Data Collection and Classification of Wireless Sensing for Robotics Applications

ZIHAO GONG* and KELVIN DUONG*, UC San Diego, USA

WiFi sensing is a promising solution to overcome the pitfalls of traditional robotics sensors like cameras and lidar. Despite its potential benefits in robotics, the Robot Operating System (ROS) lacks robust support WiFi sensing. We build upon WiROS [1] to make wireless more accessible for widespread use in robotics. We do this by developing a large-scale data collection system and develop models to classify received data. We tested 3 methods (thresholding, suppport vector machine (SVM), and convolutional neural networks (CNN))to classify our collected data and measured their performance on different smoothing windows. We saw good performance from the SVM and thresholding.

ACM Reference Format:
Zihao Gong and Kelvin Duong. 2024. Data Collection and Classification of Wireless Sensing for Robotics Applications. 1, 1 (March 2024), 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

As IoT devices and smart environments become increasingly prevalent, the demand for effective robot sensing methods is on the rise. Conventional approaches like cameras and infrared sensors, with their fixed field of view, entail substantial deployment and maintenance complexities [2] for wide scale coverage. Moreover, the acquisition of visual data raises significant privacy concerns [3]. WiFi sensing using CSI information has been proposed as a promising solution to these issues [4]. WiFi signals, which are widely present in most indoor environments, offer an extended sensing range for robots. Many papers [5][6] begin to utilize WiFi signals for improving robot common failures, such as visual occlusion, dynamic lighting, or perceptual aliasing.

Despite the potential benefits, integrating WiFi sensors into an existing robot's sensor stack poses challenges. The Robot Operating System (ROS) framework currently lacks robust support for WiFi sensors [1], hindering their widespread adoption in robotics applications.

Our project builds upon WiROS, a framework designed to facilitate the integration of WiFi sensors into robotics environments. Our primary goal is to build and debug a large-scale data collection system for wireless sensing and develop automated classification tools. In doing this, we make WiFi sensing more accessible and aid in its future research for robotics applications. The project is proposed to involve two major steps:

(1) Bring up test equipment setup and collect real-world data.
(2) Develop models to classify the AoA-ToF data.

We tested 3 methods thresholding, SVM, and CNN to classify our collected data and measured their performance on different smoothing windows. We saw good performance from the SVM and recommend it along with thresholding.

- Simple threshold method
- Support Vector Machine (SVM) model
- Convolutional Neural Network (CNN) model

The SVM model can achieve the highest 89.8% accuracy, which outperforms other two methods.

2 BACKGROUND AND RELATED WORKS

Current WiFi measurement tools have several drawbacks. The existing systems for integrating WiFi measurements into robot applications have limitations in meeting crucial design principles such as accuracy, tractability (easy to bring up and calibrate), and versatility (usable in most WiFi environment).

These systems frequently do not provide real-time access to WiFi measurements, depend on outdated protocols such as [7][8], and offer only basic measurements like Received Signal Strength Indication (RSSI). As a result, their effectiveness in seamlessly integrating into robot sensor stacks is hindered, restricting their utility in diverse robotics applications

To overcome above issue, we can integrate WiFi measurements into ROS systems. They offer seamless integration with the ROS framework. For instance, [9] utilize a fine-grained information (CSI) to measure AoA and velocity of WiFi sensors. Meanwhile, a recent work[10] provide ROS support for the 802.11n chipsets. However, these systems relies on active collaboration with the WiFi infrastructure, requiring deployed WiFi Access Points (APs) to engage in 'round-robin' packet exchanges. This dependency on infrastructure hampers the widespread deployment of WiFi sensing in indoor environments and introduces logistical, security, and networking challenges.

As a result, WiROS built on [11] utilize the 802.11ac protocol and have two different parts. It contains a CSI node for data collection and a Feature-Extraction node for data processing. The CSI node can extract diverse physical parameters, such as RSSI, angle of arrival (AoA) to help robot manipulation.

