A Mobile Sensor Platform Approach to Sensing and Mapping Pervasive Spaces and Their Contents

Hicham El-Zabadani, Abdelsalam (Sumi) Helal and Hen-I Yang
Mobile & Pervasive Computing Laboratory, CISE Department
University of Florida, Gainesville, FL 32611, USA
http://www.icta.ufl.edu
{hme, helal, hyang}@cise.ufl.edu

Abstract

We propose a novel approach to mapping and sensing smart spaces in which a mobile platform equipped with a range of sensors and actuators is used. The Mobile platform is called SensoBot and uses on-board RFID modules to locate RFID tags embedded in the carpet in the form of a grid. While driving around, SensoBot writes to the tags the corresponding coordinates which can be used later to locate specific objects. In addition to mapping, SensoBot is used to identify and locate important landmarks and objects like doors, windows, and furniture. The goal of this project is to achieve a complete self sensing space where automatic floor creation, appliance control, and content identification and location are performed.

Keywords— Pervasive Computing; Ubiquitous Computing; SLAM; Space mapping; Location tracking; Sensor platform; RFID.

I. Introduction

Smart Spaces are inhabited by many cooperating agents of different types. In addition to people, there are likely to be fixed sensors embedded in the environment and mobile agents such as robots [1]. Our vision is to create new capabilities with which smart spaces such as homes can sense themselves and their residents; create the respective real world models; and enact an accurate mapping between the elements in the real world model and the physical world. Once realized, this vision will enable many applications that rely on remote monitoring and intervention. By enabling spaces to automatically detect their main aspects (e.g., floor plans, type of rooms etc.), and to automatically identify the objects (e.g., furniture pieces, appliances, etc.),within, it will be possible

to create on-the-fly, incremental, world models of the pervasive space [2, 3].

To achieve this vision, we have previously designed the Smart Plug concept to locate, track, and remotely interact with appliances and electrical devices [4]. In this paper, we use a robot equipped with various sensors and actuators to enable automatic space mapping and content sensing. The goal is to design a mobile platform capable of automatically creating a floor plan of a space, it should also be able to locate and identify important landmarks and objects in that space.

Simultaneous localization and mapping (SLAM) is a technique used by robots and autonomous vehicles to build up a map within an unknown environment while keeping track of its current position at the same time [5, 6]. This is not as straightforward as it might sound due to inherent uncertainties in discerning the robot's relative movement from its various sensors.

During the map building process, if the measurement of distance and direction traveled include any inaccuracy, then all features added to the map will contain corresponding errors. If unchecked, these positional errors will accumulate, resulting in grossly distorting the map and hinder the robot's ability to detect its precise location. Various techniques can be employed to compensate for this, such as recognizing features that it has come across previously and re-skewing recent parts of the map to considate the two instances of the same feature. Some of the statistical techniques used in SLAM include Kalman filters, particle filters (aka. Monte Carlo methods) and scan matching of range data [7]. In this paper we use RFID technology to eliminate errors produced while building the map. The idea is to have a grid of RFID tags embedded in the carpet. The tags initially hold only the serial number, and the corresponding coordinates are filled in once visited by the robot. More details on the techniques used to achieve this goal will be presented later in this paper. Another important task is to locate important landmarks and objects like furniture. The robot will also use RFID tags embedded in those objects to perform the identification and location process.

We briefly discuss the specifications of each component used in the mobile platform in section

2. Section 3 explains the potential applications of SensoBot. In Section 4 and 5, we present the algorithms used, and the expreimental and simultation results of the algorithms. Finally, section 6 concludes the paper.

II. SensoBot: The Mobile Sensing Platform

In this project we needed to build a robot that will carry our sensing platform around. The robot's main job is to perform simple actions like move forward, turn 90 degrees clockwise, etc. Instead of building this robot from scratch, we used IRobot's famous vacuum cleaner Roomba to perform this task [8]. The reason why we chose Roomba is the fact that it allows for external control using its Serial Command Interface (SCI) [9]. In addition, we used two RFID readers and one digital compass all controlled by the Atlas sensor platform [10].

