it has previously visited, however the map it has generated internally can indicate otherwise.

A positioning system that does not suffer from accumulation of error could rectify this problem, but we assume that is not available for this application. This problem is routinely encountered by humans, and solved through the use of landmarks [2-5]. A human observer will notice unique objects or perspectives within a room in a building. When the human returns to the same room, they notice the same objects, and conclude they must have returned to the same location. In some cases this is done through a system specifically designed for that purpose, such as room and floor numbers. However even in a less structured environment, humans can often determine when they have returned to a location they have visited before.

In the design of this system, we plan to use a single camera to identify objects that appear to be unusually, if not globally unique, in an environment. The image of these objects will be stored in memory, so the device can recognize when it has located them again. While any objects can be used for this purpose, we chose to use quick response (QR) codes which we placed on walls in the structure specifically for this purpose [6]. These codes would most likely not be available in an environment where the user of the structure was not cooperating with the robotic system. However there may be a number of commercial applications where this would be useful. For example there may be systems that are used to monitor, clean, or transport items within a building. These devices are often powered by rechargeable batteries. When their batteries need to be recharged, they must locate a power outlet or other recharging station. In these applications it would be reasonable to identify such a location with a QR code posted nearby.

**ROBOT DESIGN**

The primary focus of this project was the design of a vision system and associated image processing hardware and software to locate landmarks or waypoints. Because of this, a commercial off the shelf robotic platform was used. A number of platforms were considered, and in the end we selected a small tracked vehicle, as shown in Figure 1. This was chosen because of its ability to travel over both smooth terrain and overcome small obstacles. Another useful feature is the zero-radius turning. This allows the robot to scan an entire room from one
location while most wheeled vehicles require the robot to move forward or in reverse to allow it to change its heading.

When selecting a processor, microcontrollers were initially considered but proved to have insufficient memory to be able to store and process the large images required to correctly identify waypoints from a distance. In the end, the Raspberry Pi [7] embedded Linux system was selected because of its abundance of processing power, availability of general purpose input/output (GPIO) to control motors, the onboard USB ports, and the ability to store the operating system image on a removable SD card. The Raspberry Pi model B is an inexpensive commercial off the shelf single board computer developed by the Raspberry Pi Foundation and available from many sources. The processor is based on a Broadcom BCM2835 system on a chip. This chip contains an ARM1176JZF 700 MHz processor and VideoCore graphics processing unit. There is a public domain operating system based on Debian and Arch Linux which can be run on this architecture. A photograph of this board is shown in Figure 2.

Our project utilized a webcam to take pictures of our current surroundings. Originally, we believed that the least expensive webcams would be sufficient for our needs. However during testing and evaluation we found that it had a significant restriction on the size of the image we collected. While the specifications of the device indicated 1280 x 1024 pixel resolution, in practice it was incapable of achieving an actual resolution above 640x480 pixels. This was insufficient for our application, so we elected to upgrade to a webcam that supports
H.264/MPEG-4 Part 10 advanced video coding. This camera allowed us to capture compressed images at up to 1920x1080 pixel resolution. This allowed landmarks to be identified when they were up to 4.2 meters away.

During our early testing, we noticed that our Raspberry Pi would randomly stop recognizing our webcam when we took pictures. This issue led us to create an onboard powered USB hub that we could programmatically turn off and on through our GPIO pins. The idea behind this was that it would limit the amount of time that our webcam would be on and in doing so, minimize the odds of the webcam disconnecting. After we swapped to powering our Raspberry Pi through the GPIO pins, we found that the webcam no longer had this issue.

Figure 2  Raspberry Pi (Courtesy Raspberry Pi Foundation)

Upon further analysis, we found the issue came from the current limitation of the micro USB connection used to power our Raspberry Pi in testing.

