WiFinder, a Wifi Signal Intensity Mapping Robot

Chaitanya Aluru, Sean Roberts, and Eric Wu

I. Introduction

The goal of the WiFinder project is to develop a robot that will find the strongest 802.11 signal in a room. In order to do this, the robot first maps out the 802.11 signal intensity using RSSI measurements, and then find the position of maximal signal strength intensity within the mapped area. In doing so, the robot implicitly solve the localization problem over a large, two dimensional grid. This requires the careful integration of several sensors, and precise control over the robot's movements. Here, we describe the implementation of our robot, model the behavior of the sensors used to localize and control the robot, and simulate the errors that arise due to imperfect sensor readings and errors in control.

The GitHub repository for this project is https://github.com/wueric/WifiRobot, and the demonstration video for the project is located at http://youtu.be/C_c6Be1Nw74.

II. Maximal Signal Strength Algorithms

We manually constructed several spatial maps of RSSI, which are shown below. We conducted measurements in multiple passes, to determine the amount of variability in signal intensity. While the general shapes of the spatial signal profile were approximately the same for the two passes, there were also major differences. These differences in the shape of the profile suggest that 802.11 signal strength is quite variable and unpredictable, making hill-climbing and other such algorithms infeasible and unlikely to result in finding a point of maximal strength.

As a result, we implemented simple, search-and-maximize algorithm. Rather than dynamically determine where to search, the algorithm prescribes a fixed rectangular grid pattern for the robot to trace out. The robot measures RSSI signal intensity at fixed distances along this grid, and then returns to the point of maximal signal intensity at the end of its search.

These algorithms were implemented using a state machine architecture, similar to the ones implemented in the course labs.

III. Hardware and Firmware

Our wifi-seeking robot was based upon the iRobot Create platform, manufactured by iRobot corporation. Our primary computational platform for the project was the Freescale KL25Z ARM microcontroller running the mbed framework. The iRobot Create was interfaced to the mbed board using a serial port. Because the iRobot Create required 5 V input signals, we were required to use a Sparkfun logic level converter to interface the iRobot with the mbed electrically. The serial connection between the iRobot and the mbed board was implemented using the Serial framework developed by the mbed team. Instructions were sent to robot using the iRobot Create Open Interface with code developed by our project group.

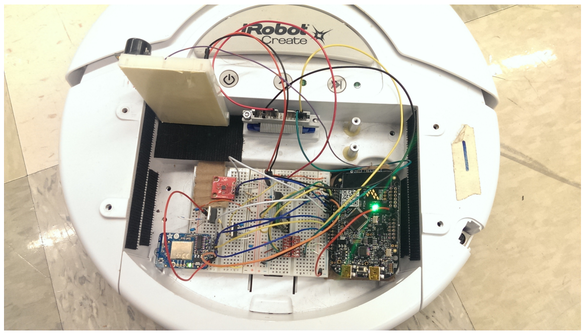

Fig. 1. Photograph of the iRobot Create electronics bay. The Maxbotix LV-EZ1 is on the forward-facing breadboard and is used to sense distance travelled. Inside the electronics bay we have a Honeywell HMC5883L magnetometer, an Adafruit CC3000 WiFi breakout board, a SparkFun BlueSMiRF Bluetooth modem, and a SparkFun logic level converter. All of these devices are integrated with a Freescale KL25Z microcontroller running the mbed platform.

We used the Adafruit CC3000 WiFi shield as our 802.11 signal intensity sensor. The wifi shield was interfaced to the mbed using the SPI interface and the mbed_cc3000 library developed initially by cc3000 chip manufacturer Texas Instruments and ported to the mbed platform by Martin Kojtal. In addition, we used a SparkFun BlueSMiRF Bluetooth modem to interface the robot with a Bluetooth-enabled laptop in order to send robot position coordinates and signal intensity readings to the laptop. This modem was interfaced to the KL25Z board using a serial port, using the Serial framework.

To help determine the robot's position, we used an ultrasound sensor to measure distance and a magnetometer to measure angular orientation. The ultrasound sensor was a Maxbotix LV-EZ1, put on a breakout board by Adafruit. This sensor was interfaced to the KL25Z using an analog connection (to conserve the KL25Z's limited number of serial ports). Data was read off of the LV-EZ1 using the KL25Z's onboard analog-to-digital converter (ADC) using the mbed AnalogIn library. Our magnetometer was a Honeywell HMC5883L 3-axis digital compass, put on a breakout board by SparkFun. The magnetometer was electrically interfaced to the KL25Z using an I2C connection, and was interfaced through software using code originally authored by Tyler Weaver, and modified by the project group to support sensor calibration.

