GEOLOCATION PREDICTION USING MACHINE LEARNING

Rohit Sharma, Paripath Inc

CONTENTS

INTRODUCTION

Localization is the task of determining the physical coordinates of a sensor node (or a group of sensor nodes) or the spatial relationships among objects. It comprises a set of techniques and mechanisms that allow a sensor to estimate its own location based on information gathered from the sensor's environment. While the Global Positioning System (GPS) is undoubtedly the most well-known location-sensing system, it is not accessible in all environments (e.g., indoors or under dense foliage) and may incur resource costs unacceptable for resource-constrained wireless sensor networks (WSNs). [1],[2]

In this work, we used WinProp™ software suite are provided by AWE Communications for extracting geo-spatial information from simulation. The ProMan Propagation software package is designed to predict the path loss accurately between transmitter and receiver including all important parameters of the mobile radio channel. The ProMan (Network) software offers network planning modules for 2G/2.5G, 3G/B3G, WLAN, WiMAX networks. It offers Static network planning modules as well as dynamic network simulators

We propose nonlinear polynomial regression as a machine learning method to obtain excellent results on predicting emitter location.

SCENARIO DESCRIPTION

We received ray data in an ASCII (*.str format) to evaluate characteristics of radio channel. We extracted channel's impulse response from this data to create dataset for machine learning application. Ray data format was extracted from following two statements given below:

```
POINT 396.00 580.00 515.46

PATH 2349.565300 58.71 0 1 414.0000 623.9900 539.3000 D 423 1
```

Figure: impulse response of channel in ASCII format

In the format shown above, statement `POINT` is nothing but geolocation coordinate of emitter location. First two fields in the second statement `PATH` is delay of path (in ns) and field strength (in dBµV/m). We used these two statements to create dataset for our machine learning application by parsing rat data file.

MACHINE LEARNING IN EMITTER LOCALIZATION

Localization research has garnered a good deal of interest from both academia and industry, with numerous systems being proposed using a variety of technologies. A major disadvantage of many of these systems—such as infrared [3], ultrasound [4], and RFID [5], [6]—is that they require dedicated sensors and substantial infrastructure changes and as a result, incur a significant cost to deploy. Effort has been made to devise localization systems that require little to no infrastructure change using Bluetooth [7], [8] and WiFi signal strengths [9], [10], [11], [12] with some success. The systems developed using WiFi signal

Delay(ns)	Strength(dBuV/m)	X-coordinate	Y-coordinate	Z-coordinate
2349.565	58.71	396.00	580.00	515.46
926.787	74.92	736.00	980.00	515.06
......	.....	.....	......	......
......	.....	.....	......	......
808.820	76.04	756.00	980.00	517.05

Figure: Dataset adaption using parser written in python

strengths for localization show promise but have yet to receive widespread adoption. These systems can be divided into two categories: those using a fingerprinting approach using algorithms for "nearest neighbor in signal space" and those using more complex signal propagation algorithms to determine a device's distance from the access points in range.

DATA ANALYSIS

Once the data was prepared, several data analysis and data visualization techniques were used as a basis for debate to come up with a suitable machine learning model. Parallel Plot is a popular visualization technique used to plot individual data elements across many dimensions. This gives us a sense of relationship between path delay, field strength and their coordinates.

