DESIGN AND DEVELOPMENT OF A SINGLE ROUTER DECIMETER LEVEL INDOOR LOCALISATION SYSTEM

Voggu Aravind Reddy

Master of Technology Thesis

June 2020

International Institute of Information Technology, Bangalore

DESIGN AND DEVELOPMENT OF A SINGLE ROUTER DECIMETER LEVEL INDOOR LOCALISATION SYSTEM

Submitted to International Institute of Information Technology,
Bangalore
in Partial Fulfillment of
the Requirements for the Award of
Master of Technology

by

Voggu Aravind Reddy
IMT2015524

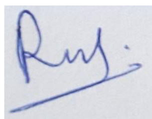
International Institute of Information Technology, Bangalore
June 2020

Dedicated to

Friends and Family

Thesis Certificate

This is to certify that the thesis titled **Design and Development of a Single Router Decimeter Level Indoor Localisation System** submitted to the International Institute of Information Technology, Bangalore, for the award of the degree of **Master of Technology** is a bona fide record of the research work done by **Voggu Aravind Reddy**, **IMT2015524**, under my supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma. The thesis conforms to plagiarism guidelines and compliance as per UGC recommendations.



Dr. Madhav Rao

Bengaluru,

The 28th of June, 2020.

DESIGN AND DEVELOPMENT OF A SINGLE ROUTER DECIMETER LEVEL INDOOR LOCALISATION SYSTEM

Abstract

This Thesis presents the design and implementation of a CSI Fingerprinting based Indoor Localisation system with a resolution of 10 cm using a single off-the-shelf router and a Raspberry Pi 3B+, using no special hardware requirements or hardware modifications. A De-Noising procedure to filter noise from CSI and use temporal-data-fusion to improve accuracy to above 99.90%. Our experiments in both Line-of-Sight and Non-Line-of-Sight have shown a mean error of 3.5 cm at a 10 cm resolution.

Acknowledgements

I would like to thank my mentor and advisor Prof. Madhav Rao for the continuous support, help, patience, guidance, and motivation. I'm deeply grateful for his supervision and numerous hours of collaboration on this Thesis and the work that preceeded it, despite his many other academic and professional commitments.

Besides my mentor, I would like to thank the rest of my thesis committee: Prof. Debabrata Das, and prof. Girish kumar for their help and insights. Also thanks to Ms. Ramadevi S for help with editing and formatting this Thesis.

I would also like to thank Dr. Vikas, whose friendship I've had the good fortune of having. His brilliant ideas, and insights constantly amaze me, and have kept me motivated. My deepest gratitude for his support, help, and advice on things academical and beyond.

Thanks to Prof. Jyotsna Bapat, Prof. T.K. Srikanth, Prof. K Subramaniam, Prof. Subhajit Sen, Prof. Sachit Rao, Prof. V.N. Muralidhara, and other faculty at IIITB, for their teaching, help, support, and encouragement. Also, thanks to Prof. Sazonov for hosting me in his lab during last summer.

I would also like to thank Ms. Adhisaya and Ms. Aswini for their constant help, support, and companionship. Thanks to Mathias Seemoo, Jakob Link, and Mikhael Zakharov, Jet Yu, Lee Yaohua, and Junye for helping me with collecting CSI with the Nexmon Framework.

My best friends Vinay and Krishna were the best part of my stay at IIITB. Thanks for putting up with the shenanigans and bad puns. The memories I had with them will forever be cherished, and they will always have a special place in my heart.

Thanks to Aparna, Akshi, Kedia, Praneeth, and Vivek for being such good friends, and all the good times we had together.

Thanks to Parshi, Alan, Aditya T, Bharath, Utkarsh, and Anshika for the completely unexpected surprises! Thanks to Rachith and Shivang for accompanying me to movies, and for being such cool people to hangout with. Thanks to Mahith for always being up for Pizza. Thanks to Dharmik for all the cool ideas and confusing doubts. Thanks to Vishal, for keeping me company till late night in the lab.

Thanks to Anshuk, Prachi, Sankeerth, Angshuman, Sravya, Juhi, Anagha, Suggu, Suraj, Surya, Vaibhav, Mohith, Ritvik, Gnaneshwar, Harsha, Sai Charan, Rohil pal, Shiloni, Abhishek, Soumya, Abhiramon, Atharva, Pranith, Kedar, Kushagra, Charan, Shebaz, Naman, Rakshith, Pranav, Atibhi, Sushma, Sri Ram, and the rest of my classmates. You have made my stay and IIITB truly enjoyable.

Thanks to Arvind, Ronak, Ajay, Nithin, Anurag, Sailesh, Shubhayu, Tanmay, Saad, and others for working with me on several projects, and whom I've had a great pleasure teaching. Thanks to Sanjay, Mounika, Monika, Ajay, Anmol, Swagatika, and Aarthi, for all the Badminton. Thanks to Manonmaie, for the friendship, and company which made the day long journey between Khammam and Bangalore seem not so much.

Thanks to Sai Kiran, Saad, Khaveesh, Shalini, Shubhayu, and Tanmay, without whose help, I could never have lead the Internet Committee. Also special thanks to Mr. Murugan, Sasi, Prem, and others in the Data Center who helped us tremendously.

Thanks to Ajay, Abhishek, Shashank, Raghav, Viprav, Tanmoy, Delwar, Masudul, and Connor for helping me every step of the way at Alabama.

Special thanks to Prof. T.K. Srikanth, the E-Health Research Center, the Visvesvaraya Young Faculty Research Fellowship, The National Institute of Mental Health and Neuro-Sciences (NIMHANS), and the International Institute of Information Technology Bangalore (IIITB) for the support, and for funding this research needed for this Thesis.

I'm deeply thankful for my ever loving and supportive family. My beloved parents, who constantly inspired, supported and encouraged me to realize my dreams and push my limits. They have alway understood and gave me the freedom to explore. Thanks to my siblings, for all the years of mischeif and love, which got me through the toughest of times.

Contents

List of Figures

List of Tables

List of Abbreviations

AP .... Access Point

AoA .... Angle of Arrival

CM .... Centimetre

CSI .... Channel State Information

DTC .... Decision Tree Classifier

GNBC .... Gaussian Naive Bayes Classifier

GPS .... Global Positioning System

IIITB .... International Institute of Information Technology Bangalore

LOS .... Line of Sight

MM .... Millimetre

NLOS .... Non Line of Sight

NNR .... Non-Noise-Reduced

NR .... Noise-Reduced

OS .... Operating System

Pi .... Raspberry Pi

RFC .... Random Forest Classifier

RSSI .... Received Signal Strength Indication

SVC .... Support Vector Classifier

ToF .... Time of Flight

CHAPTER 1

INTRODUCTION

Outdoor localisation systems like GPS (Global Positioning System) have had a profound impact on the World, allowing technologies ranging from Navigation to self-driving cars, and trans-continental flights that can operate without having to rely on landmarks. A similarly impactful indoor counterpart doesn't exist yet because of the challenging Radio-Environment indoor locations present, and the complexity and requirements of existing Indoor Localisation systems. While GPS enjoys of Line-of-Sight signal availability, and virtually absent multipath effect, indoor systems are challenged by Multipath effect, Fading, and Shadowing. This Thesis describes a method of indoor localisation by turning these very challenges into advantages, and using them to localise with a single commodity off-the-shelf router indoors.

This Thesis uses a localisation technique called Fingerprinting. Localisation based on Fingerprinting consists of two phases: Online and Offline. The Online phase is also called the Training Phase, and includes collecting measurements at various points at the localisation site. Offline phase is also called Test Phase, and includes finding the location of a device whose location is unknown, by making measurements again and comparing them to measurements taken during the online phase.

Traditionally, Fingerprinting based Localisation has been done using RSSI (Received Signal Strength Indication). RSSI is an estimation of how good the wireless link between the Device and the Access Point is, and roughly correlates with the distance between them. Although this is easy to do, techniques using RSSI have acheived a poor accuracy because of the unstability of RSSI, with their median accuracy ranging from 2-4 metres. Alternatively, CSI (Channel State Information) can be used for Fingerprinting. CSI is the frequency response of the channel between Access Point and Device [1]. DeepFi [1] uses CSI and Deep Learning to acheive a mean error of 94 cm. Please refer to Chapter 2 for a more detailed survey of related work.