A photograph of our hardware is provided in Figure 1, and a block diagram illustrating our hardware setup is provided in Figure 2.

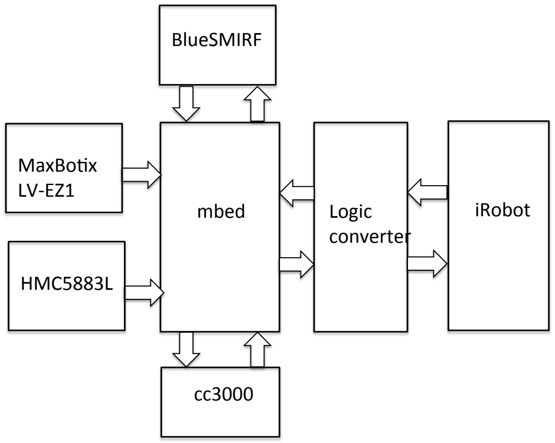

Fig. 2. Block diagram for our hardware setup.

IV. Characterizing and Modeling Sensors

e had a variety of sensors to choose from to aid us in localization. Our requirements were to be able to solve the localization problem on a grid up to 10 meters per side to within +/- $\frac{1}{3}$ m. To do this, we needed to be able to move forward one unit of distance at a time, and turn as close to 90 degrees as possible at the edges of the grid. For the forward motion, we considered the onboard accelerometer, an ultrasound module.

A. Accelerometer

To determine the accuracy of the accelerometer module, we laid it flat at rest on a table and read the measured acceleration values at one second intervals. In addition to this, we added integration code to measure the distance traveled by the robot in each direction. Initially, it measured about 2-3 $\frac{m}{s^2}$ on each axis. We realized that most of the error was systematic error, which could be removed. As a basic first step, we took the average of the first ten readings from the accelerometer, and subtracted this value from subsequent readings. After adjusting for bias, the readings were surprisingly accurate, usually within 0.1 $\frac{m}{s^2}$ on each axis. Unfortunately, even this small error quickly compounded. Within five readings, the integrated position was already about 0.3 m off on each axis, or about a foot in each direction. This level of inaccuracy was unacceptable for us. Although there are more advanced filtering techniques which could have gotten us better results, we realized that the accelerometer was unlikely to work for us. This is because the iRobot moves at a relatively slow velocity, and all of the acceleration is done in very short bursts. In order for an accelerometer to work in such conditions, the sampling rate must be very high during acceleration. Combined with heavy filtering code, the accelerometer would consume a significant portion of the processor's clock cycles, which could potentially conflict with other sensor readings and controlling the robot. Furthermore, since using dead reckoning requires that the accelerometer be continuously sampled, using it would introduce substantial concurrency issues in our software. For these reasons, we decided to perform localization using other sensors.

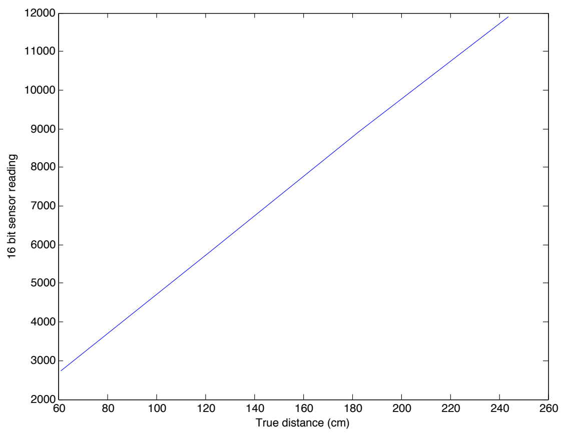

Fig. 3. Plot showing 16-bit ADC values for the LV-EZ1 ultrasound rangefinder plotted against the true distance of an object placed in front of it. The plot is very linear, which suggests that the sensor response can be modeled with the affine model.