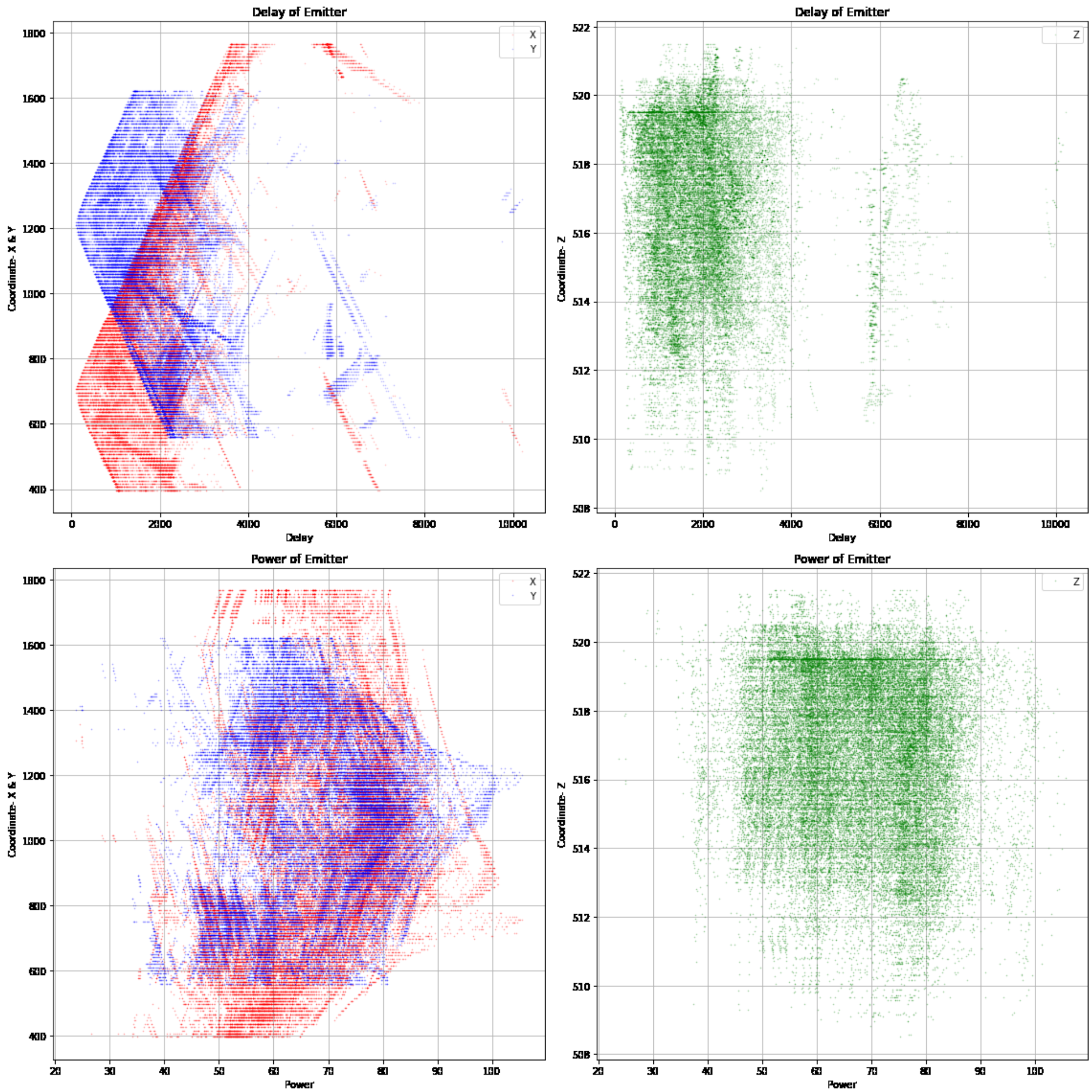

Figure: Parallel Plot Visualization for Path Delay and Field Strength

Figure: Parallel Plot Visualization for Path Delay and Field Strength shows parallel plots for following 6 relations.

1. X coordinate and path delay
2. Y coordinate and path delay

3. Z coordinate and path delay
4. X coordinate and field strength
5. Y-coordinate and field strength
6. Z-coordinate and field strength

There were several other plots that were visualized in an effort to find data relationship that can be harnessed in the choices of machine learning models. Some of the other plot (not shown here, will be delivered along with code) were 3D scatter plots. Euclidean distance plots and Power Delay Profile of the dataset.

DATA MODEL

Based on the visualization results, we came to conclusion that regression using single kernel was most appropriate model for the localization task. Mathematical equation below describes final regression model used in the delivery.

$$X_{pred} = \sqrt[2]{dw_x.delay^2 + pw_x.(power^2 - power) + b_x}$$

$$Y_{pred} = \sqrt[2]{dw_y.delay^2 + pw_y.(power^2 - power) + b_y}$$

$$Z_{pred} = \sqrt[2]{dw_z.delay^2 + pw_z.(power^2 - power) + b_z}$$

This model gave us a total of 9 model parameters to train enumerated below:

1. dw_x, pw_x, b_x
2. dw_y, pw_y, b_y
3. dw_z, pw_z, b_z

MACHINE LEARNING FRAMEWORK

We used set of tools as machine learning framework to develop and test the application

1. Jupyter Notebook version 5.1.0 (http://jupyter.org)
2. Python version 3.6.3 (https://www.python.org)
3. Python numpy version 1.13.3 (http://www.numpy.org)
4. Tensorflow version 1.4.0 (http://tensorflow.org)
5. CentOS version 7.4.1708 (https://www.centos.org) in Google cloud (http://cloud.google.com)

DATA CONDITIONING

Later analysis revealed that adapting the raw data obtained from simulation to machine leaning framework wasn't enough, since it performed poorly on training and accuracy expectation. We performed following operation to adapt the data to our framework.