The main contribution of this Thesis is a localisation system that uses WiFi CSI Fingerprinting to acheive an indoor localisation resolution of 10 cm with a mean error of 3.50 cm using a single off-the-shelf WiFi router and a Raspberry Pi3B+. No modifications to the Hardware are required for our method to work. We use two techniques to acheive this accuracy:

1.1 A CSI De-Noising Algorithm

CSI data is sensitive to movement in the Environment, and this noise reduces localisation accuracy. We use data from multiple CSI samples to create a new CSI sample free of noise, and use it for localisation.

1.2 Data Fusion to improve accuracy

We predict location several times a second, and use the data from those predictions to create a new, much more accurate prediction.

The Denoising Algorithm is presented in Chapter 3.3.3 and the Data Fusion technique is described in Chapter 4.

CHAPTER 2

REVIEW OF RELATED LITERATURE

The research into Indoor Localisation Systems has produced several techniques, including localisation based on Angle of Arrival (AoA), Time of Flight (ToF), and Fingerprinting. A significant part of the literature review in this Thesis has been adapted from the same section in *SpotFi* [2].

2.1 Angle of Arrival

Localisation is acheived by triangulating device location using Angle of Arrival between device and router calculated using multipath signals [3–9]. AoA based systems use multiple antennas [3] or moving antennas [5] to measure angle. SpotFi acheives the best known resolution at 40 cm [2].

SpotFi [2] uses a super-resolution algorithm to compute Angle of Arrival accurately and acheives a median accuracy of 40 cm, but needs atleast 3 antennas, and has been implemented only on an Intel 5300 WiFi Network Interface Card. The Intel chip can only operate at 20MHz bandwidth, supports WiFi technologies only upto IEEE 802.11n, and cannot use Wireless Access Points that use encryption for localisation.

2.2 Time based

Best known accuracy obtained from a Time Based system that doesn't need time synchronisation between Access Point and Device is 2 meters [10–12].

Distance between the Access Point and the Device can be calculated by multiplying the Time of Flight between them, and the speed of light. This distance can be used to triangulate the position of the Device. *Pinpoint*, and other techniques acheive results in the range of a few metres [8, 10–14].

Synchronicity [15], *Sourcesync* [16], and others [17–20] synchronise the clock between the Device and the Access Point and apply super resolution algorithms to obtain a more accurate ToF. *SI-JADE* [21], and others [22–25] combine AoA and ToA estimates to improve accuracy. The papers by Picheral [26], and others [27–30] test the same idea in simulation.

2.3 Fingerprinting based

Fingerprinting based techniques maintain a database of measurements a quantity at points of interest, like Magnetism, or RSSI, or CSI and use that to find location.

Zee [31], *SurroundSense* [32] *Centaur* [33], *Horus* [33], and others [34–38] use Fingerprinting for localisation. *Horus* provides the best known accuracy with a median error of 60 cm, and a tail accuracy around 1.3 m.

CHAPTER 3

SYSTEM DESIGN AND METHODOLOGY

3.1 Device Setup

3.1.1 Raspberry Pi

A Raspberry Pi 3B+ / or a Raspberry Pi 4B can be used for localisation. We used a Raspberry Pi 3B+ because of it's lower power requirements and more widespread availability and adoption at the time of writing. No additional antennas or wireless equipment other that what was built into the Raspberry Pi 3B+ was used for collecting CSI data.

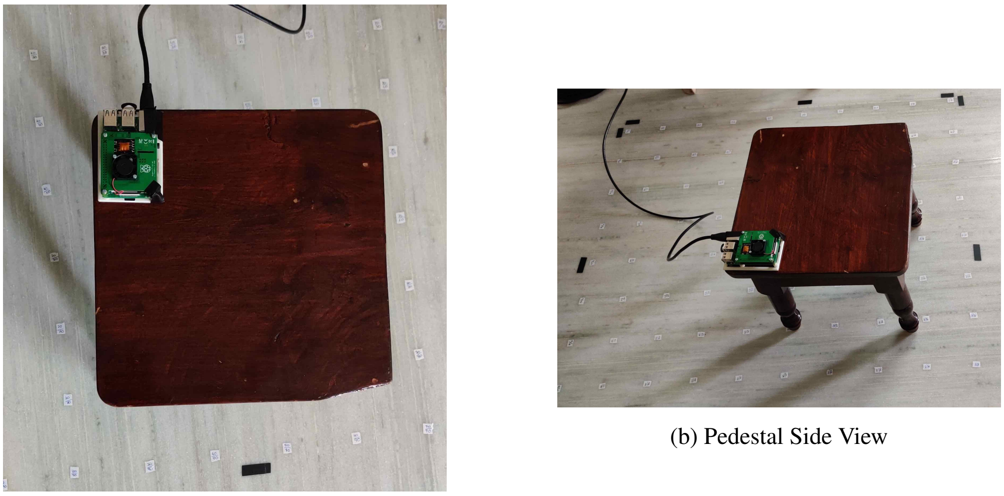
The default Broadcomm firmware doesn't allow modifications and collection of CSI data. A C-based firmware patching framework Nexmon [39, 40] was used to patch Nexmon_CSI [41] firmware to enable CSI collection.

Although Nexmon supports CSI collection while connected to a Wireless Access Point, we disconnected the Raspberry Pi from Wireless Access Points to prevent any potential interference. The Raspberry Pi was controlled over an Ethernet link instead. The Ethernet link was also used to provide power to the Raspberry pi using a Raspberry Pi Power over Ethernet HAT.

At the time of this writing, there exist a few Null pointer Dereferences and other issues in Nexmon_CSI that lead to Firmware Traps. This leads to CSI collection stalling, and needs a reboot to continue collection. An unmerged fix [42] by Mikhail Zakharov was manually added to correct this behaviour.

Nexmon_CSI embeds the CSI data into a UDP stream which can be collected by a utility like TCPdump.

The Raspberry Pi was fixed atop a wooden pedestal to maintain a 30 cm clearance above the floor, and to keep it immobile while collecting data. The pedestal can be seen in Figure FC3.1. An Aluminium stand made of extruded beams, as seen in Figure FC3.2 was also tested and gave good results. When the CSI collection device is too close to a flat surface, reflections of Radio waves from that surface don't reach the device. So a clearance from the floor is theorised to help increase the variance of CSI samples collected and improve accuracy.



(a) Pedestal Top View

(b) Pedestal Side View

Figure FC3.1: Picture of the wooden pedestal with a Raspberry Pi 3B+ fixed on top. The Raspberry Pi is encased in a 3D printed plastic case and has a Power over Ethernet Hat on top.

Figure FC3.2: Picture of an Aluminium stand made from Makerbeam extruded Aluminium beams. The Raspberry Pi is fixed at the top, while a Power Bank used to power is secured at the bottom. The Raspberry Pi is encased when using this stand to prevent Electrical Short-circuits as it's backed against metal.

3.1.2 Wireless Access Point

Any Wireless Access Point with support for 802.11ac is suitable. We used a TP-link Archer C20 set to operate in channel 36 at a maximum bandwith of 40 MHz. Access Points created with the WiFi Hotspot feature in Cellphones may not be suitable for localisation. Tests with a OnePlus 7T operating as an Access Point show increased noise in CSI data considerably lower localisation accuracy.

3.2 Collecting CSI

CSI data was collected at two locations, a Line-of-Sight location named *los-a*, and a Non-Line-of-Sight location named *nlos-b*. A Raspberry Pi 3B+ device was used to collect CSI data from a TP-link Archer C20 802.11ac compatible Access Point operating at 40MHz bandwidth on 802.11ac Channel 36.

12000 samples of CSI data was collected at each of the 100 locations spaced 10 cm apart at *los-a*, and at 18 locations spaced 10 cm apart at *nlos-b*. The Raspberry Pi was fixed atop a wooden pedestal while collecting data, to maintain a clearance above ground. Photos of locations *los-a* and *nlos-b* are in Figure FC3.3 and Figure FC3.4 respectively.

