Object Location Detection with Wireless Sensor Networks

Arun Ganesan
EECS, University of Michigan

Abstract

In this project, I aim to locate objects in a room using a wireless sensor network of ultrasonic rangefinders. The goal is to accomplish object localization using computationally cheap sensors, such as the ultrasonic rangefinders, without requiring any tagging or modification to the object being tracked, i.e., requiring invasive techniques. To approach this problem through rapid prototyping, I built a simulator and tested three different sensor layouts and algorithms for object detection. I compared the three layouts using error maps which visualized the error in the predicted location and the actual location of objects that are tracked. Finally, I also validate one of the layouts with measurements from a physical experiment. This project supports the possibility of non-invasive object localization using ultrasonic rangefinders.

1 Introduction

Object position detection is useful in many contexts. For example, facial position detection is useful in photography, car location detection is useful in traffic management, and the field of intelligent environments uses position detection to provide services to inhabitants of homes. Current approaches to location detection are 1) invasive, requiring the placement of lights, tags, or some kind of marker on the object to be tracked, or 2) computationally expensive, requiring sophisticated techniques from computer vision, and expensive sensors such as cameras. The goal of this project is to approach this problem non-invasively and with computationally cheap sensors. I used a set up of ultrasonic rangefinders arranged in a wireless sensor network around a room to locate objects using only the distance measurements.

This project has three independent aspects. The first is the the simulation, including a way to interact with the simulation, and a way to poll the algorithm for updates in the measurement of the sensors and location prediction. The simulation is powered by a model of the problem capturing the sensor detection
area, and the behavior of objects moving around a room. The second aspect of
the project is the algorithm for location prediction given a particular layout of
sensors. The final aspect is a physical setup of sensors to validate and inform
the simulations.

Given the time and resource constraints of this project, I’ve had to make many
simplifying assumptions in the detection environment. These are discussed in
the Discussion section below, however a few assumptions that are important
to understanding the methods and results are described here. First, I assumed
the algorithm is only detecting one object in the environment free from all
obstructions. Second, I assumed the sensors had no noise in their readings
and returned a perfect reading every time. Third, I assumed the room being
detected has a simple square shape.

Section 2 surveys some previous work done on location detection in indoor
environments. Section 3 presents the methodology for designing and testing
various sensor layouts in object location detection using the simulator. Section
4 presents a comparison of the results from various sensor layouts. Section 5
validates one of these simulations using a physical set up of ultrasonic rangepro-
ders. Finally section 6 discusses additional improvements and limitations of the
simulation, section 7 outlines future work, and section 8 concludes.

2 Related Works

A large number of works have focused on indoor localization of objects. The
field of smart environments aims to sense and build models of the environment
a home in order to provide more intelligent services such as selective lighting.
Smart Floor \cite{1} is a project from this field which determines the location using
pressure-sensitive tiles.

Other localization efforts involving similar technologies as used in this project
are Active Bat, Active Badge, and CRICKET.

The Active Bat \cite{2} approach involves tracking location by using ultrasonic re-
civers and transmitters. The object of interest, say a person, carries around an
ultrasonic transmitter and sends coordinated sound waves to the receivers. The
receivers are placed along the ceiling of a building and help locate the object
by determining the time it takes for the ultrasonic pulse to reach each of the
receivers.

CRICKET \cite{3} performs a similar localization by placing ultrasonic transmitters
on the objects and receivers in the environment. However, the way the
localization is accomplished is very different than Active Bat.

Active Badge \cite{4} localizes objects by placing infrared transmitters on the objects
and receivers in the environment. Using this technology leads to different sets
of challenges such as interference from existing sources of infrared.
3 Methodology

3.1 Simulation

The simulation was built using a server-client architecture to accommodate different ways of interaction. The server handles the logic of the model with the sensor readings, location prediction, and model of the world with the location of the sensors and the walls of the room. The sensor locations, the shape of the sensor beams, and the structure of the room were input to the model using a simple text file format. In addition, the server handles interacting with clients and sending updates of the sensor readings, and position prediction. The client interaction was implemented as a Python web server which passes data back and forth in JSON format to a web frontend. Separating the server from the ways of interacting with the model gives the advantage of being able to build different clients. For example, an interactive and visual client can be built for prototyping, and a batch-test, command-line client can be built for running large experiments.