III. Applications

There are two main goals in this project. The first is to automatically create a two-dimensional floor plan of the space using SensoBot. The second is to use SensoBot to locate furniture and other important landmarks based on the floor plan that has been created.

A. Creating Floor Plan

Automatic creation of a floor plan is not an easy task for a robot, as SLAM is considered to be a fundamental problem in robotics. But its success is essential for safe and efficient navigation of intelligent wheelchair, home-helper or floor-cleaning robots, etc. Previous SLAM methods work successfully in stationary environments, but they fail in dynamic environments due to map uncertainty as well as environment changes [11]. To solve this problem, we will use RFID tags to create landmarks that SensoBot can use to create the corresponding floor plan.

We deployed passive, low-cost, High Frequency RFID tags in a grid under the flooring. A single

passive RFID tag represents a single grid point in the system. The RFID tags can be integrated by carpet manufacturers as part of the weaving process or be integrated into a thin layer of material that can be applied under the carpet or hard surface flooring. Rooms that have existing carpeting could be easily upgraded by rolling up the carpet, applying the RFID flooring material and then reinstalling the existing carpet. Figure 1 shows the carpet tiles used in the Gator Tech Smart House [12]. The size of each tile is 20 square inches and each holds a tag in the middle. Initially, each tag only holds a unique serial number assigned by the manufacturer. SensoBot is responsible to fill the corresponding data for each tag. Starting from a specific location in the space, SensoBot will start moving until it finds the first tag. This tag will be appointed as the origin with coordinates [0,0]. The next step is to find a neighbor of this tag. Two tiles are neighbors if they share the same side. After locating one neighbor, SensoBot continues to look for all other neighbors of the original tag until all tags in the space have been located. The details of the algorithm used to find the neighbors are presented later in this paper.

B. Locating Important Landmarks

In the previous section we discussed how SensoBot can produce a 2D floor plan of the space. However, such an operation is time consuming because SensoBot runs on batteries and needs to be recharged from time to time. The operation is also not error-free. There will be a certain amount of error that needs to be fixed over time.

The floor plan created will be used to locate important landmarks and objects in the space. Landmarks are fixed and cannot be moved from one place to another, such as doors, windows, and power outlets. Objects we are interested in locating consist mainly of furniture. Recall that SensoBot has two RFID readers. The one pointing forward is used to complete this task. The idea is to place RFID tags on the wall base below the landmarks of interest. For example, Figure 2

shows two RFID tags located on the wall base of a door, one on the right side and the other on the left side of the door. When SensoBot reads the tags, it knows that which one is the door side and which one is the opening side. Using the RFID tags in the carpet, SensoBot can identify its approximate location, hence identifying the location of the door.

Locating objects is similar, if a piece of furniture has an RFID tag located somewhere accessible by SensoBot, this piece of furniture can be located. For example, a four legs table could have four RFID tags describing the table, each on one leg. Therefore locating two legs would allow SensoBot to identify the table's approximate location.

SensoBot can locate different landmarks using the format shown in table 1. The first column describes the landmark and the second and third column describes the data stored on the first and second byte of the RFID tag respectively. In the case of furniture, we need information more than just the name. That is why we created a set of rules that can be used to get a better description. By reading a tag with the value 'FF' in the first byte, SensoBot knows that this is an object. It will refer to table 2 to find out the category of that object, which can be found in the first byte of block 1. The length, width, and color can be found in bytes 1, 2, and 3 respectively.

IV. Algorithms

In the previous section we discussed the two applications that SensoBot is expected to perform. The first one is creating a floor plan of the space. The second application is finding important landmarks. We present next the detailed algorithms employed in SensoBot to guarantee a successful outcome in this section.

Choosing the best algorithm is not easy because we are not dealing with pure software algorithms anymore. Many external factors, such as time to complete, power consumed, and maximum area covered, all have great impact on the efficiency of the algorithms. We will use the

term tag to represent a tile in the algorithms discussed later. As shown in figure 3, each tag has four adjacent tags known as neighbors. Let us assume that all tags are initially marked white. A tag is marked gray if it is visited but not all neighbors are visited yet. A tag becomes black when all its neighbors have been visited. We start by discussing a naive algorithm in the following section. Later we will discuss some improvement made to improve the results.