A laptop was used to generate WiFi traffic so that CSI can be measured. While passive Wireless traffic exists at all times around a Wireless AP, it is slow, and sometimes occupies only 20MHz of the available 40MHz bandwidth. A graphic depicting CSI collection is in Figure FC3.5.

No additional equipment other than what's built-in to Raspberry Pi is used for the collection of CSI. The *wpa-supplicant* linux module was removed to prevent any potential interference with CSI collection. The Raspberry Pi stays disconnected from any

Figure FC3.3: Picture of Line of Sight site A: *los-a*.

WiFi Access Points, and was controlled over an Ethernet connection.

Figure FC3.4: Picture of Non Line of Sight site B: *nlos-b*.

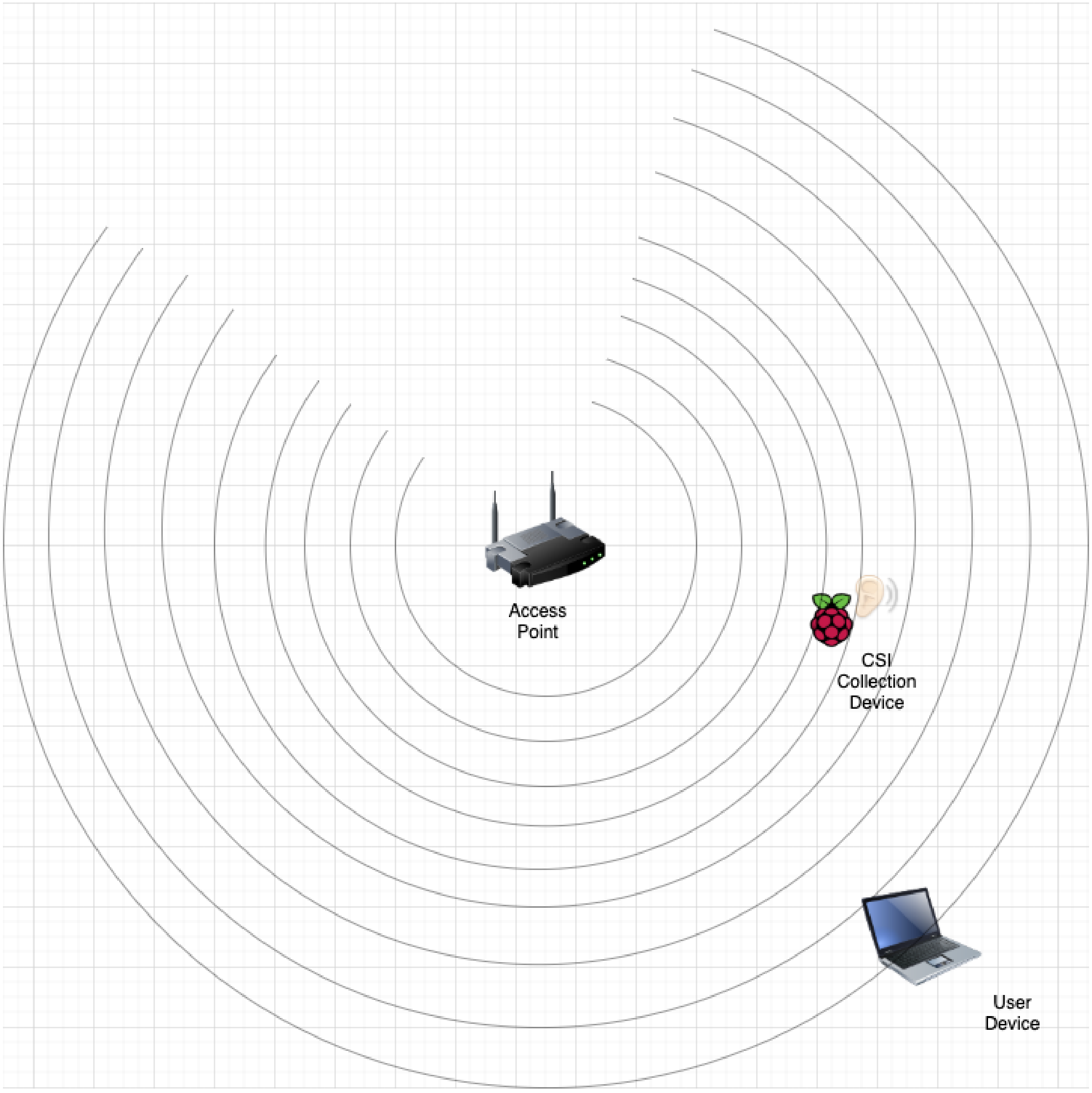


Figure FC3.5: Graphic depicting CSI collection. Wireless traffic is generated as the User Device accesses the network, the Raspberry Pi measures the CSI data for this traffic.

3.3 Preprocessing CSI

3.3.1 Reading CSI from pcap files

CSI data from Nexmon is calculated from both I&Q components, and as such exists as a Complex-128 number in a pcap file. A custom Python script was used to read CSI values, calculate the magnitude, and typecast it to Float32. Phase information in CSI was not used for localisation.

3.3.2 Removing unwanted subcarriers

Subcarriers -64, -63, -62, -61, -60, -59, -1, 0, 1, 59, 60, 61, 62, 63 of a 801.11ac 40MHz WiFi link are called Null-Subcarriers, and have arbitrary CSI values. These are useful for Wireless Coexistence, but are not useable for localisation. Similarly, Subcarriers 11, 25, 53, -11, -25, -53 are Pilot subcarriers used to control the Wireless link. We remove both these sets of Subcarriers from our CSI data. An image of CSI data after removing these subcarriers is in Figure FC3.6

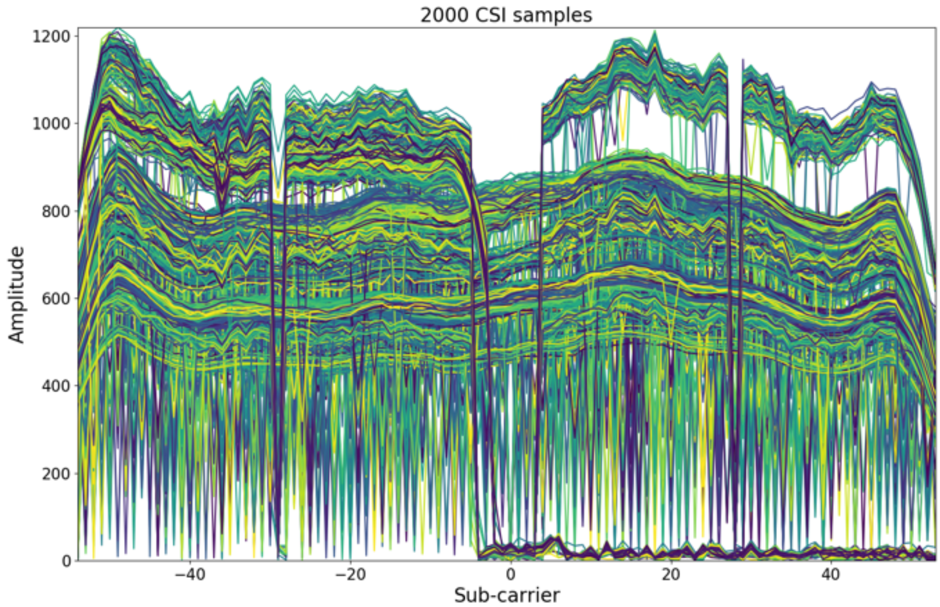


Figure FC3.6: CSI data after removing Null and Pilot Subcarriers at Line of Sight location (0, 0).

3.3.3 Noise Reduction

Further, for every sample, subcarriers with Magnitude greater than 4000 are considered noise and were made equal to 0.

CSI data is sensitive to movement in the environment, and such noise will affect localisation accuracy. At each location, an average CSI value is calculated, and 1/3rd of the samples that are furthest from this mean are considered noise and are removed. From the remaining 2/3rd samples, every 4 samples are averaged. Since we collected 12000 samples at each location, after processing, we have 2000 samples per location. An image of CSI data after noise reduction is in Figure FC3.7. The decision to average every 4 consecutive samples was arrived at heuristically. According to our experiments, the least number of samples to be averaged for a reasonably noise-less CSI is 4. Using a higher number of samples may increase the prediction accuracy, but will also increase the number of CSI samples needed to localise.