For this project, I built a visual web-client which allows the user to control the location of an object (depicted by a human icon) on the map using arrow keys and polls the server for updates of the location predictions at every step. The client also requests and displays an error map of the current map and sensor layouts. This is described in the evaluation section. Figure 1 shows the client side of the simulation.

Figure 1: The client side of the simulation displays the map takes arrow key input to move the object around the map. The sensor values are displayed in the text boxes in the middle column, and the actual location versus the prediction and the error are displayed in the last column.
3.1.1 Model

The simulation involves modeling the physical components in software. Based on the data sheet provided by the manufacturer of the rangefinders I used in the physical experiment, I modeled the sensor beam as a narrow beam with a convex beginning, as seen in figure 2. I modeled objects being tracked as a square-object. For the sake of the visual representation of the model, I used an icon of a human, but the shapes of the icon are not included in the model.

Figure 2: The top figure shows the comparison between the beam shape specified in the datasheet [5] of the sensor from the manufacturer and the model of the sensor used in my simulations. The bottom figure shows the image I used to model the object being tracked in the simulation. The simulation actually just considers the bounding box, the image is just for looks.

3.2 Algorithm

The flexible nature of the problem allows many configurations of sensors and room shapes. Each of these configurations also have a corresponding algorithm which takes as input the sensor readings, and a knowledge of the layout, and outputs the prediction of the location of the object on the map. I studied the configurations and corresponding algorithms for three different sensor layouts in a square room. The first layout is included in figure 3.

This layout has sensors placed one beam-width apart along the left side of the wall facing the opposite wall. The beams are assumed to reach across the room the other wall. The corresponding algorithm is show in figure 4.
Based on initial simulations, I noticed that layout 1 had a high X precision, down to the inch as provided by the sensors, but lacked Y precision. That is, the location of the object anywhere within one sensor beam was forced to take the same Y value since it cannot be differentiated any further. In order to address this problem, I created layout 2 shown in figure 5. The corresponding algorithm is shown in figure 6 right below figure 5.

Layout 2 attempts to address the loss of Y precision by placing the sensors closer together, thus theoretically giving two times the precision along the Y axis. The algorithm is accordingly modified to account for much more frequent sensor beam crossings, and the more crowded placement of the sensors.

Finally, another intuitive sensor placement was to line two perpendicular walls with sensors spaced one beam-width apart. This is essentially building a grid over the map using the distance measurements. Figures 7 and 8 show this layout and the corresponding algorithm.
Figure 5: Layout 2. The sensors are placed closer together in this layout. Specifically, they are 1/2 a beam width apart.

Input: $S_1, S_2, \ldots, S_n$ # sensor readings

active_sensors = \{S_i \mid S_i < \text{distance to wall}\}

$X = \text{average} \{S_i \mid S_i \in \text{active_sensors}\}$

$Y = \text{average} \{(i + 1) \times \text{beamwidth}/2 \mid S_i \in \text{active_sensors}\}$

Figure 6: Algorithm for layout 2.

3.3 Physical setup

In order to validate and inform the simulation, I purchased a Maxbotix Ultrasonic Rangefinder LV-EZ1 [7] and an Arduino Uno [8] microprocessor. I connected the rangefinder to the Arduino which was connected to a computer via a USB cable. The Arduino read the analog signal from the sensor and displayed the value in a computer. The details of experimentation with this physical setup is included in the Physical Experiment section.