1) Naive Space Mapping Algorithm

The first algorithm is based on the Breadth First Traversal algorithm. We will call it Breadth First Mapping (BFM). We assume in BFM that SensoBot is able to get from one tag to another without errors. We start at a tag t and mark it as having been reached. The tag t is at this time said to be unexplored (gray). A tag is said to have been explored when the algorithm has visited all tags adjacent from it (black). All unvisited tags adjacent from t are visited next. These are new unexplored tags. Tag t has now been explored. The newly visited tags haven't been explored and are put onto the end of a list of unexplored tags. The first tag on this list is the next to be explored. Exploration continues until no unexplored tag is left. The list of unexplored tags operates as a queue and can be represented using any of the standard queue representations. Figure 3 shows the four adjacent tags (neighbors) of the origin in the middle. The numbers on each tile shows the sequence that the algorithm follows to traverse all the adjacent tags.

Although BFM will guarantee that SensoBot visits all tags, it is an expensive and time consuming one. The fact that this algorithm is based on the traditional breadth first search algorithm makes it time consuming and repetitive. SensoBot has to visit all neighbors of a tag before moving to the next one. If a tag has only one neighbor that is not visited yet, SensoBot has to visit all four tags instead of visiting the not-visited one alone. The second disadvantage in this algorithm is the fact that SensoBot runs on batteries. If, for example, this algorithm reaches level five, the distance that SensoBot has to travel back and forth could reach up to 20 feet.

It is obvious by now that we need a more efficient algorithm that can eliminate the repetition and minimize unnecessary traveling.

2) Improved Space Mapping Algorithm

To eliminate revisiting tags, we need to look at a more efficient algorithm. The new algorithm is based on the Depth First Search Traversal and we call it Depth First Mapping (DFM). In DFM, SensoBot finds the first tag and marks it as origin. Next it moves in one direction and pushes each visited tag into a stack. When blocked, it changes direction choosing the one that might lead outwards. If all three neighbor tags are either inaccessible or visited, SensoBot pops a tag from the stack and backtracks to that tag and tries to find an open path. SensoBot finishes when the stack is empty. DFM is faster and more efficient than BFM. This is due to the fact that SensoBot does not look for the neighbors of each tag before moving to the next one. On the contrary, it keeps on moving in one direction. By default, using this approach, SensoBot will discover the neighbors of most of the visited tags automatically.

3) More Improvements

SensoBot, as mentioned before, uses a digital compass to determine orientation. Digital compasses are able to give exact heading most of the times. However, under certain circumstances, the compass might give false headings when soft or hard-iron distortions are present. If lost, SensoBot simply starts looking for a tag; if the tag is already visited, (SensoBot knows from the tag's unique serial number) it resumes running the algorithm. However, if the tag is new (not visited before), SensoBot restarts the algorithm starting from the new tag as the origin. If encountered with a previously visited tag, SensoBot combines the two occupancy grids and continues filling the new combined grid.

The improved algorithm is called IDFM and is described in Algorithm 1. There is one extra improvement over DFM which saves more time during the backtracking process. In DFM,

SensoBot used to backtrack to the popped tag before checking for an open path from a specific tag. In IDFM, each time SensoBot pops a tag, it checks for an open path before driving to that tag. If it happens to be an explored tag (all neighbors are visited), it pops another tag. SensoBot repeats this process until it finds a tag that is not explored. At this moment, it backtracks to that tag. In small areas this change will not significantly improve the performance; however, in large areas, a big percentage of tags in the stack become explored before they get popped. Checking the status of those tags before driving to them saves a significant amount of time and power.

Algorithm 1 IDFM

- **Mark the current tag t visited, and push it to the stack**
- **While the stack is not empty or not lost**
  - **Drive to the next tag**
  - **If path is open**
    - **Push the tag to the stack and marked it visited. Increase S_T**
  - **Else if path is blocked**
    - **Check for open path in different direction.**
    - **If none exist, keep popping the stack until you find a tag that has an open path.**
    - **Backtrack to the tag found. Increase S_T**
- **If lost, save old data and restart. When old tag is found, merge two grids and continue.**

V. Performance Evaluation

We evaluated SensoBot's performance in the Gator Tech Smart House which is the perfect environment to do such evaluation. The fact that it is a real house with real furniture brings to this research accurate results that can be used later to decide the feasibility of this approach. All three algorithms, BFM, DFM, and IDFM, were used during the evaluation process.