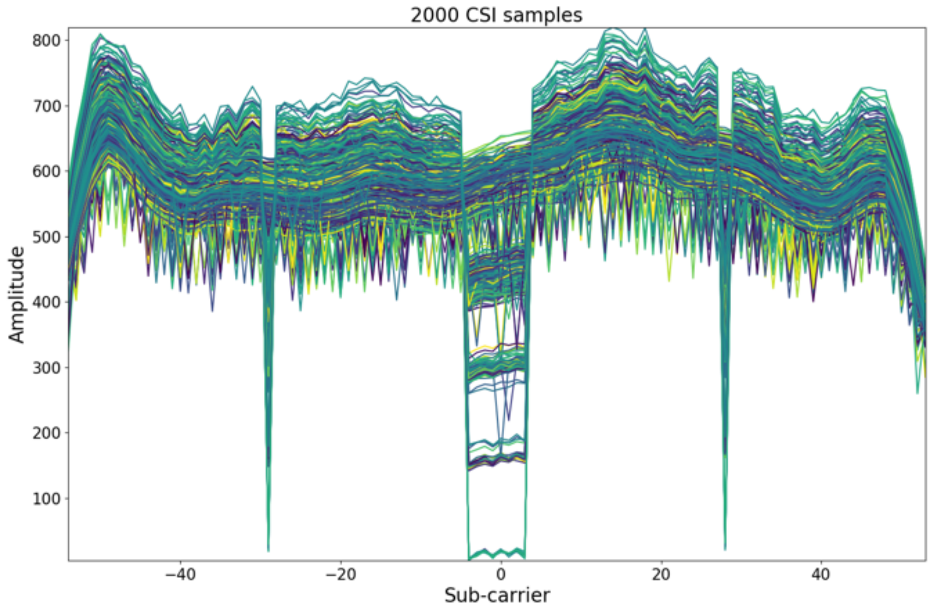


Figure FC3.7: CSI data after noise reduction at Line of Sight location (0, 0).

3.3.4 Normalisation

Classification algorithms like Support Vector Classifier recommend normalisation of data by shifting the Mean to 0, and scaling the data to have a unit variance, but we found that such a normalisation substantially reduces localisation accuracy. **No normalisation was done on the dataset**.

3.4 Localising with CSI

Four Classification algorithms from Python Scikit-learn library were used for localisation: Gaussian Naive Bayes Classifier, Support Vector Classifier, Decision Tree Classifier, Random Forest Classifier. Classification accuracies were measured at 3 localisation resolutions: 10 cm, 20 cm, 40 cm. Classification accuracy with no noise-reduction was also measured for all cases except Line of Sight with 10 cm resolution.

All the Classifiers use the default Hyperparamters as of Scikit-learn version 0.23.1, but Hyperparamters for each of the Classification algorithms are furnished below for reproducibility.

- Gaussian Naive Bayes Classifier
  - Smoothening = 1e-9
- Support Vector Classifier
  - Regularisation Parameter - C = 1.0
  - Kernel = Radial Basis Function 'rbf'
  - Kernel Coefficient - gamma = 'scale'
  - Tolerance = 1e-3
- Decision Tree Classifier
  - Split Critereon = 'gini'
- Random Forest Classifier
  - Number of Estimators = 20
  - Split Critereon = 'gini'

CHAPTER 4

RESULTS

Classification accuracies and Mean Distance Errors for Gaussian Naive Bayes, Support Vector, Decision Tree, and Random Forest Classifiers were measured in both Line-of-Sight and Non-Line-of-Sight scenarios, at resolutions of 10 cm, 20 cm, and 40 cm, with and without noise reduction.

Table TC4.1 shows the Classification Accuracies for Gaussian Naive Bayes, Support Vector, Decision Tree, and Random Forest Classifiers, in Line-of-Sight and Non-Line-of-Sight scenarios, with and without Noise Reduction at a resolution of 10 cm.

Table TC4.2 shows the Mean Distance Error in Centimetre for Gaussian Naive Bayes, Support Vector, Decision Tree, and Random Forest Classifiers, in Line-of-Sight and Non-Line-of-Sight scenarios, with and without Noise Reduction at a resolution of 10 cm.

Table TC4.3 shows the Classification Accuracies for Gaussian Naive Bayes, Support Vector, Decision Tree, and Random Forest Classifiers, in Line-of-Sight and Non-Line-of-Sight scenarios, with and without Noise Reduction at a resolution of 20 cm.

Table TC4.4 shows the Mean Distance Error in Centimetre for Gaussian Naive Bayes, Support Vector, Decision Tree, and Random Forest Classifiers, in Line-of-Sight and Non-Line-of-Sight scenarios, with and without Noise Reduction at a resolution of

20 cm.

Table TC4.5 shows the Classification Accuracies for Gaussian Naive Bayes, Support Vector, Decision Tree, and Random Forest Classifiers, in Line-of-Sight and Non-Line-of-Sight scenarios, with and without Noise Reduction at a resolution of 40 cm.

Table TC4.6 shows the Mean Distance Error in Centimetre for Gaussian Naive Bayes, Support Vector, Decision Tree, and Random Forest Classifiers, in Line-of-Sight and Non-Line-of-Sight scenarios, with and without Noise Reduction at a resolution of 40 cm.

Table TC4.1: Classification Accuracy in percentage for 10 cm resolution

Classification Accuracy in percentage for 10 cm resolution				
Classifier	LOS	LOS unfiltered	NLOS	NLOS unfiltered
NBC	71.17%	N/A	90.90%	76.16%
SVC	92.73%	N/A	98.71%	95.50%
DTC	86.33%	N/A	96.46%	91.68%
RFC	93.15%	N/A	98.01%	94.10%

Table TC4.2: Mean Error in Centimetre for 10 cm resolution

Mean Error in Centimetre for 10 cm resolution				
Classifier	LOS	LOS unfiltered	NLOS	NLOS unfiltered
NBC	13.80 cm	N/A	2.24 cm	8.16 cm
SVC	3.43 cm	N/A	0.42 cm	1.18 cm
DTC	6.53 cm	N/A	1.27 cm	2.46 cm
RFC	3.18 cm	N/A	0.65 cm	1.54 cm

Table TC4.3: Classification Accuracy in percentage for 20 cm resolution

Classification Accuracy in percentage for 20 cm resolution				
Classifier	LOS	LOS unfiltered	NLOS	NLOS unfiltered
NBC	85.78%	26.80%	95.10%	77.19%
SVC	98.70%	96.11%	98.50%	96.60%
DTC	97.04%	95.44%	95.83%	92.51%
RFC	98.95%	98.02%	97.29%	94.55%

Table TC4.4: Mean Error in Centimetre for 20 cm resolution

Mean Error in Centimetre for 20cm resolution				
Classifier	LOS	LOS unfiltered	NLOS	NLOS unfiltered
NBC	7.57 cm	36.49 cm	2.01 cm	7.99 cm
SVC	0.79 cm	1.88 cm	0.49 cm	0.96 cm
DTC	1.59 cm	2.37 cm	1.62 cm	2.40 cm
RFC	0.53 cm	0.97 cm	1.01 m	1.56 cm

Table TC4.5: Classification Accuracy in percentage for 40 cm resolution

Classification Accuracy in percentage for 40 cm resolution				
Classifier	LOS	LOS unfiltered	NLOS	NLOS unfiltered
NBC	97.14%	29.82%	85.70%	96.00%
SVC	98.56%	97.73%	98.09%	95.81%
DTC	98.58%	97.58%	96.59%	92.38%
RFC	99.53%	98.88%	97.47%	92.25%

Table TC4.6: Mean Error in Centimetre for 40 cm resolution

Mean Error in Centimetre for 40 cm resolution				
Classifier	LOS	LOS unfiltered	NLOS	NLOS unfiltered
NBC	2.00 cm	44.90 cm	5.72 cm	1.60 cm
SVC	1.06 cm	1.50 cm	0.77 cm	1.68 cm
DTC	0.99 cm	1.57 cm	1.38 cm	3.05 cm
RFC	0.27 cm	0.72 cm	1.02 cm	3.10 cm

We find that the performance is similar in both Line-of-Sight and Non-Line-of-Sight scenarios. Support Vector Classifier and Random Forest Classifier are consistently the highest performers. Accuracy of the Naive Bayes Classifier improved significantly with the noise-reduction procedure described in Chapter 3.3.3.