A 7.6V, 2200 mAh lithium polymer battery was used to power the entire robot. The motors were powered with 7.6V via the H-bridge while the Raspberry Pi, camera, and Wi-Fi adaptor (which was used for debugging) were powered via a 5V switching regulator made by Sparkfun Electronics, which is based on a Lineage Power Pico TLynx power module. The projected
current consumption was 1 amp from the motors, 1 amp from the Raspberry Pi, and 1 amp from the camera and wifi module for a total of the 3 amps, which could easily be supplied by the lithium polymer battery, which had a maximum current of 60 amps, and the regulator, which had a maximum current of 6 amps. The measured average current consumption for different stages of the QR code search is shown in figure 3. A full circuit diagram is shown in figure 4.

![Current Consumption](image.png)

**Figure 3  Current Consumption**

**IMAGE PROCESSING AND NAVIGATION**

Since all waypoints were being identified with a QR code, we were able to leverage open source software solutions for this application. The QR code library we originally used confirmed the presence of a QR code and what data the QR code contained. However it failed to indicate the area of the image that contained the code, or its dimensions. This is not surprising, since that is the most common use of QR code software. However for navigation purposes this was insufficient, so it was necessary to modify the software to allow it to return the pixel coordinates of the code within the image.

The robot’s movement is modeled after a tank as this allows zero-radius turning. Being able to turn without travelling facilitates easily scanning an entire room for a QR code and making heading corrections as the robot travels to the QR code. One of the most challenging aspects of designing the robot was determining the position of the robot relative to the QR code (distance and angle) with the limited sensory input from the camera. The zero-radius turning helped
alleviate this challenge by allowing the robot, even as an open loop system, to turn and drive the angles and distances specified by the software with acceptable accuracy.

To estimate the distance, the focal length of the webcam was calculated by equation (1) in a controlled test environment.

$$ f = \frac{d \cdot Z}{D} $$

(1)

Where $f$ is the focal length, $d$ is the width of the QR code in pixels, $Z$ is the known distance away from the QR code, and $D$ is the width of the QR code in inches. Using this focal length, the distance was approximated using equation (2).

$$ Z' = \frac{D \cdot f}{d'} $$

(2)

Where $Z'$ is the unknown distance and $d'$ is the measured pixel width of the QR code.

Upon finding a QR code, the robot turns and travels the calculated distance by assuming a fixed motor speed measured during testing. The calculated distance to the QR code is related to turning on the motors for a certain amount of time by equation (3)

$$ T = \frac{D'}{V} $$

(3)

Where $T$ is the time that the motors remain on, $D'$ is the distance to the QR code calculated earlier, and $V$ is the motor speed calculated during testing. The same theory was used to turn the
robot the calculated angle. Using this method, the robot could navigate to directly below the QR code roughly 2 cm from the wall on which the QR code was mounted.

One challenge to accurate navigation was imbalanced motors. While the robot only incorporated two motors (left and right), they were not speed balanced so the robot would not drive straight even when both motors were running on the same voltage. The motors were driven by a STMicroelectronics L293D H-bridge to enable two-quadrant (bi-directional) operation of the motors via the Raspberry Pi GPIO. To balance the motors, pulse width modulation (PWM) was used. A ratio between the duty cycles used on the left and right motors was determined to balance the motors and was implemented regardless of the motor direction. PWM was also used to slow the motors because the motor speed at a 100% duty cycle was unacceptably fast (50 cm/sec) and threatened the stability of the robot during turning.

An additional challenge was controlling the motors at startup. The output of the GPIO pins of the Raspberry Pi proved unpredictable and uncontrollable whenever the processor was initially booting. This was problematic as the random GPIO signals could activate the motors during boot up. This risk was mitigated by using a logic circuit between four GPIO pins and the enables of the H-bridge, effectively creating a hardware level “password” that only the QR code-finding program could supply. After implementation of this logic circuit, no more problems with motor operation at startup were encountered.

CONCLUSION

An autonomous robot was designed and built to navigate indoors to a specified location designated by a QR code using an imbedded Linux system and a webcam. Possible future work could include the introduction of either a second camera with stereo image processing or an infrared distance sensor to more accurately gauge distances and, with additional programming, to avoid obstacles. Providing feedback to the navigation algorithm via encoders on both motors would also be valuable as it would provide confirmation of the distance travelled by the robot and any change in direction. With these improvements, larger areas could be searched and the functionality could even be extended to room mapping.

REFERENCES