B. Ultrasound Rangefinder

In order to accurately measure distance travelled using the Maxbotix LV-EZ1 ultrasound sensor, we first had to model and calibrate the sensor. This was done by connecting the analog output of the LV-EZ1 (where V_{CC} for the sensor was set to the recommended 3.3 V) to the KL25Z's onboard 16-bit analog-to-digital converter (ADC). We took data points by placing a large flat surface at 60 cm increments in front of the sensor and reading off the value returned by the ADC. The results of these measurements are plotted in Figure 3.

Because the sensor response was approximately linear with distance, we modeled the response of the analog output of the LV-EZ1 coupled ADC using the affine model of a sensor. Briefly, the output of the sensor is modeled as

$$f(x(t)) = a \cdot x + b$$

where f is the output of the sensor, and x is the true distance. Using MATLAB, we were able to calculate best-fit values for parameters a and b. According to these calculations, a was 50.18 $\frac{\text{bits}}{\text{cm}}$ and b was -292 cm. The affine model was then implemented in software using these values to accurately determine distance travelled.

C. Magnetometer

In order meaningfully read orientation from the HMC5883L magnetometer, the magnetometer had to be calibrated to offset hard-iron effects and soft-iron effects. Hard-iron effects correspond to magnetic fields that generate a generate a consistent, constant offset to the magnetometer readings. They correspond to permanent sources of magnetic fields (in our case, the magnets and coils for the iRobot's motors contributed significantly

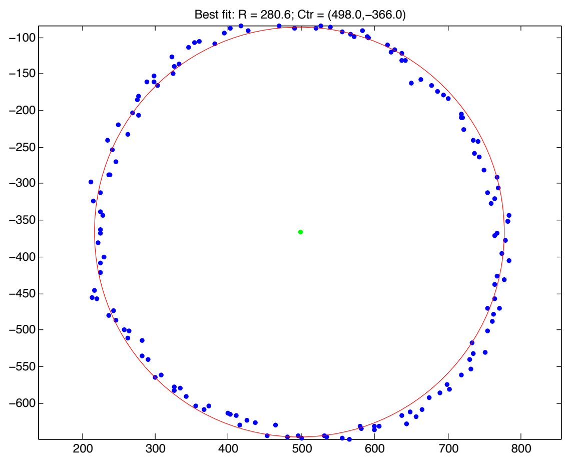

Fig. 4. Plot showing the x and y (labeled as z) components of the magnetic field as the sensor is rotated, and the best-fit circle for the data. The shift in the center of the circle nature suggests that there are significant hard-iron effects on the magnetometer, while the circular shape of the data indicates that soft-iron effects are minimal.

to the hard-iron magnetic fields). Soft-iron effects correspond to magnetic field distortions caused by nearby metallic objects (i.e. screws on the iRobot case).

We characterized the hard-iron and soft-iron effects on the magnetometer by placing the magnetometer on the iRobot and spinning the iRobot in place. While the iRobot spun, we repeatedly measured the magnetic fields in the x and y directions (relative to the robot's orientation) using an accelerometer. A scatterplot of these readings is plotted in Figure 4. In the ideal, no distortion case, the scatterplot should trace out a perfect circle with the center at the origin. Hard-iron effects will show up as a shift in the center of the circle in the plot, and soft-iron effects will show up as a distortion to the shape of the circle (i.e. making it more elliptical). Using the gathered data, we used MATLAB to find the best-fit circle and determine the constant bias terms corresponding to hard-iron effects. Furthermore, since the data points on the scatterplot form a nearly perfect circle, we assumed that the soft-iron effects on the magnetometer were negligible. We then implemented the offset in software to compensate for hard-iron effects.

V. Augmenting Sensor Readings with Wheel Speed

After calibration, our sensors worked well most of the time. However, we noticed that in certain situations, their readings would be off by an unacceptable margin. In these situations, it was very hard to guarantee worst case behaviour. For example, the ultrasound module was only accurate in a range of 2 - 10 feet, so when a wall was not present within that range, distance readings were very unpredictable. With the magnetometer, disturbances would occasionally cause angle measurements to become wildly inaccurate, sometimes causing the robot to spin endlessly in place. In order to combat this, we used the iRobots wheel speed measurements to bound the time taken for any action.