We installed tags in the living and dining areas, which consist of about half the size of the

house. Figure 4 shows the dining area with part of the living room area. There are 207 carpet tiles in both areas, with one RFID tag in each tile. The size of the carpet tile is 20 inches. As shown in Figure 4, the yellow shaded area represents one carpet tile with one RFID tag in the middle. We removed all furniture from the two areas except for three items (two are shown in Figure 4). SensoBot started from one corner where the home base is located. The first tag it encountered was used as the origin ([1,1] coordinates).

Figure 5 shows the chart were results from all three algorithms were plotted. The pink line represents BFM which was able to cover the 207 tags in about 69 minutes. The green line represents DFM. Using this algorithm SensoBot was able to find all 207 tags in about 42 minutes. The brown line represents IDFM. SensoBot covered all 207 tags using this algorithm in about 26 minutes. IDFM did indeed improve the execution time from 69 minutes, which is BFM completion time, down to 26 minutes, the time taken by IDFM to cover the entire RFID grid. SensoBot uses WI-FI to send all results to a desktop application that keeps track of all tags found and draws the floor plan on the fly. The progress of mapping the living and dining areas is shown in Figure 6. Notice how SensoBot starts from the corner and uses the improved algorithm discussed earlier to draw the floor plan of the house. The result is shown in the bottom right part of Figure 6.

Figure 7 shows the resultant floor plan superimposed over the real floor plan of the house. Notice that some areas, especially on the borders, were not recognized by SensoBot. This is caused by either the furniture which blocked the way, or because the tile was too small to have a tag on it (border tiles). Later, in chapter 6, we will discuss how furniture can be located using SensoBot. The experimentation with different algorithms discussed above shows that the performance of the improved algorithm is indeed substantially superior. We next present a quantitative performance evaluation of this algorithm and show the effect of different

configurations of physical obstacles by running simulations. To quantitatively evaluate the performance of the improved algorithm, we first define L as the absolute lower bound of the time needed to cover and map the entire target area. For an area containing |V| tags, the lower bound L would be in O(|V|). Using L as the reference, we specify the performance factor ε to indicate that the improved algorithm can finish mapping within (1+ε) L. Next, we introduce two parameters B_p and B_d to characterize the extent and distribution of the obstacles in the space. B_p represents the percentage of tags blocked by objects, and B_d indicates how many separate chunks these objects are divided into.

Since the tags are arranged in grid, consecutive readings of tags require SensoBot to move a fixed distance. Therefore the time and energy needed to cover the entire target area with |V| tags can be estimated by the total number of times the tags have been visited before each single tag has been visited at least once. Assuming S_T is the number of times a tag T has been visited, then the absolute lower bound L occurs when every tag is visited exactly once, that is $S_T = 1$, $\forall_T$. We can then calculate ε as below,

$$\varepsilon(B_p, B_d) = \frac{\sum S_T}{|V|} - 1 \qquad (1)$$

For each simulation, we create an area containing 324 tags, and randomly generate and distribute the obstacles based on the pre-defined B_p and B_d. For each combination of B_p and B_d, we run a total of 1000 simulations. The value of B_p is set to be between 5% and 30%, and obstacles are divided (B_d) from single lump to 16 separate chunks. The reason for limiting B_p to 30% is based on the assumption that a typical room should have 70% of its floor reachable by SensoBot, including the area underneath some furniture objects. Figure 8 shows the results of the simulation. To further quantify the effects of B_p and B_d on ε, we perform linear two-variable surface fitting using MATLAB, which returns the following equation,

$$\varepsilon = 0.0765\ B_p + 0.4869\ B_d - 0.3437 \qquad (2)$$

The performance of the improved algorithm degrades when more tags are obstructed by objects or when the obstacles are scattered. The equation further indicates that the distribution of the obstacles has a much greater impact on the performance of the algorithm than how much area is obstructed. One item worth mentioning is that when running the SensoBot in a reasonable setting, the performance factor ε is always under 11% even at the worst case (B_p =30%, B_d =14), which shows the performance of the improved algorithm is reasonably good.