However, sometimes we might receive bad packets. For instance, we might receive a signal with very low RSSI which can make it more difficult to extract useful information. Additionally, we might receive a signal with widely varying magnitude or phase across frequency subcarriers indicating strong multipath or signal attenuation. This leads to noisy AoA-ToF plots and can greatly decrease

*Both authors contributed equally to this research.

Authors' address: Zihao Gong, z4gong@ucsd.edu; Kelvin Duong, k6duong@ucsd.edu, UC San Diego, La Jolla, CA, USA.

ACM XXXX-XXXX/2024/3-ART
https://doi.org/10.1145/nnnnnnn.nnnnnnn

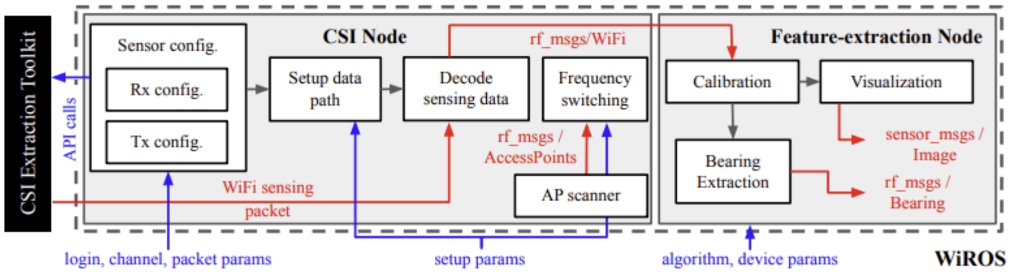


Fig. 1. WiROS extracts and processes CSI measurements in 2 stages. The blue text indicates the control plane parameters, whereas the red text indicates the exposed measurements. WiROS extends the functionality of the underlying black box 'CSI Extraction Toolkit.' Used with permission from Hunter, Arun [1]

the accuracy of the AoA predictions. These inaccuracies pose significant challenges, especially in real-time robotics applications as it can trigger unintended actions based on corrupted data. Consider an autonomous navigation scenario, a robot is moving towards an obstacle and needs to correct its course to avoid it.

If the received signal is noisy, the calculated AoA may be incorrect, causing the robot to inadequately correct its trajectory and potentially collide with the obstacle. This highlights the critical need for robust data validation and error correction mechanisms to ensure the reliability of information used in real-time robotics applications. We aim to develop a classification system for WiROS to identify and reject unreliable data, thereby facilitating the creation of a dependable dataset.

3 DESIGN

3.1 Data Collection and Processing

For our data collection and processing, we use WiROS, a WiFi sensing toolbox for robotics [1]. WiROS utilizes Nexmon to extract exposed Channel State Information (CSI) measurements and publishes them into ROS topics for further analysis, as depicted in Figure 2. We filter the data by channel, bandwidth, and MAC address to isolate our desired testing device. The collected data is saved using `rosbag -record` and sent to the Feature-extraction Node for visualization. WiROS implements several algorithms for calculating Angle of Arrival (AoA), such as Multiple Signal Classification[12] (MUSIC) and Subspace Pursuit for Fast Imaging (SPOTFI)[13]. We opt to use the 2D-FFT algorithm as it is less computationally intensive than the others, making it a suitable option for real-time applications. The node then generates a bearing-range likelihood profile and exports the data as a .mat file for further classification in MATLAB.

3.2 Experimental Setup

In our our experimental setup, a mobile phone serves as the transmitter, while a communication Access Point (AP) functions as the receiver as seen in Figure 2. The mobile phone initiates communication by sending packets, or pinging, to the communication AP. Simultaneously, the laptop, acting as a central control unit, is connected to the same communication AP. The Angle of Arrival (AoA) AP, strategically positioned to capture the transmitted packets, reads and extracts the exposed measurements from the phone's ping. Leveraging these measurements, the AoA AP calculates the Angle of Arrival (AoA) of the phone relative to its position. To enhance sensing accuracy, we opt for an Access Point (AP) equipped with four RX antennas, favoring it over alternatives featuring three antennas.