4 Evaluation

At each point in the map, the predicted xy coordinates can be compared to the actual xy coordinates to get the error in the prediction. This error measurement can be done exhaustively for the entire map by placing the object in every possible xy location, and calculating the error. I calculated and visualized the exhaustive error in what I called an error map. Each point in the error map is a color ranging from white to red. The closer to red indicates a higher prediction error. I created the error map for each layout/algorithm combination in order
Input: SX_1, SX_2, ... , SX_n; SY_1, SY_2, ... , SY_n
active_sensors_x = {SX_i | SX_i < distance to wall}
active_sensors_y = {SY_i | SY_i < distance to wall}
X = average {SX_i | SX_i in active_sensors_x}
Y = average {SY_i | SY_i in active_sensors_y}

Figure 7: Layout 3.

Figure 8: Algorithm for layout 3.

to compare their effectiveness and limitations. Figure 9 shows the error maps for each of the layouts described above.

The error map for layout 1 shows a gradient of error that changes vertically but not horizontally. The change only happens in vertical change because the horizontal location detection is very precise and not subject to error. It is the vertical location prediction that has a loss in precision due to the layout of the sensors and therefore contributes to the error. The gradient occurs because regardless of where the object is within the beam, it can only be assigned one vertical value. If the object intersects two beams, it can only be assigned one vertical value, in this case the chosen value was the average vertical position of the two sensors. Therefore, as it gets closer to the actual sensor’s placement, this estimate of the vertical position becomes more and more accurate, hence the gradient.

Surprisingly, the error map for layout 2 shows much higher error measurements than for layout 1. This is due to the naive algorithm of averaging all intersecting beams vertical positions when calculating the vertical prediction. Perhaps an alternative would be to only average the vertical positions of the middle intersection sensor beams, and ignore the outside ones. It is unclear whether this algorithm would yield a more accurate result than for layout 1, and warrants
further analysis and simulation.

Unsurprisingly for layout 3, the error map is almost entirely white, denoting almost no error throughout. This is because the grid-like layout is basically measuring the vertical and horizontal offset of the object from the walls, which is basically directly calculating its position. This solution seems to solve the problem of one-object detection in an obstacle-free environment, however has the major drawback of requiring a large amount of sensors. Additionally, within each cell of the grid, it loses some accuracy because if an object is in two places inside one cell, the two locations cannot be differentiated.

In all the error maps, there are triangular areas of high error placed roughly around the location of the sensors. This is caused by the blind spot of the sensor beams. As show in the model diagram above, the sensor beams are roughly half-ellipsoid shaped and have some area near the start of the beam where they cannot sense objects. It is in these pockets where the high error occurs.

5 Physical Experiment

The arrangement of the sensors and the stage of the experiment is shown in figure 10.

Due to limitations in time and parts, I used one ultrasonic rangefinder to take multiple measurements from multiple locations to simulate using multiple sensors. With a real set up of multiple rangefinders, the measurements would be spaced apart slightly to avoid interference between each other, so this method of simulation isn’t inaccurate.

I first took measurements with no obstruction to determine the distance to
Figure 10: This figure shows the setup of the physical experiment. This entire diagram is a rectangular space where I conducted the experiments. The shaded boxes on the right are where I placed the ultrasonic rangefinder and took measurements. They are facing the left side of the diagram. I placed the object in different locations intercepting the beam inside the room.
<table>
<thead>
<tr>
<th>Actual location</th>
<th>Sensor reading</th>
<th>Predicted location</th>
<th>Error in centimeters</th>
</tr>
</thead>
<tbody>
<tr>
<td>N/A</td>
<td>235, 235, 235</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>76 165</td>
<td>235 152 154</td>
<td>82 166.5</td>
<td>6.18</td>
</tr>
<tr>
<td>75 30</td>
<td>152 235 235</td>
<td>83 72</td>
<td>45.58</td>
</tr>
<tr>
<td>225 88</td>
<td>235 235 235</td>
<td>Not present</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Table 1: actual XY, sensor readings, predicted XY, error – 3 rows of this. The sensor readings are in centimeters. The locations are offset from the upper left corner with the horizontal offset listed first.