VI. Conclusion

In this paper we described a new approach to mapping and sensing spaces using an Atlas sensor platform. We called our mobile platform SensoBot. A collection of sensors were used to move SensoBot in an intelligent way and read RFID tags embedded in the carpet arranged in a grid. Several algorithms were proposed and tested and the results were analyzed. This paper presents the use of one mobile sensor platform that communicates with a desktop application. However, this approach may be further improved when deploying more than one mobile sensor platforms, are to work in parallel in ad-hoc mode. We are also investigating on the use of better algorithms that might produce error-free results. Finally, this work allows us to move one step closer to achieving a complete self sensing space where a space can sense itself and its contents, keep track of changes, and report to a remote monitoring application when emergencies occurs.

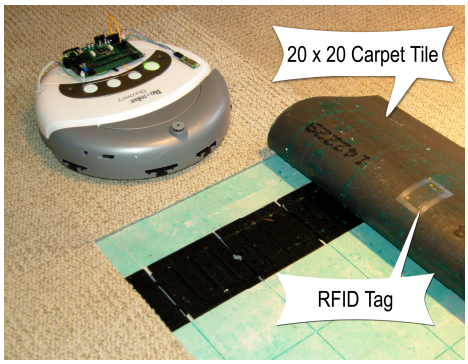

Figure 1. RFID tag in the middle of a carpet tile

Figure 2. RFID tag describing a door

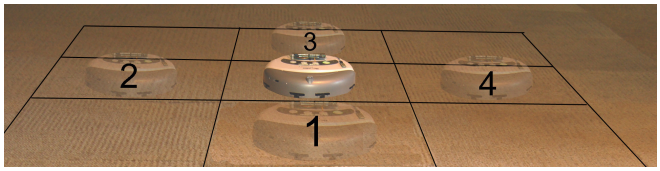

Figure 3. Finding the 4 neighbors of each tag

Table 1. Landmark Identification

Landmark	**Block 0, First Byte**	**Block 0, Second Byte**
Door (Opening Side)	00000001	00000001
Door (Fixed Side)	00000001	00000000
Front Door (Opening Side)	10000001	00000001
Front Door (Fixed Side)	10000001	00000000
Window (Left)	00000010	00000001
Window (Right)	00000010	00000000
Power Outlet	00000011	-
Closet (Left)	00000100	00000001
Closet (Right)	00000100	00000000
Dynamic Landmark	11111111	-
…	…	…

Table 2. Dynamic Landmark Identification

Name	**Address**
Category	Block 1, byte 0
Length	Block 1, byte 1
Width	Block 1, byte 2
Color	Block 1, byte 3