Following data acquisition, these measurements are transmitted to a laptop running WiROS via an Ethernet connection.This setup enables the real-time assessment of signal propagation characteristics and facilitates the analysis of wireless communication performance in dynamic environments.

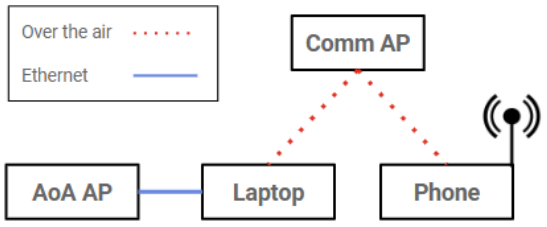


Fig. 2. A communication AP is set up to receive packages over the air from the phone. The AoA AP will read the exposed measurements from the phone's broadcast and send to the laptop via Ethernet

3.3 Configuration

We will briefly go over the setup procedure for setting up the CSI node, more detailed instructions can be found in WiROS github [1]. To setup AoA AP, plug into an outlet and perform a factory reset. Use an Ethernet cable to connect your computer or laptop to one of the LAN ports on the AoA AP, configure the ethernet interface to DHCP. Use router.asus.com to setup the AP. Manually assign the IP address of the AP (e.g. 192.168.43.xxx) and create a new ethernet connection profile on the computer with the same subnet (e.g. 192.1268.43.yyy). The AP will reboot, switch the laptop from DHCP to the new connction profile (192.1268.43.yyy). Enter the AP's IP address (192.168.43.xxx) to go the AP's network settings. Navigate to Administration->System->Service and set enable SSH with a timeout of 0. Once the AP and laptop are configured, install ROS and Nexmon CSI using the commands provided in the github repository [1]

To setup the communication AP, plug into an outlet and perform a factory reset. Open the Wi-Fi settings on the laptop and search for available networks. You should see the default Wi-Fi network broadcasted by your router. Connect to this network. Open a web browser and enter the default IP address of the router (e.g., 192.168.50.1) into the address bar. Follow the on-screen instructions to set up basic configurations such as the administrator password, time zone, and language. Open the Wi-Fi settings on the phone and connect to the communication AP's network.

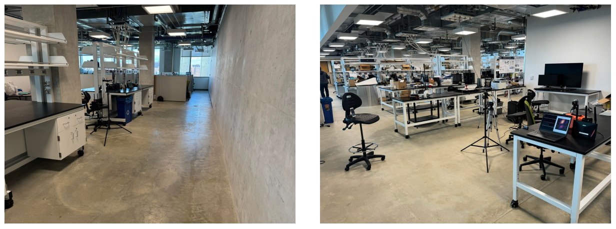
Optionally, you can collect compensation data. Due to differing wire lengths between setups each RX chain may have varying bias per subcarrier. To accurately measure AoA these biases need to me measured and removed. Because our project involves classifying received data based on signal quality it is not necessary to apply compensation. We already had compensation files available for our setup so we opted to use it. This data was collected by injecting the same signal into all receivers via a 4-to-1 power splitter. The data was then recorded and sent to a script to calculate compensation in the processing node.

3.4 Data Collection

Once the receiver and transmitter are set up, you can start collecting data. Update the channel parameters (channel, bandwidth, MAC address) in `basic.launch` and run using `roslaunch`. Ping the communication AP from the mobile device in Termux to start sending packets. The packets measured from the AoA AP will start to show in the terminal window. You can then use `rosbag -record` to save the data for future analysis. Rosbag will save the data in a `.bag` file. Move the phone around the room to simulate variations in signal strength, phase, and multipath effects. We collected data from 3 diverse locations in Frankin Antonia Hall as seen in Figure 3. By sampling from different areas, it enables us to assess the robustness of our classification system across different environments. We moved the phone on a rolling lab chair to keep the height consistent between locations. The default ping frequency is 1 packet/second, we measured packets for around 3-5 minutes at each location for a total of 815 packets.