The assumptions made in the simulation and the experiment have several serious limitations. First, they assume the room in which the detection is taking place is free of obstacles. In reality, obstacles are abundant and may interfere with the measurements. To address this limitation, the detection algorithm can happen in two phases. Suppose the goal is to track humans in a room. The first phase is detecting the background room structure by taking measurements when there are no humans around. This can be used to build a memory of the shape of the room as measured by the rangefinder. I also measured the actual distance and used that to calibrate the sensor. Afterwards, I placed an object, a round pot with a plant measuring 30 cm in diameter, in three different locations and recorded the measurements. Given these measurements, I used the algorithm developed for layout 1 to calculate the position prediction of the object from the sensor readings. The predicted and actual location, the error, and the sensor readings are shown in table 1.

The three locations were chosen for three different scenarios in the algorithm. The first row in table 1 is just the background measurement. The second row is when an object intersects two sensor beams. The third row is when the object fits entirely inside one beam, and finally the last measurement is if the object is in the error pocket caused by the blind spot of the sensors. The sensor readings and the predicted location reflect the results found in the simulation section.

There was a large amount of noise in the sensor readings. In choosing a number to use for the readings, I had to wait a few seconds to see which number occurred most often. This may be caused because of the noise in the ultrasonic beam. The sensor takes as a measurement whichever portion of the beam reflects back first. Dust particles in the air may cause the beam to reflect incorrectly before reflecting off an object. Other factors such as ambient noises or interference from other bands may also register on the sensor as a beam reflection. In order to choose the right value, some amount of smoothing must be used.

6 Discussion

The assumptions made in the simulation and the experiment have several serious limitations. First, they assume the room in which the detection is taking place is free of obstacles. In reality, obstacles are abundant and may interfere with the measurements. To address this limitation, the detection algorithm can happen in two phases. Suppose the goal is to track humans in a room. The first phase is detecting the background room structure by taking measurements when there are no humans around. This can be used to build a memory of the shape of the room.
room. The second phase is the same as before – treating the obstacles simple as part of the boundary and detecting any obstructions to ultrasonic beams.

Second, they assume perfect readings from the sensors. In reality, the ultrasonic beams may be interrupted by many sources and cause bad readings. For instance, if not timed properly, the sound waves from different sensors in the same setup may interfere with each other and give incorrect readings. Another problem of interference is wireless RF communication, such as with XBee radios [6]. In order to solve this problem, some intelligence needs to be implemented at each wireless node in processing the data. For example, averaging the readings over a few readings can give a more accurate measurement. Since the readings happen at a fast rate, taking multiple and averaging is a reasonable solution.

7 Future Work

Future work can address the limitations in the work, and make several additions to improve the accuracy of the simulation. First, a main improvement to the work is multiple object tracking. Multi-object tracking can be trivially added to the algorithms and layouts described above by detecting if there are clearly disjoint measurements. If that is the case, it cannot be caused by the same object and therefore must be the result of multiple objects. This suggests a more robust algorithm capable of dealing with multiple objects. The first step of the algorithm can be determining how many of these disjoint measurements there are, which are created by multiple objects. After determining the number, the algorithm can then assign the best possible prediction to each of these disjoint sensor readings.

A second improvement is to accommodate complex maps, beyond the simple square or rectangular rooms studied in this project. Clearly in reality, most of us do not live in square-like living arrangements; homes have walls, rooms, and corridors. The sensor layouts and corresponding algorithms have to be expanded to accommodate these.

Finally, another improvement for future work is more physical experimentation. Due to limitations in time and parts, I could only implement and test a very simple physical experiment. However, I would like to develop independent motes using XBee radios and ultrasonic rangefinders, which then communicate the information to a central computer that does the computation. In addition, I would like to acquire more sensors and Arduinos so that I do not have to use the same one to simulate taking measurements from multiple sensors.
8 Conclusion

In this project, I’ve introduced the idea of object location detection using a wireless sensor network of ultrasonic rangefinders. The key contribution of this project is achieving object localization using cheap sensors, and non-invasive techniques. I presented a simulator for rapid prototyping of layouts and algorithms of sensors. I also presented three sensor layouts and their corresponding algorithms for object localization, and compared the three using error maps.

References