In order to do this, we first needed to experimentally determine the accuracy of the iRobots wheel speed measurements. The wheel speeds are given in $\frac{mm}{s}$, with a range or 0 - 250 $\frac{mm}{s}$. We found that the actual speed of the robot was consistently higher than what the speed was set to. However, actual wheel speed varied approximately linearly with the measurement, and we were able to find a speed and amount of time in which the robot would go very close to one foot. We used this to set a small window of time during which the robot was guaranteed to be close to a foot from its starting position. If the ultrasound module failed to measure a one foot distance within that time frame, then the ultrasound reading was discarded and the robot stopped at the end of the window.

Rotation was handled similarly. Because the robot always turns in place, we knew the wheels would move on a circle with diameter equal to the robots wheelbase. Using this, we were able to calculate the distance the wheels needed to travel for the robot to turn 90 degrees. We set a window of time for the magnetometer, and relied on it to prevent excessive rotation. Using these bounds, the robot was able to consistently sweep out a grid and return to any location within the grid with relatively small error.

VI. Simulating Effects of Sensor Error on Robot Movement

A. Modeling Assumptions

In order to model the behavior of the iRobot Create, we developed a Python simulation to simulate both the movement of the iRobot, and the effects that errors in localization and movement could have on the final path taken by the iRobot. Our model made several critical assumptions, namely that the robot accelerates instantaneously, travels on a level surface, and suffers from no wheel slippage. Noisy simulated sensor data was generated from the predicted ideal sensor data, and several parameters regarding error were tested.

B. Effects of Varying Angle Tolerance on Path

Because our sensors took a significant amount of time to update and because our software was unable to constantly monitor sensor values, our finite state machine allowed for transitions between the turning and driving forward states when the difference between the desired angle and the measured angle was within a certain tolerance. In order to determine the effect that threshold angles had on the robot's traversal of the grid path, we simulated the robot's behavior for different turn threshold angle values. The simulated paths are given below in Figure 5. From the plot, it is clear that increasing angle tolerance causes increasing errors in the angle turned by the robot, and thus increasing errors in the path taken by the robot. This simulation emphasizes the importance of writing software that reads sensor values as often as possible, as the inability to do so leads to compounding errors in position.

C. Effects of Magnetometer Calibration Error on Path

In addition, we modeled the effects of errors in magnetometer calibration error on the path taken by the robot. This

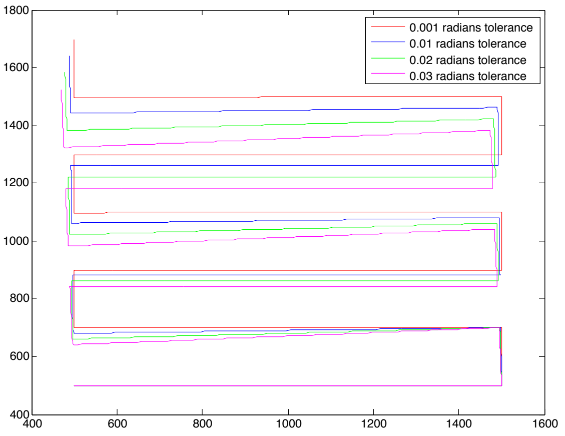

Fig. 5. Plot showing the simulated path taken by the robot for different amounts of turn angle tolerance. Tolerance is given in radians. While increasing the tolerance leads to increasing error, the error is relatively insignificant to other sources of angular orientation error.

simulation determined how the angle turned by the robot, and thus the grid path of the robot would be altered if hard-iron effects on the magnetometer were not adequately compensated for. We added a constant bias to simulated magnetometer readings, and determined the path that the robot would follow if given these erroneous readings. The paths for various amounts of hard-iron calibration error are given in Figure 6. These simulations demonstrate that even seemingly insignificant errors from the magnetometer (on the order of a few percent) could have substantial impact on the angle turned by the robot and thus on the path taken by the robot. They also suggest that very accurate measurement and control of angular orientation is required to maintain a rectangular path, and further demonstrates the need to augment the magnetometer with some other information to accurately sense orientation.

The hourglass-shaped paths generated by the simulations matched quite closely with paths taken by the actual robot in testing, suggesting that measurement errors with the magnetometer play a substantial role in the real-world deviation of the robot from its ideal path.

VII. Results

Using the search and maximize algorithm described above, and the sensor-plus-wheelspeed localization system, we were able to construct spatial maps of RSSI signal intensity for parts of Cory Hall. In Figure 7, we illustrate a 3D heatmap of RSSI intensity for a 12 foot by 12 foot square outside of the EECS 149 lab. The location of the router corresponds to the back corner of the plot.