Figure 4. RFID grid in the GTSH

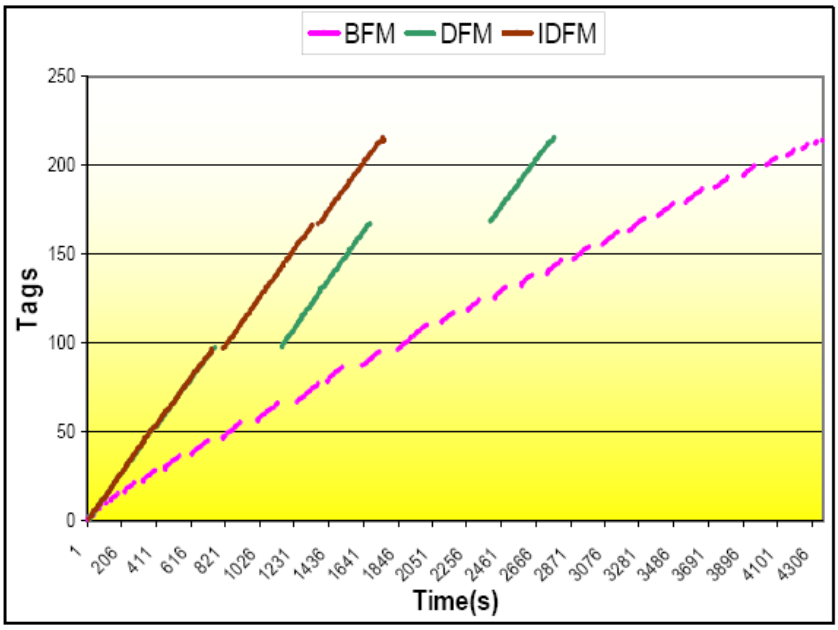

Figure 5. Test Results

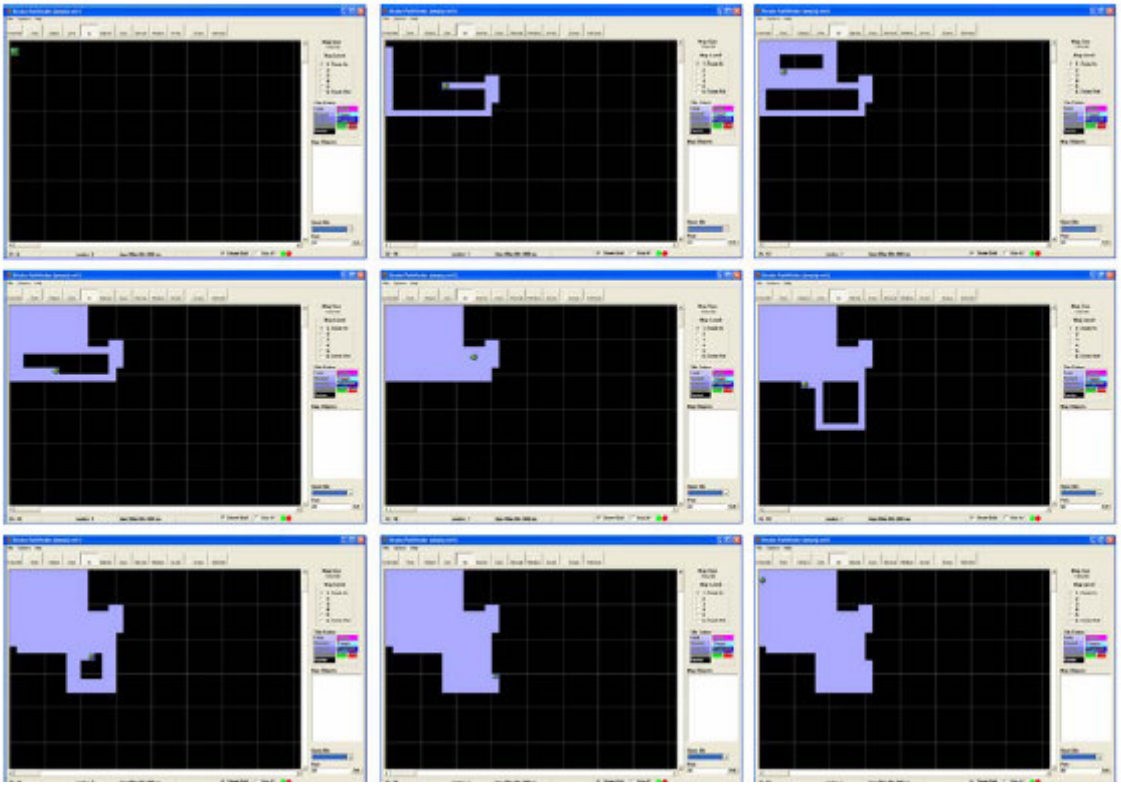

Figure 6. Floor Plan Created

Figure 7. Floor plan produced superimposed over the real floor plan of the GTSH

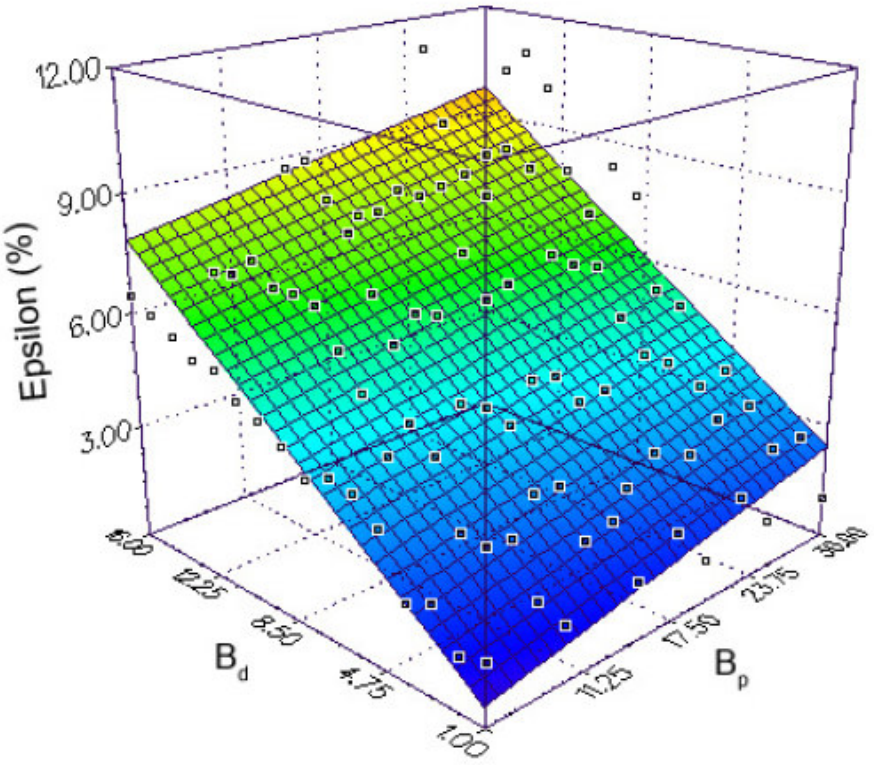

Figure 8. 3-D plots of the effects of Bp and Bd on ε

Reference

[1] R. Want, "Enabling Ubiquitous Sensing with RFID," *Computer, IEEE*, vol. 37, pp. 84-86, 2004.

[2] S. S. Intille, J. W. Davis, and A. F. Bobick, "Real-Time Closed-World Tracking," presented at 1997 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1997.

[3] Y. Kaddoura, A. Helal, and J. King, "Cost-Precision Tradeoffs in Unencumbered Floor-Based Indoor Location Tracking," presented at Third International Conference On Smart homes and health Telematic (ICOST), Sherbrooke, Canada, 2005.

[4] H. Elzabadani, A. Helal, B. Abdulrazak, and E. Jansen, "Self-Sensing Spaces: Smart Plugs for Smart Environments," presented at Third International Conference On Smart homes

and health Telematic (ICOST), Sherbrooke, Canada, 2005.

[5] G. Dissanayake, P. Newman, S. Clark, H. Durrant-Whyte, and M. Csorba., "A solution to the simultaneous localisation and map building (slam) problem," *IEEE Transactions of Robotics and Automation*, vol. 17, pp. 229-241, 2001.