You can visualize the data in real-time using the feature-extraction node. This is an optional step, but it can be helpful for confirming that data collected is good. Run `visualize_profiles.launch` with `roslaunch` to display the bearing-likelihood profile. The feature-extraction node lets you set smoothing window and RSSI thresholds, these values will be overwritten in post processing so you can just use the defaults provided.

You can use `generate_channel_files.py` in the feature extraction node to convert the `.bag` data to a `.mat` file for analysis in Matlab. When exporting data to Matlab, you can modify the smoothing window for the data points. The choice of the smoothing window size directly impacts the level of smoothing applied to the data. A larger window size results in more aggressive smoothing, which can blur out fine details and potentially oversimplify the profile. On the other hand, a smaller window size preserves more detailed information but may not effectively eliminate noise or fluctuations. We will analyze the effects of the smoothing window and weigh the trade-off of speed and accuracy.


(a) (b)

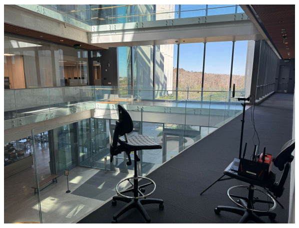

(c)

Fig. 3. Test environments in Frankin Antonio Hall: (a) empty hallway, (b) cluttered work station, and (c) open building

3.5 Data labeling

We classified our data by looking at the bearing-likelihood profiles in Matlab. We ran a simple script to show images one at a time and ranked each image 1-4, 1 represents a bad profile while 4 represents a good profile as shown in Figure 4. Characteristics of a bad profile include scattered or diffused distributions without distict peaks. This suggest uncertainty in the AoA-ToF, bad packets will be rejected to insure reliability of information. Conversly, characteristics of a good packet include clear, well defined, and distinct peaks. This indicates a strong signal with limited multi path interference or attenuation. This will output reliable AoA-ToF estimations.

4 IMPLEMENTATION

For our project, we try three different methods to classify the AoA-ToF profile 2D data.

4.1 Simple Threshold Method

Figure 5 illustrates the comparison between the original and processed AoA-ToF 2D profiles. In instances of good profile data, distinct clusters are discernible within the 2D profile. To filter the original data, a manual threshold value of 0.6 is applied, followed by the utilization of the image processing toolbox to identify clusters with high magnitudes.

Once the processed 2D profile is obtained, the area ratio of the bright area relative to the entire image area is calculated. It is anticipated that a superior AoA-ToF profile will exhibit a lower area ratio. For example, in the first row, indicative of a good profile data, a single cluster is obtained post-filtering, while in the second row, two clusters are discerned, resulting in a larger area ratio.