VIII. Conclusion

We have designed, developed, and constructed a robot that searches a given space for the position of strongest 802.11 RSSI signal intensity. Using our grid search algorithm, the robot searches the space for the point of strongest signal, and returns to the point with the strongest signal. In addition, we have modeled the responses and sensitivities of the various sensors used to help localize and control the robot, and have simulated the effects that sensor error and imprecise control can have on the path that the robot has followed.

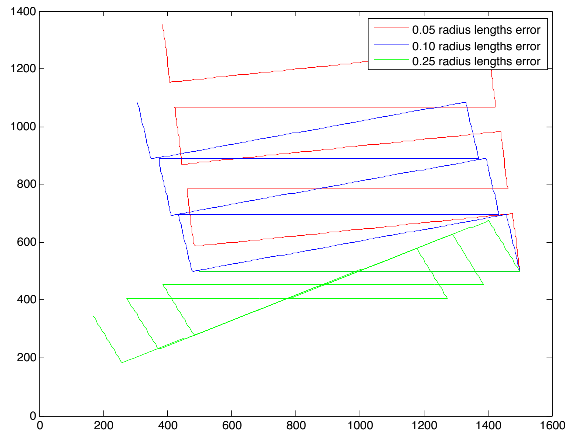

Fig. 6. Plot showing the simulated path taken by the robot for different amounts of magnetometer hard-iron calibration error. The key is given in radius lengths, which correspond to the amount of shifting of the center of the circle traced out by the simulated magnetometer readings. As calibration error increases, the error in the path increases dramatically, suggesting that accurate magnetometer readings are absolutely critical.

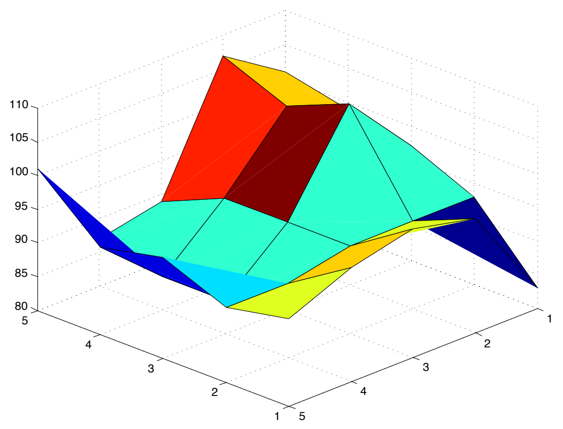

Fig. 7. A heatmap showing sample RSSI readings for a corner of Cory Hall. The location of the router corresponds to the far corner in the plot.

IX. Citations

1) Thomas, Teyvonia. "Vision-based Obstacle Detection and Path Planning for the IRobot Create Using an Android-powered Smartphone." *Vision-based Obstacle Detection and Path Planning for the IRobot Create Using an Android-powered Smartphone*. University of Pennsylvania, 18 May 2011. Web. 19 Dec. 2014.
2) "Hard-Iron and Soft-Iron Calibration PTM by Digi-Key and Honeywell." *Hard-Iron and Soft-Iron Calibration*

PTM by Digi-Key and Honeywell. Digikey, 2014. Web. 19 Dec. 2014.

3) Edward A. Lee and Sanjit A. Seshia, Introduction to Embedded Systems, A Cyber-Physical Systems Approach, Edition 1.5, http://LeeSeshia.org, ISBN 978-0-557-70857-4, 2014.
4) Weaver, Tyler. “HMC5883L”. *HMC5883L - a Mercurial Repository - mbed*. mbed, 2014. Web. 19 Dec. 2014.
5) Kojtal, Martin. “cc3000_hostdriver_mbedsocket.” *Cc3000_hostdriver_mbedsocket - a Mercurial Repository — mbed*. mbed, 2014. Web. 19 Dec. 2014.
6) iRobot. “iRobot Create Open Interface.” (n.d.): n. pag. IRobot. IRobot, 2014. Web. 19 Dec. 2014.
7) “Mbed in a Nutshell.” *Interactive Device Design Fall 2014*. UC Berkeley, 19 Sept. 2014. Web. 19 Dec. 2014.
8) “Wireless 1.” *Interactive Device Design Fall 2014*. UC Berkeley, 20 Oct. 2014. Web. 19 Dec. 2014.