To further improve accuracy, localisation can be performed 5 times, and the majority prediction among the five predictions can be picked as the final prediction. The improved accuracy with this technique can be calculated with Equation Eqn 4.1.

$$p_{final} = 1 - ((1 - P_c)^3 * P_c^2 + (1 - P_c)^4 * P_c^1 + (1 - P_c)^5 * P_c^0) \qquad \text{(Eqn 4.1)}$$

where:

P_c = Initial probability of correct Location Classification.

P_{final} = Post Data-Fusion probability of correct Location Classification.

Figure FC4.1 shows the relationship between Classification Accuracy and Improved Accuracy by combining 5 predictions. The Final Accuracy is above **99.90%** for all Classification Accuracies above **90%**.

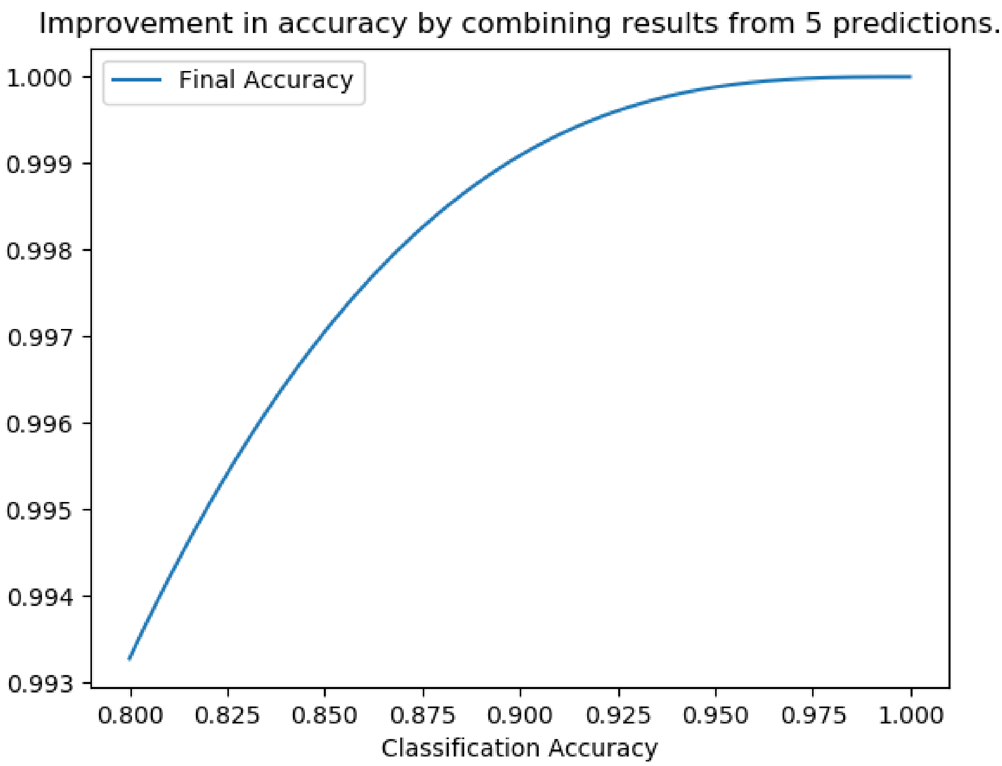


Figure FC4.1: Relationship between Classification Accuracy and Improved Accuracy by combining 5 predictions.

CHAPTER 5

CONCLUSIONS

We have successfully localised a Raspberry Pi 3B+ with no hardware modifications or additional hardware with a precision of 10 cm using a single off-the-shelf WiFi Access Point. The Nexmon_CSI framework supports more devices like Raspberry Pi 4B, Nexus 5, Nexus 6P, and Asus RT-AC86U. We hope that more chips allow collection of CSI data, and with emergence of simple but accurate techniques like this, Indoor Localisation becomes ubiquitous.

Time of Flight, and RSSI fingerprinting techniques require multiple Access Points to operate. Similarly, Angle of Arrival techniques need multiple antennas to compute the angle. Our ability to localise accurately with a single router is because of the multipath effect, which makes the channel response unique at each location, and creates a distinct fingerprint.

Bibliography

[1] X. Wang, L. Gao, S. Mao, and S. Pandey, “Csi-based fingerprinting for indoor localization: A deep learning approach,” *IEEE Transactions on Vehicular Technology*, vol. 66, no. 1, pp. 763–776, 2017.

[2] M. Kotaru, K. Joshi, D. Bharadia, and S. Katti, “Spotfi: Decimeter level localization using wifi,” *SIGCOMM Computer Communication Review*, vol. 45, p. 269–282, Aug. 2015.

[3] J. Xiong and K. Jamieson, “Arraytrack: A fine-grained indoor location system,” in *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation*, nsdi’13, (USA), p. 71–84, USENIX Association, 2013.

[4] S. Kumar, S. Gil, D. Katabi, and D. Rus, “Accurate indoor localization with zero start-up cost,” in *Proceedings of the 20th annual international conference on Mobile computing and networking - MobiCom ’14*, (Maui, Hawaii, USA), pp. 483–494, ACM Press, 2014.

[5] S. Kumar, E. Hamed, D. Katabi, and L. Erran Li, “LTE radio analytics made easy and accessible,” in *Proceedings of the 6th annual workshop on Wireless of the students, by the students, for the students - S3 ’14*, (Maui, Hawaii, USA), pp. 29–30, ACM Press, 2014.

[6] J. Gjengset, J. Xiong, G. McPhillips, and K. Jamieson, “Phaser: enabling phased array signal processing on commodity WiFi access points,” in *Proceedings of the*

20th annual international conference on Mobile computing and networking - MobiCom '14, (Maui, Hawaii, USA), pp. 153–164, ACM Press, 2014.

[7] S. Sen, J. Lee, K.-H. Kim, and P. Congdon, "Avoiding multipath to revive inbuilding WiFi localization," in *Proceeding of the 11th annual international conference on Mobile systems, applications, and services - MobiSys '13*, (Taipei, Taiwan), p. 249, ACM Press, 2013.

[8] M. Youssef, A. Youssef, C. Rieger, U. Shankar, and A. Agrawala, "Pinpoint: An asynchronous time-based location determination system," in *Proceedings of the 4th International Conference on Mobile Systems, Applications and Services*, MobiSys '06, (New York, NY, USA), p. 165–176, Association for Computing Machinery, 2006.

[9] D. Niculescu and B. Nath, "VOR base stations for indoor 802.11 positioning," in *Proceedings of the 10th annual international conference on Mobile computing and networking - MobiCom '04*, (Philadelphia, PA, USA), p. 58, ACM Press, 2004.

[10] A. Marcaletti, M. Rea, D. Giustiniano, V. Lenders, and A. Fakhreddine, "Filtering Noisy 802.11 Time-of-Flight Ranging Measurements," in *Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies - CoNEXT '14*, (Sydney, Australia), pp. 13–20, ACM Press, 2014.

[11] M. Ciurana, F. Barcelo-Arroyo, and F. Izquierdo, "A ranging system with ieee 802.11 data frames," in *2007 IEEE Radio and Wireless Symposium*, pp. 133–136, 2007.

[12] S. A. Golden and S. S. Bateman, "Sensor Measurements for Wi-Fi Location with Emphasis on Time-of-Arrival Ranging," *IEEE Transactions on Mobile Computing*, vol. 6, pp. 1185–1198, Oct. 2007.

[13] A. T. Mariakakis, S. Sen, J. Lee, and K.-H. Kim, “SAIL: single access point-based indoor localization,” in *Proceedings of the 12th annual international conference on Mobile systems, applications, and services - MobiSys ’14*, (Bretton Woods, New Hampshire, USA), pp. 315–328, ACM Press, 2014.