1. Randomize
2. Normalize
3. Divide for training and validation

TRAINING AND MODEL GENERATION

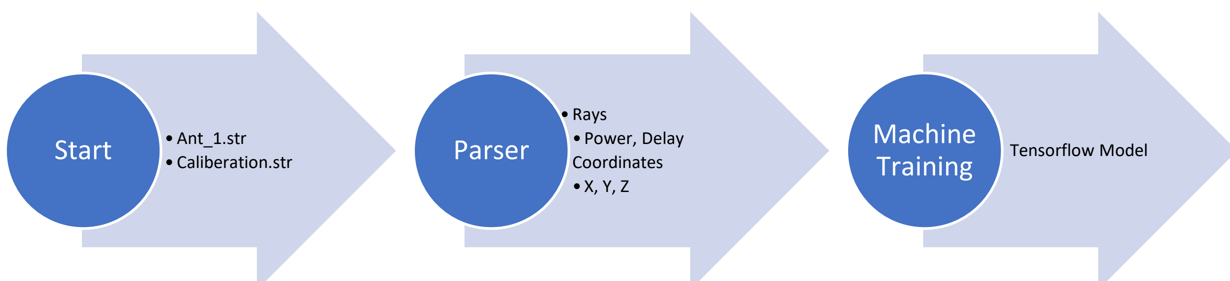

Figure: Training and Model Generation for Predicting Emitter Location in WSN

Figure on Training and Model Generation for Predicting Emitter Location in WSN shows software control flow. We used square mean error as a loss function with gradient descent optimization method to give us model parameters with best possible accuracy for the given dataset. We trained the model for 1000 epochs to achieve acceptable level of accuracy. At the end of 1000th iteration, mean square error was 0.142. Model parameters were saved in a file model_param.py ready for use in generated regression model. The model is supplied with model.py

USING THE MODEL

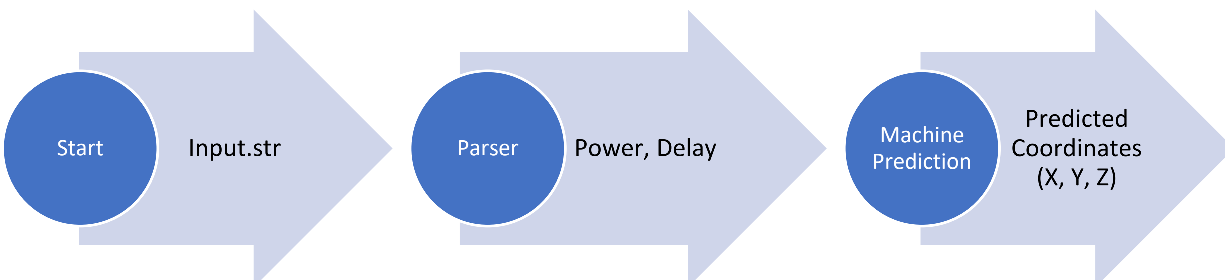

Figure: Using Regression Model with new dataset

Figure 'using the regression model with new dataset' shows the data flow for testing the dataset. Remember to limit the new test dataset with a range (X, Y, Z, path delay and field strength) similar to training dataset to continue getting expected accuracy. If the new test dataset is beyond the range, remember to train the model before using it.