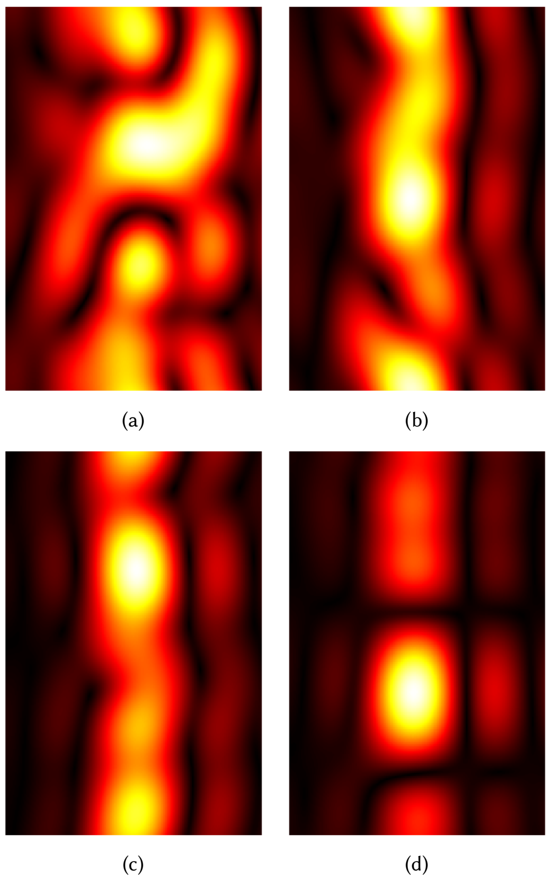


Fig. 4. Examples of each rank: (a) 1, (b) 2, (c) 3, (d) 4

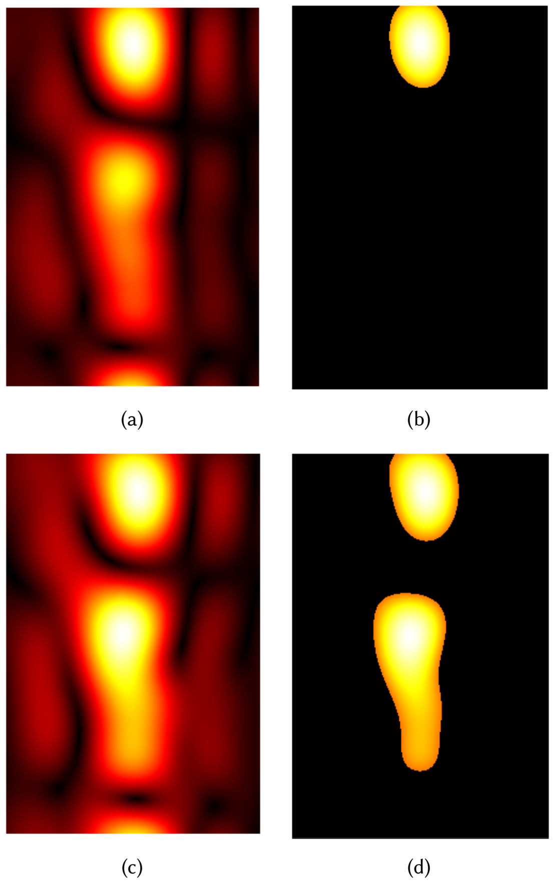


Fig. 5. Filtered 2D profile image examples

4.2 SVM Model

Support vector machine (SVM) is a classification technique where the classes are separated by finding a linear hyperplane that maximizes the minimum distance from the plane to the points in the classes. Maximizing the distance ensures that the training data is classified with maximum accuracy and confidence. Fig.6 illustrates two classes - crosses and dots separated by the optimal hyperplane. The two dotted lines passing through the closest point to the plane are called the support vectors. The parameters for the hyperplane w, b are obtained in the training phase by solving the optimization problem in (1) under the constraints (2) and (3).

$$\max_{\gamma, w, b} \gamma \tag{1}$$

$$s.t: \quad y^i(w^T x^i + b) \geq \gamma, \; i = 1, 2, ..., m \tag{2}$$

$$\|w\| = 1 \tag{3}$$

The quadratic nature of the constraint (3) requires quadratic programming tools to solve. MATLAB supports sequential minimal optimization and various kernel methods to solve fit the hyperplane.

For our project, we simply build a classic SVM model utilizing three features below as our input:

- Area ratio: calculated by simple threshold method
- isMultiCluster: a bool variable indicating whether a 2D profile image has multiple clusters after filtering
- Normalized RSSI

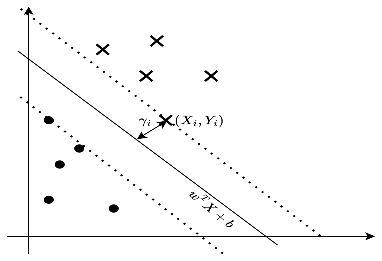


Fig. 6. Illustration of SVM: The two classes are separated by an optimal hyperplane

To assess whether a 2D profile exhibits multiple clusters, we analyze the two clusters with the largest areas and compare their average magnitudes. For example, in Fig. 5 (a), two clusters are present, but the cluster in the center displays a lower average magnitude compared to the upper cluster. Consequently, we can establish a threshold for the magnitude difference to filter out clusters with lower magnitudes.

In the case of Fig. 5 (c) and (d), both clusters exhibit similar average magnitudes, indicative of poor data quality likely due to multipath effects. Hence, we anticipate that a well-filtered 2D profile will display a lower area ratio, consist of only one cluster, and possess higher RSSI values.