[14] S. Lanzisera, D. Zats, and K. S. J. Pister, “Radio frequency time-of-flight distance measurement for low-cost wireless sensor localization,” *IEEE Sensors Journal*, vol. 11, no. 3, pp. 837–845, 2011.

[15] J. Xiong, K. Jamieson, and K. Sundaresan, “Synchronicity: pushing the envelope of fine-grained localization with distributed mimo,” in *Proceedings of the 1st ACM workshop on Hot topics in wireless - HotWireless ’14*, (Maui, Hawaii, USA), pp. 43–48, ACM Press, 2014.

[16] H. Rahul, H. Hassanieh, and D. Katabi, “SourceSync: a distributed wireless architecture for exploiting sender diversity,” in *Proceedings of the ACM SIGCOMM 2010 conference on SIGCOMM - SIGCOMM ’10*, (New Delhi, India), p. 171, ACM Press, 2010.

[17] A. F. Cavanaugh, M. Lowe, D. Cyganski, and J. Duckworth, “Wpi precision personnel location system: Rapid deployment antenna system and sensor fusion for 3d precision location,” in *Proceedings of the 2010 International Technical Meeting of The Institute of Navigation*, (Catamaran Resort Hotel, San Diego, CA, USA), pp. 214–219, 2010.

[18] V. Amendolare, D. Cyganski, and R. J. Duckworth, “Transactional Array Reconciliation Tomography for Precision Indoor Location,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 50, pp. 17–32, Jan. 2014.

[19] S. Venkatraman and J. Caffery, “Hybrid toa/aoa techniques for mobile location in non-line-of-sight environments,” in *2004 IEEE Wireless Communications and*

Networking Conference (IEEE Cat. No.04TH8733), vol. 1, pp. 274–278 Vol.1, 2004.

[20] F. Zhao, W. Yao, C. C. Logothetis, and Y. Song, "Super-resolution toa estimation in ofdm systems for indoor environments," in *2007 IEEE International Conference on Networking, Sensing and Control*, pp. 723–728, 2007.

[21] A.-J. van der Veen, M. Vanderveen, and A. Paulraj, "SI-JADE: an algorithm for joint angle and delay estimation using shift-invariance properties," in *First IEEE Signal Processing Workshop on Signal Processing Advances in Wireless Communications*, (Paris, France), pp. 161–164, IEEE, 1997.

[22] M. Wax and A. Leshem, "Joint estimation of time delays and directions of arrival of multiple reflections of a known signal," *IEEE Transactions on Signal Processing*, vol. 45, pp. 2477–2484, Oct. 1997.

[23] M. Vanderveen, A.-J. Van der Veen, and A. Paulraj, "Estimation of multipath parameters in wireless communications," *IEEE Transactions on Signal Processing*, vol. 46, pp. 682–690, Mar. 1998.

[24] M. Vanderveen, B. Ng, C. Papadias, and A. Paulraj, "Joint angle and delay estimation (JADE) for signals in multipath environments," in *Conference Record of The Thirtieth Asilomar Conference on Signals, Systems and Computers*, (Pacific Grove, CA, USA), pp. 1250–1254, IEEE Comput. Soc. Press, 1997.

[25] Yung-Yi Wang, Jiunn-Tsair Chen, and Wen-Hsien Fang, "TST-MUSIC for joint DOA-delay estimation," *IEEE Transactions on Signal Processing*, vol. 49, pp. 721–729, Apr. 2001.

[26] J. Picheral and U. Spagnolini, "Shift invariance algorithms for the angle/delay estimation of multipath space-time channel," in *IEEE VTS 53rd Vehicular Technology*

Conference, Spring 2001. Proceedings (Cat. No.01CH37202), vol. 1, (Rhodes, Greece), pp. 83–87, IEEE, 2001.

[27] D. Inserra and A. M. Tonello, "A Frequency-Domain LOS Angle-of-Arrival Estimation Approach in Multipath Channels," *IEEE Transactions on Vehicular Technology*, vol. 62, pp. 2812–2818, July 2013.

[28] Jiunn-Tsair Chen, Joonsuk Kim, and Jen-Wei Liang, "Multichannel MLSE equalizer with parametric FIR channel identification," *IEEE Transactions on Vehicular Technology*, vol. 48, pp. 1923–1935, Nov. 1999.

[29] G. Raleigh and T. Boros, "Joint space-time parameter estimation for wireless communication channels," *IEEE Transactions on Signal Processing*, vol. 46, pp. 1333–1343, May 1998.

[30] Jin He, M. N. S. Swamy, and M. O. Ahmad, "Joint Space-Time Parameter Estimation for Underwater Communication Channels with Velocity Vector Sensor Arrays," *IEEE Transactions on Wireless Communications*, vol. 11, pp. 3869–3877, Nov. 2012.

[31] A. Rai, K. K. Chintalapudi, V. N. Padmanabhan, and R. Sen, "Zee: zero-effort crowdsourcing for indoor localization," in *Proceedings of the 18th annual international conference on Mobile computing and networking - Mobicom '12*, (Istanbul, Turkey), p. 293, ACM Press, 2012.

[32] M. Azizyan, I. Constandache, and R. Roy Choudhury, "SurroundSense: mobile phone localization via ambience fingerprinting," in *Proceedings of the 15th annual international conference on Mobile computing and networking - MobiCom '09*, (Beijing, China), p. 261, ACM Press, 2009.

[33] R. Nandakumar, K. K. Chintalapudi, and V. N. Padmanabhan, "Centaur: locating devices in an office environment," in *Proceedings of the 18th annual international*

tional conference on Mobile computing and networking - Mobicom '12, (Istanbul, Turkey), p. 281, ACM Press, 2012.

[34] S. Sen, B. Radunovic, R. R. Choudhury, and T. Minka, "You are facing the Mona Lisa: spot localization using PHY layer information," in *Proceedings of the 10th international conference on Mobile systems, applications, and services - MobiSys '12*, (Low Wood Bay, Lake District, UK), p. 183, ACM Press, 2012.

[35] Z. Yang, C. Wu, and Y. Liu, "Locating in fingerprint space: wireless indoor localization with little human intervention," in *Proceedings of the 18th annual international conference on Mobile computing and networking - Mobicom '12*, (Istanbul, Turkey), p. 269, ACM Press, 2012.

[36] H. Liu, Y. Gan, J. Yang, S. Sidhom, Y. Wang, Y. Chen, and F. Ye, "Push the limit of WiFi based localization for smartphones," in *Proceedings of the 18th annual international conference on Mobile computing and networking - Mobicom '12*, (Istanbul, Turkey), p. 305, ACM Press, 2012.

[37] H. Wang, S. Sen, A. Elgohary, M. Farid, M. Youssef, and R. R. Choudhury, "No need to war-drive: unsupervised indoor localization," in *Proceedings of the 10th international conference on Mobile systems, applications, and services - MobiSys '12*, (Low Wood Bay, Lake District, UK), p. 197, ACM Press, 2012.

[38] M. Youssef and A. Agrawala, "Small-scale compensation for WLAN location determination systems," in *2003 IEEE Wireless Communications and Networking, 2003. WCNC 2003.*, vol. 3, (New Orleans, LA, USA), pp. 1974–1978, IEEE, 2003.

[39] M. Schulz, D. Wegemer, and M. Hollick, "Nexmon: The c-based firmware patching framework," *GitHub repository `https://github.com/seemoo-lab/nexmon` as of commit '87218fb'*, 2017.

[40] F. Gringoli, M. Schulz, J. Link, and M. Hollick, "Free your csi: A channel state information extraction platform for modern wi-fi chipsets," in *Proceedings of the 13th International Workshop on Wireless Network Testbeds, Experimental Evaluation & Characterization*, WiNTECH '19, p. 21–28, 2019.

[41] M. Jakob Link, "Nexmon csi: Channel state information extraction on various broadcom wi-fi chips," *GitHub repository* https://github.com/seemoo-lab/nexmon_csi *as of commit 'aa370c6'*, 2019.