[6] M. Montemerlo and S. Thrun, "Simultaneous localization and mapping with unknown data association using FastSLAM," *IEEE International Conference on Robotics and Automation*, vol. 2, 2003.

[7] D. Hähnel, R. Triebel, W. Burgard, and S. Thrun, "Map Building with Mobile Robots in Dynamic Environments," presented at International Conference on Robotics and Automation, 2003.

[8] "Roomba," in *http://www.irobot.com*.

[9] "iRobot Roomba Serial Command Interface (SCI) Specification," in *http://www.irobot.com/images/consumer/hacker/Roomba_SCI_Spec_Manual.pdf*.

[10] R. Bose, J. King, H. El-zabadani, S. Pickles, and A. Helal, "Building Plug-and-Play Smart Homes Using the Atlas Platform," presented at the 4th International Conference on Smart Homes and Health Telematic, 2006.

[11] T. Kanji, H. Tsutomu, Z. Hongbin, K. Eiji, and O. Nobuhiro, "Mobile Robot Localization with an Incomplete Map in Non-Stationary Environments," presented at the 2003 IEEE International Conference on Robotics and Automation, 2003.

[12] A. Helal, W. Mann, H. Elzabadani, Y. Kaddoura, E. Jansen, and J. King, "Gator Tech Smart House: A Programmable Pervasive Space," *IEEE Computer magazine*, pp. 64-74, 2005.