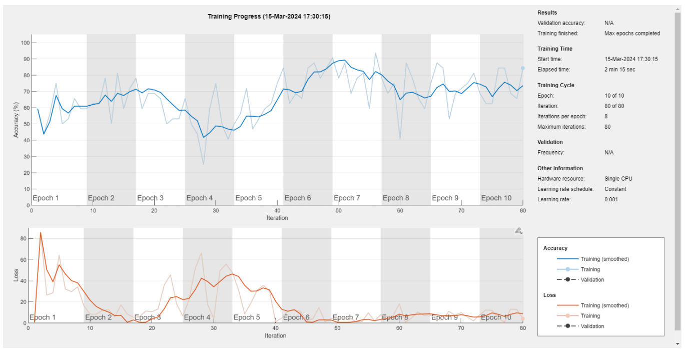
4.3 CNN model

Neural networks, especially convolutional neural networks (CNNs), have emerged as powerful tools for image classification tasks, exhibiting impressive performance across various domains. Given the inherently visual nature of our AoA-ToF 2D profiles, which can be conceptualized as images, we aim to leverage the strengths of CNNs by constructing a simple yet effective CNN model for classification purposes.

Ultimately, our goal is to develop a CNN-based classification system capable of accurately categorizing AoA-ToF 2D profiles as good or bad.

Our CNN model is defined as follows.

- Input Layer: Filtered 2D profile image with size 360*240
- Convolutional Layers:Three convolutional layers with 3x3 filters.
- Number of filters: 16, 32, and 64 respectively.
- Padding: ’same’ to maintain spatial dimensions. Followed by batch normalization and ReLU activation.
- Pooling Layers:Two max-pooling layers with 2x2 pool size and stride 2. Fully Connected Layer:
- Single fully connected layer with 4 neurons.
- Output Layers: Softmax layer for class probabilities and classification layer for final classification.



Results
Validation accuracy: N/A
Training finished: Max epochs completed

Training Time
Start time: 15-Mar-2024 17:30:15
Elapsed time: 2 min 15 sec

Training Cycle
Epoch: 10 of 10
Iteration: 80 of 80
Iterations per epoch: 8
Maximum iterations: 80

Validation
Frequency: N/A

Other Information
Hardware resource: Single CPU
Learning rate schedule: Constant
Learning rate: 0.001

Fig. 7. Training Curve for CNN model

The model is trained using the Adam optimizer with a maximum of 10 epochs and a mini-batch size of 32. During training, the network learns to minimize the categorical cross-entropy loss between the predicted and actual labels.

After training, the performance of the CNN model is evaluated using a separate test dataset. The trained model is used to predict labels for the test images, and the accuracy is computed and shown in Section 5.

5 EVALUATION

The table 1 presents the accuracy of three methods. The SVM model exhibits superior performance compared to the other two models, attributed to the generation of three effective features for classification.

Despite the CNN model’s relatively lower performance, there remains potential for enhancement through preprocessing of the input 2D profile data. Presently, the CNN model’s performance is hindered by simplistic filtering techniques applied to the 2D data, leading to significant noise in certain data points and subsequent degradation in performance. The model could also benefit from a larger training set.

Method	SW5	SW20
Simple Threshold	70.4%	70.8%
SVM	72.4%	89.8%
CNN	60.1%	67.1%

Table 1. Model Comparison

We see that across the board the data with a smoothing window of 20 had higher accuracy than the smoothing window of 5. This is expected as the larger smoothing window helps reduce noise and fluctuations in data resulting in more accurate classification. We can see that the smoothing window had the biggest impact on the SVM model and the least impact on the thresholding model. This makes sense as the SVM relies heavily on the separation of data points into distinct classes based on a hyperplane. When the data is smoothed it can help define stronger boundaries between classes resulting in a massive improvement. The CNN benefited moderately from the smoothing window. This also makes sense as the CNN is inherently robust to noise as it learns hierarchical representations of data. CNNs can adapt to variations and extract features in the presence of noise. While it still benefits from a reduction in noise, it has a less significant impact compared to the SVM. Lastly the simple thresholding model benefited very little from the smoothing window. This makes sense as it is just a binary classification method that doesn’t consider nuanced variations in the data. As a result, it is less sensitive noise sees less of an improvement than the other methods.