[42] M. Zakharov, "Pull request to fix null pointer dereference, backport a-msdu fixes for raspberry pi, rssi, frame control, phy bits, etc.," *GitHub repository* https://github.com/seemoo-lab/nexmon_csi/pull/46 *as of commit 'c4782be'*, 2020.

APPENDIX A

COMPUTER CODE

```
from pathlib import Path
import numpy as np
from joblib import dump
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.metrics import plot_confusion_matrix
import matplotlib.pyplot as plt
from lib import hdf
import config

def mean_dist_error(cmat, labels):
    nmat = np.array(cmat)
    numel = nmat.sum()

    coords = np.array([(int(x.split('-')[0]), int(x.split('-')[1])) for x in
    labels])

    distmat = np.zeros((len(coords), len(coords)))
    for i in range(len(coords)):
        distmat[i] = np.sqrt(np.square(coords - coords[i]).sum(axis=1))

    distmean = (distmat * nmat).sum() / numel
    accuracy = (nmat.trace() * 1.0) / numel

```

```
    print(distmean, accuracy)

    return (distmean, accuracy)


for site in [Path(x) for x in Path(config.path_data).iterdir() if x.is_dir()]:
    model_info = {
        'nbc': {'name': 'Naive Bayes Classifier'},
        'svc': {'name': 'Support Vector Classifier'},
        'dtc': {'name': 'Decision Tree Classifier'},
        'rfc': {'name': 'Random Forest Classifier'}
    }

    models = {
        # 'nbc': GaussianNB,
        'svc': lambda: SVC(
            cache_size=1000
        ),
        # 'dtc': DecisionTreeClassifier,
        'rfc': lambda: RandomForestClassifier(n_estimators=20)
    }

    print(site)

    scoredir = Path(str(site).replace(config.path_data, config.path_scores))
    modeldir = Path(str(site).replace(config.path_data, config.path_models))
    if not scoredir.exists():
        scoredir.mkdir()
        modeldir.mkdir()

        hdf_files = site.glob('*.h5')

        X, y, X_t, y_t = hdf.split_dataset(
            hdf_files,
            0.2,
        )

        for model in models:
            print(model, 'fitting')
```

```python
            m = models[model]()
            m.fit(X, y)

            dump(m, str(modeldir) + '/%s.joblib' % (model))

            print(model, 'fit complete')
            continue

            cmatplot = plot_confusion_matrix(m, X_t, y_t, cmap=plt.cm.Blues, values_format='d')
            cmatplot.figure_.set_size_inches(80, 60)

            cmat = cmatplot.confusion_matrix
            (distmean, accuracy) = mean_dist_error(cmat, cmatplot.display_labels)

            ax = cmatplot.ax_
            ax.set_title(
                'Confusion Matrix of ' + model_info[model]['name'] + ' at location ' + site.stem +\
                '.\n' + 'Mean error is %.2fcm, and Accuracy is %.2f%%.' % (distmean, 100*accuracy)
                ,
                fontsize=100)
            ax.set_xlabel('Predicted Label', fontsize=70)
            ax.set_ylabel('True Label', fontsize=70)
            plt.savefig(str(scoredir) + '/' + model + '.png', dpi=100, fontsize=70)
            plt.close('all')
            print(model, 'confusion matrix plotted.')
```

Listing A.1: 'measure-scores.py'

```python
# plot-availability.py

from pathlib import Path
import pandas as pd
from lib import plot
import config

for site in [Path(x) for x in Path(config.path_data).iterdir() if x.is_dir()]:
```

```python

    plotdir = Path(str(site).replace(config.path_data, config.path_plots))
    if not plotdir.exists():
        plotdir.mkdir()

        for hdfFile in site.glob('*.h5'):
            print(hdfFile)

            csi = pd.read_hdf(hdfFile, key='all')
            csi_np = csi.to_numpy()

            plot.csi(csi_np,
                title='%d CSI samples at location (%s,%s) on site %s.' % (
                    csi.shape[0],
                    hdfFile.stem.split('-')[0],
                    hdfFile.stem.split('-')[1],
                    site.stem
                ),
                filepath=str(hdfFile).replace(config.path_data, config.path_plots).replace('.h5', '.png')
            )
```

Listing A.2: 'plot-availability.py'

```python
# pre-process.py
'''
Pre-process
===========

Convert .pcap files to hdf5
files. Acts on all sites in
config.path_pcap, except if
they are already in config.
path_data.
'''

from pathlib import Path
import pandas as pd
import numpy as np
from lib import pcap
```

```
from lib import plot
import config

for site_pcap in [Path(x) for x in Path(config.path_pcap).iterdir() if x.is_dir()
]:
    site_hdf = Path(str(site_pcap).replace(config.path_pcap, config.path_data))
    site_hdf_nonfil = Path(str(site_pcap).replace(config.path_pcap, config.
path_data) + '-nonfil')

    if not site_hdf.exists():
        site_hdf.mkdir()
        site_hdf_nonfil.mkdir()

        for file_pcap in site_pcap.glob('*.pcap'):
            print(file_pcap)

            csi = pd.DataFrame(pcap.read_csi(str(file_pcap)))

            nullsubcarriers = [x+64 for x in [-64, -63, -62, -61, -60, -59, -1,
0, 1, 59, 60, 61, 62, 63]]
            csi = csi.drop(columns=nullsubcarriers)

            pilotsubcarriers = [x+64 for x in [11, 25, 53, -11, -25, -53]]
            csi = csi.drop(columns=pilotsubcarriers)

            csi[csi > 4000] = 0

            file_hdf_unfil = str(file_pcap).replace(config.path_pcap, config.
path_data).replace('.pcap', '.h5')
            file_hdf_unfil = file_hdf_unfil.replace(str(site_hdf), str(
site_hdf_nonfil))
            csi.to_hdf(file_hdf_unfil, key='all')

            # Calculate Distance from Mean of CSI
            csi_distfrommean = ((csi - csi.mean())**2).sum(axis=1).sort_values(
ascending=True)

            # Remove 1/3 of the samples based on distance from mean.
            csi_fil = csi[csi.index.isin(csi_distfrommean[:8000].index)]
```

```python

            # Average every 4 consecutive samples.
            csi_fil = csi_fil.groupby(np.arange(len(csi_fil))//4).mean()

            file_hdf = str(file_pcap).replace(config.path_pcap, config.path_data).replace('.pcap', '.h5')
            csi_fil.to_hdf(file_hdf, key='all')
    else:
        print('Skipping existing location', site_hdf)
```

Listing A.3: 'pre-process.py'

```python
sites = [
    'nlos-b',
    # 'los-a'
]

path_pcap = '../thesis-pcap'
path_data = '../thesis-dataset'
path_scores = '../thesis-scores'
path_models = '../thesis-models'
path_plots = '../thesis-plots'

channel = 36
bandwidth = 40
macid = "34:e8:94:bd:e1:cc"

def confirm(name):
    '''
    Ask for confirmation before potentially
    destructive activities.
    '''
    confirmation = input('This overwrites data. Type %s to proceed: ' % (name))
    if confirmation != name:
        print('Exiting.')
        exit(0)
```

Listing A.4: 'config.py'

```python
import pandas as pd
from sklearn.model_selection import train_test_split
```

```python

def read_dataset(hdf_files):
    df_s = []
    for hdf_file in hdf_files:
        df = pd.read_hdf(hdf_file, key='all')
        df['loc'] = hdf_file.stem

        df_s.append(df)

    df = pd.concat(df_s)
    return (df.drop(columns=['loc']), df['loc'])


def split_dataset(hdf_files, ratio=0.2, random_state=32):
    Xtrain_s, Xtest_s, ytrain_s, ytest_s = [], [], [], []
    for hdf_file in hdf_files:
        df = pd.read_hdf(hdf_file, key='all')
        df['loc'] = hdf_file.stem

        X = df.drop(columns=['loc'])
        y = df['loc']

        Xtrain, Xtest, ytrain, ytest = train_test_split(X, y,
                                    test_size=ratio,
                                    shuffle=False,
                                    random_state=random_state
                                )

        Xtrain_s.append(Xtrain)
        ytrain_s.append(ytrain)
        Xtest_s.append(Xtest)
        ytest_s.append(ytest)

    Xtrain = pd.concat(Xtrain_s)
    ytrain = pd.concat(ytrain_s)
    Xtest  = pd.concat(Xtest_s)
    ytest  = pd.concat(ytest_s)