RESULTS

With regression model, we could bring the prediction error to under 2% as shown below:

1. Training   error mean= 1.82759938221 %
2. Validation error mean= 1.7151372943 %

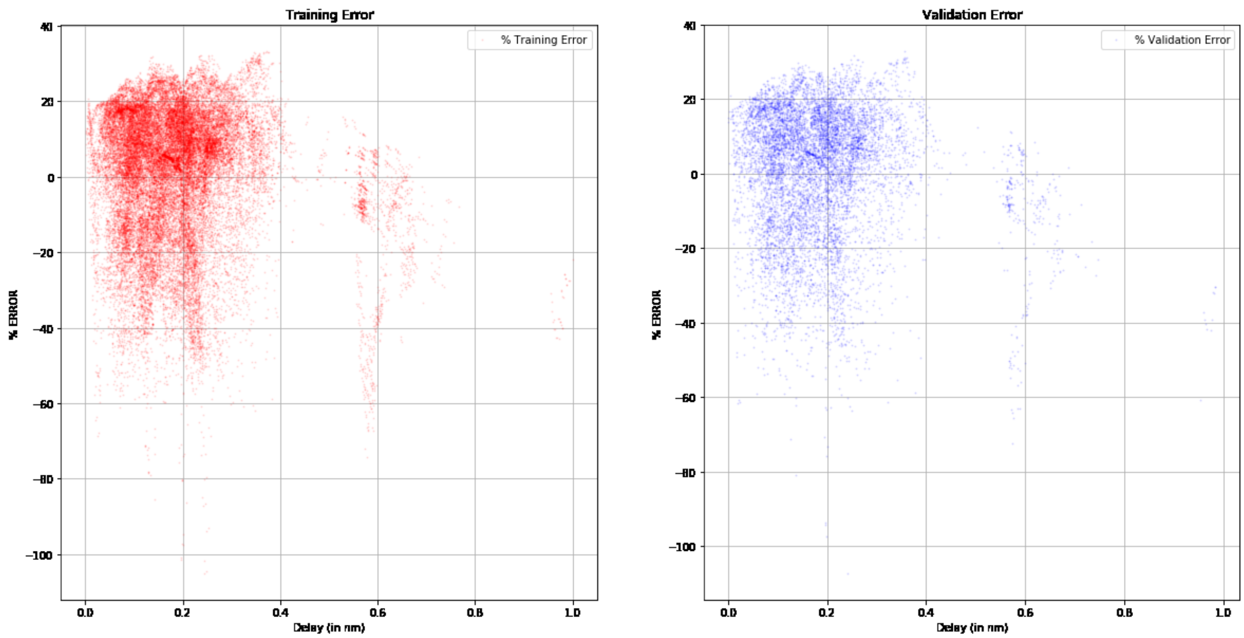

Figure: Error Distribution Plot

These numbers may differ slightly depending on the permutation of randomization used before training. They were always observed under 2%, which is well below accepted range of 5%. Figure 'Error Distribution Plot' shows training and validation error.

SETUP

To make the setup easy, following python files will be delivered.

1. Generating+Model+for+Predicting+Geo+Location+of+Emitters.py
2. Using+Model+for+Predicting+Geo+Location+of+Emitters.py
3. model_parms.py

Remember to install python with numpy and tensorflow libraries with versions mentioned in Machine Learning Framework section.