```

```
    return (Xtrain, ytrain, Xtest, ytest)
```

Listing A.5: 'lib/hdf.py'

```
'''
pcap
====

Provides fast methods to read CSI data from .pcap files.
'''

__all__ = [
    'read_csi'
]

import os
import numpy as np

def _read_csi_next(pcapfile, csi_size):
    """
    Note: Designed for internal use only.

    Parameters
    ----------
        pcapfile : File Object
        csi_size : Expected length of CSI in bytes. NFFT * 4
    """

    # Read Frame Size
    pcapfile.seek(8, os.SEEK_CUR)
    frame_size = int.from_bytes(
        pcapfile.read(4),
        byteorder = 'little',
        signed = False
    )

    # Skip some stuff
    pcapfile.seek(56, os.SEEK_CUR)

    # Read CSI data
```

```
    pcapfile.seek(8, os.SEEK_CUR)
    csi = np.frombuffer(
        pcapfile.read(csi_size),
        dtype = np.int16,
        count = int(csi_size / 2)
    )

    # Skip any zero-padding
    pcapfile.seek((frame_size - csi_size - 60), os.SEEK_CUR)

    return csi

def read_csi(pcap_file_path):
    """
    Read CSI data from PCAP file.
    Supports only 40MHz bandwidth,
    and only one Mac ID. You have
    to remove null subcarriers
    yourself.

    Parameters
    ----------
        pcap_file_path : str
    """

    bandwidth = 40

    NFFT = int(bandwidth * 3.2) # Number of channels in FFT
    chunksize = 1024

    csi  = np.zeros((chunksize, NFFT * 2), dtype = 'int16')

    with open(pcap_file_path, 'rb') as pcapfile:
        filesize = os.stat(pcap_file_path).st_size
        pcapfile.seek(24, os.SEEK_SET)

        npackets = 0
        while pcapfile.tell() < filesize:
            if not (npackets % chunksize):
```

```
                csi = np.vstack((csi, np.zeros((chunksize, NFFT * 2), dtype = '
    int16')))

            csi[npackets] = _read_csi_next(pcapfile, NFFT * 4)

            npackets += 1

    # Convert CSI complex numbers to Magnitude.
    csi_converted = np.abs(
        np.fft.fftshift(csi[:npackets, ::2] + 1.j * csi[:npackets, 1::2], axes
    =(1,))
    )

    return csi_converted
```

Listing A.6: 'lib/pcap.py'

```
'''
plot
====

Provides convenient plotting methods for CSI analysis.

hist : Plot histograms.
csi  : Efficiently plot one or more CSI samples.
multiple : Efficiently plot multiple lines in one plot.
'''

__author__      = 'Aravind Reddy V'
__email__       = 'aravind.reddy@iiitb.org'
__copyright__   = 'Copyright 2020, Aravind Reddy V'
__license__     = 'MIT'


__all__ = [
    'hist',
    'csi',
    'multiple',
    'simple'
]
```

```

import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
from matplotlib.collections import LineCollection

def simple(y):
    '''
    Plotting tool for simple data.

    The hope is, you'd never plot this way.
    So no saveplot functionality is provided.
    '''
    x = np.arange(0, y.size)
    x = x - 64

    fig, ax = plt.subplots()

    fig.set_size_inches(16, 10)

    ax.set_xlabel("Subcarrier", fontsize=20)
    ax.set_ylabel("Amplitude", fontsize=20)
    ax.tick_params(axis='both', which='major', labelsize=16)
    ax.tick_params(axis='both', which='minor', labelsize=18)
    ax.set_title('CSI', fontsize=20)

    ax.plot(x, y)

    plt.autoscale(enable=True, axis='both', tight=None)
    plt.savefig('./simple.png')
    plt.close('all')

def hist(data, title, savepath, nbins=100):
    '''
    Plot a histogram of given data.

    Parameters
    ----------
        data : Row vector of ints.
```

```
    '''
    plt.hist(data, align='mid', bins=nbins)

    plt.title(title)
    plt.savefig(savepath, dpi=100, fontsize=20)
    plt.close('all')

def multiple(x, ys, title, savepath, xlabel, ylabel):
    '''
    Efficiently plot several lines in one plot.
    '''

    lines = LineCollection([np.column_stack([x, y]) for y in ys])
    lines.set_array(x)

    fig, ax = plt.subplots()
    fig.set_size_inches(16, 10)

    ax.add_collection(lines)
    ax.set_xlim(np.min(x), np.max(x))
    ax.set_ylim(np.min(ys), np.max(ys))
    ax.set_xlabel(xlabel, fontsize=20)
    ax.set_ylabel(ylabel, fontsize=20)
    ax.tick_params(axis='both', which='major', labelsize=16)
    ax.tick_params(axis='both', which='minor', labelsize=18)
    ax.set_title('2000 CSI samples', fontsize=20)

    # nullsubcarriers = [-64, -63, -62, -61, -60, -59, -1, 0, 1, 59, 60, 61, 62,
    63]
    # for ns in nullsubcarriers[:-1]:
    #     plt.axvline(x=ns, c='r')
    # plt.axvline(x=nullsubcarriers[-1], label='Null Subcarriers', c='r')

    # pilotsubcarriers = [11, 25, 53, -11, -25, -53]
    # for ps in pilotsubcarriers[:-1]:
    #     plt.axvline(x=ps, c='k')
    # plt.axvline(x=pilotsubcarriers[-1], label='Pilot Subcarriers', c='k')

    # plt.legend(loc='upper right', fontsize=18)
```

```
    plt.savefig(savepath, dpi=100, fontsize=20)
    plt.close('all')

def csi(data, title, filepath):
    '''
    Efficiently plot several CSI samples in one plot.

    Parameters
    ----------
        data : Should be a numpy array of dim (m, NFFT), and m >= 1.

    Examples
    --------
    >>> import plot
    >>> plot.csi(csi, 'CSI data', './csi.png')
    >>> plot.csi(np.array([csi[0]]), 'Single CSI sample', './csi.png')
    '''
    nfft = int(data.shape[1])

    x = np.arange(-1 * nfft/2, nfft/2)

    multiple(x, data, title, filepath, xlabel="Sub-carrier", ylabel="Amplitude")

def crossvalidation(df, model, title, filepath):

    ax = df.plot(
        colormap='jet',
        title=title,
        fontsize=20,
        figsize=(16, 10),
        y=['original order', 'sorted']
    )

    ax.set_title(title, fontsize=20)
    ax.set_xlabel('Index of AP', fontsize=20)
    ax.set_ylabel('Crossvalidated Accuracy', fontsize=18)
    ax.tick_params(axis='both', which='major', labelsize=16)
    ax.tick_params(axis='both', which='minor', labelsize=18)
```

```
    ax.legend(prop={'size': 20})

    plt.gca().lines[0].set_alpha(0.3)

    plt.autoscale(enable=True, axis='both', tight=None)
    plt.savefig(filepath)
    plt.close('all')

def stat(statistic, title, filepath, xmin=20, xmax=100):
    statistic['trainpercent'] = range(1, 100)
    cropped = statistic[statistic['trainpercent'].isin(range(xmin, xmax+1))]

    ax = cropped.plot(
        x='trainpercent',
        y=['nbc', 'svc', 'dtc', 'rfc'],
        colormap='jet',
        title=title,
        fontsize=20,
        figsize=(16, 10)
    )

    ax.set_title(title, fontsize=20)
    ax.set_xlabel('Percentage of Dataset used for Training', fontsize=20)
    ax.set_ylabel('Accuracy', fontsize=20)
    ax.tick_params(axis='both', which='major', labelsize=16)
    ax.tick_params(axis='both', which='minor', labelsize=18)
    ax.legend(prop={'size': 20})

    if title=='Standard Deviation':
        ax.set_ylabel('Standard Deviation of Accuracy', fontsize=20)

    plt.autoscale(enable=True, axis='both', tight=None)
    plt.savefig(filepath)
    plt.close('all')
```

Listing A.7: 'lib/plot.py'