Once installation requirements are met, you can run it on command line. Our development environment was in google cloud. We downloaded our scripts on a windows-10 computer to check for a cross platform test. A typical session of training and test on windows10 is shown below:

```
Windows PowerShell

PS C:\Users\srohit\Downloads\fiverr> python.exe ..\Generating+Model+for+Predicting+Geo+Location+of+Emitters.py .\Ant_1.s
tr
Min= 9.67797344297e-05 Max= 1.00009677973 Mean= 0.185761421955 sigma= 0.111279072513
Min= 0.00290535583272 Max= 1.00290535583 Mean= 0.538774195008 sigma= 0.144676403079
Min= 0.00289051094891 Max= 1.00289051095 Mean= 0.40914726192 sigma= 0.244345989001
Min= 0.00528301886792 Max= 1.00528301887 Mean= 0.499474515836 sigma= 0.253718911403
Min= 0.391153846154 Max= 1.39115384615 Mean= 1.01887830531 sigma= 0.178626027054
Dataset size (46391, 5)
2018-01-26 11:41:11.508990: I C:\tf_jenkins\home\workspace\rel-win\M\windows\PY\36\tensorflow\core\platform\cpu_feature_
guard.cc:137] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX
Epoch: 0100 cost= 0.155911833
Epoch: 0200 cost= 0.154088929
Epoch: 0300 cost= 0.152468234
Epoch: 0400 cost= 0.151023477
Epoch: 0500 cost= 0.149738297
Epoch: 0600 cost= 0.148600176
Epoch: 0700 cost= 0.147602245
Epoch: 0800 cost= 0.146741197
Epoch: 0900 cost= 0.146024063
Epoch: 1000 cost= 0.145483941
Optimization Finished!
1.11095 -0.256167 0.79049 0.483389 0.31717 0.789653 0.224165 0.335358 1.18359
3056.4 74.7 : [  706.    1270.     518.99]  =>  1013.967 1072.746 516.839
2320.2 66.0 : [ 1166.    1040.     515.39]  =>  935.222 1077.795 516.574
773.0 63.5 : [  726.   1430.    516.6]  =>  842.993 1089.635 516.349
1604.8 76.4 : [ 1036.     980.     519.07]  =>  901.653 1091.939 516.549
1801.5 54.1 : [ 1186.    1350.     512.16]  =>  909.223 1088.404 516.551
1045.7 60.7 : [  716.    1340.     518.03]  =>  854.817 1089.119 516.381
2198.5 76.2 : [ 1186.     810.     520.97]  =>  942.417 1086.120 516.652
1217.7 52.1 : [  456.    1480.     516.41]  =>  882.170 1095.058 516.505
2412.6 52.6 : [ 596.    880.    518.09]  =>  959.673 1083.618 516.699
1616.4 60.2 : [ 506.    770.    518.93]  =>  885.346 1085.574 516.454
Training   error mean= 1.29047480921 %
Validation error mean= 1.25610464433 %
PS C:\Users\srohit\Downloads\fiverr>
```

Figure: Training Regression Model with ASCII Ray Data file.

And the next figure shows typical session of using the model on a different ray data file for prediction

```
Windows PowerShell
PS C:\Users\srohit\Downloads\fiverr> python.exe ..\Using+Model+for+Predicting+Geo+Location+of+Emitters.py .\Calib_1.str
Dataset properties:
Dataset size (784, 5)
Min= 0.00127944267486 Max= 1.00127944267 Mean= 0.179466008977 sigma= 0.18481504455
Min= 0.00622068024842 Max= 1.00622068025 Mean= 0.62657493406 sigma= 0.144190847335
Min= 0.0312525560868 Max= 1.03125255609 Mean= 0.47824121045 sigma= 0.186312056997
Min= 0.00469889924264 Max= 1.00469889924 Mean= 0.403156520141 sigma= 0.2641023245
Min= 0.0262294279157 Max= 1.02622942792 Mean= 0.586458040488 sigma= 0.190530368099
# Path_Delay Field Strength : [X Y X] => predicted_X predicted_Y predicted_Z
2299.2 58.7 : [ 426.381  587.823  572.276]  =>  [ 937.557 1079.633 516.593]
2537.2 53.5 : [ 426.381  587.823  572.276]  =>  [ 968.038 1081.017 516.712]
2325.7 52.1 : [ 426.381  587.823  572.276]  =>  [ 954.106 1085.236 516.688]
2444.7 52.0 : [ 426.381  587.823  572.276]  =>  [ 964.236 1083.928 516.716]
2311.5 53.7 : [ 426.381  587.823  572.276]  =>  [ 948.618 1083.609 516.657]
2246.1 59.9 : [ 395.597  580.393  504.295]  =>  [ 931.672 1079.625 516.572]
2233.8 55.2 : [ 395.597  580.393  504.295]  =>  [ 938.621 1082.976 516.618]
2323.9 50.4 : [ 395.597  580.393  504.295]  =>  [ 959.323 1087.489 516.721]
2525.6 55.3 : [ 395.597  580.393  504.295]  =>  [ 962.848 1079.377 516.683]
2441.8 55.2 : [ 395.597  580.393  504.295]  =>  [ 955.719 1080.477 516.663]
2244.9 62.1 : [ 376.232  610.186  480.097]  =>  [ 929.680 1078.875 516.560]
2350.0 61.0 : [ 376.232  610.186  480.097]  =>  [ 939.153 1077.947 516.588]
2385.1 59.0 : [ 376.232  610.186  480.097]  =>  [ 944.230 1078.393 516.609]
2340.0 52.0 : [ 376.232  610.186  480.097]  =>  [ 955.675 1085.234 516.693]
2317.6 60.9 : [ 395.642  643.962  526.865]  =>  [ 936.511 1078.366 516.581]
2164.7 56.2 : [ 395.642  643.962  526.865]  =>  [ 930.967 1082.863 516.590]
2354.3 56.1 : [ 395.642  643.962  526.865]  =>  [ 946.627 1080.825 516.633]
2149.2 64.3 : [ 391.532  585.304  531.821]  =>  [ 921.298 1079.678 516.536]
```

Figure: Using Regression Model with ASCII Ray Data file.

REFERENCES

[1] M. Abu Alsheikh, S. Lin, D. Niyato, and H.-P. Tan, "Machine learning in wireless sensor networks: Algorithms, strategies, and applications," IEEE Communications Surveys & Tutorials, vol. 16, no. 4, pp. 1996–2018, 2014.

[2] W. Dargie and C. Poellabauer, Localization. John Wiley & Sons, Ltd, 2010, pp. 249–266

[3] B. Sohn, J. Lee, H. Chae, and W. Yu, "Localization system for mobile robot using wireless communication with ir landmark," in Proceedings of the 1st International Conference on Robot Communication and Coordination, ser. RoboComm '07. Piscataway, NJ, USA: IEEE Press, 2007, pp. 6:1–6:6. [Online]. Available: http://dl.acm.org/citation.cfm?id=1377868.1377876

[4] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan, "The cricket location-support system," in Proceedings of the 6th Annual International Conference on Mobile Computing and Networking, ser. MobiCom '00. New York, NY, USA: ACM, 2000, pp. 32–43. [Online]. Available: http://doi.acm.org/10.1145/345910.345917

[5] D. Hahnel, W. Burgard, D. Fox, K. Fishkin, and M. Philipose, "Mapping and localization with rfid technology," in Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on, vol. 1, April 2004, pp. 1015–1020.

[6] L. Ni, Y. Liu, Y. C. Lau, and A. Patil, "Landmarc: indoor location sensing using active rfid," in Pervasive Computing and Communications, 2003. (PerCom 2003). Proceedings of the First IEEE International Conference on, March 2003, pp. 407–415.

[7] L. Aalto, N. Gothlin, J. Korhonen, and T. Ojala, "Bluetooth and wap ¨ push based location-aware mobile advertising system," in Proceedings of the 2Nd International Conference on Mobile Systems, Applications, and Services, ser. MobiSys '04. New York, NY, USA: ACM, 2004, pp. 49–58. [Online]. Available: http://doi.acm.org/10.1145/990064.990073

[8] R. Bruno and F. Delmastro, "Design and analysis of a bluetooth-based indoor localization system," in Personal Wireless Communications, ser. Lecture Notes in Computer Science, M. Conti, S. Giordano, E.

Gregori, and S. Olariu, Eds. Springer Berlin Heidelberg, 2003, vol. 2775, pp. 711–725. [Online]. Available: http://dx.doi.org/10.1007/ 978-3-540-39867-7 66

[9] P. Bahl and V. N. Padmanabhan, “Radar: an in-building rfbased user location and tracking system.” Institute of Electrical and Electronics Engineers, Inc., March 2000. [Online]. Available: http://research.microsoft.com/apps/pubs/default.aspx?id=68671

[10] A. Goswami, L. E. Ortiz, and S. R. Das, “Wigem: A learning-based approach for indoor localization,” in Proceedings of the Seventh COnference on Emerging Networking EXperiments and Technologies, ser. CoNEXT ’11. New York, NY, USA: ACM, 2011, pp. 3:1–3:12. [Online]. Available: http://doi.acm.org/10.1145/2079296.2079299

[11] E. Martin, O. Vinyals, G. Friedland, and R. Bajcsy, “Precise indoor localization using smart phones,” in Proceedings of the International Conference on Multimedia, ser. MM ’10. New York, NY, USA: ACM, 2010, pp. 787–790. [Online]. Available: http://doi.acm.org/10.1145/1873951.1874078

[12] M. Youssef and A. Agrawala, “The horus wlan location determination system,” in Proceedings of the 3rd International Conference on Mobile Systems, Applications, and Services, ser. MobiSys ’05. New York, NY, USA: ACM, 2005, pp. 205–218. [Online]. Available: http://doi.acm.org/10.1145/1067170.1067193