6 CONCLUSION

During this project, our exploration of the AoA-ToF profile in a real testbed has provided valuable insights into the functioning of wireless sensing in practical scenarios. We tested 3 methods of classification and saw that the SVM produced the strongest result. We saw the effects of smoothing window and how it improved accuracy especially in the SVM. For offline applications such as data collection we would recommend the SVM with a smoothing window of 20. This provides the highest accuracy across the models we tested. For real time applications such as autonomous driving we would recommend simple thresholding with a smoothing window of 5. Simple thresholding has the lowest computational complexity across the models without a major reduction in quality compared to a smoothing window of 20 or a more complex model.

Moving forward, our future endeavors will focus on the following objectives:

- Data collection: Due to time limitation, we only collect 815 data images and label them by our eyes. With a larger dataset would could improve the accuracy with more training samples.
- Optimization of Data Processing: Developing more sophisticated algorithms for data processing and analysis to extract meaningful information from the AoA-ToF profiles. Meanwhile, we fail to utilize the ranked label for our dataset, so we can probably to find a better model incorporate different weight for different data images.
- Integration with more complex ML techniques: We only try very basic model in ML, so we can probably implement a more complex ML mdoel to improve the performance.

7 ACKNOWLEDGEMENTS

We gratefully acknowledge Aditya Arun and William Hunter for their invaluable contributions to this research. Their provision of hardware resources, including the Access Point (AP), laptop, and mobile phone, was instrumental in enabling our test setup and data collection. Their support and assistance greatly facilitated the execution of this study, and we extend our sincere appreciation for their generosity and collaboration.

We would also like to extend our gratitude to ChatGPT for its invaluable assistance during the debugging phase of our project. ChatGPT's guidance and support were instrumental in resolving syntax issues encountered in MATLAB and LaTeX.

8 TEAM CONTRIBUTIONS

8.1 Kelvin

In this project, each team member made significant contributions to its success. Specifically, I focused on several key aspects, including the setup of hardware components, data collection, accurate data labeling, and active participation in the development of code for various functionalities. My involvement in these tasks ensured the smooth execution of the project and contributed to achieving our research objectives effectively. Additionally, I collaborated closely with Zihao to exchange ideas, troubleshoot issues, and enhance the overall project outcomes. I would like to give thanks to Zihao for taking charge of the software development and working with on this project. I am grateful for the opportunity to contribute to this collaborative endeavor.

8.2 Zihao

In this project, my primary role involves developing classification models using MATLAB, encompassing tasks such as data processing, data labeling, and model design. Specifically, I focus on transforming raw data from AoA-ToF 2D profiles into a format suitable for analysi. Additionally, I undertake the crucial task of assigning accurate labels to the data, facilitating supervised learning and model training.

Moreover, I lead the design and implementation of classification models, leveraging MATLAB's capabilities to create effective algorithms tailored to our project's objectives. Through meticulous parameter tuning and architectural refinement, I strive to optimize model performance and enhance classification accuracy.

I extend my gratitude to Kelvin for his invaluable support in data collection and hardware setup. His expertise and assistance have been instrumental in overcoming technical challenges and ensuring the smooth operation of our experimental setup. Additionally, I appreciate the contributions of Aditya and William, who provided valuable assistance in configuring the hardware.

Overall, my contributions in data processing, labeling, and model development play a pivotal role in advancing the project's objectives. Through collaborative teamwork and dedication, we aim to achieve meaningful insights and impactful outcomes in our classification endeavors

REFERENCES

[1] William Hunter, Aditya Arun, and Dinesh Bharadia. Wiros: Wifi sensing toolbox for robotics. *arXiv preprint arXiv:2305.13418*, 2023.
[2] Zheng Yang, Yi Zhang, Guoxuan Chi, and Guidong Zhang. Hands-on wireless sensing with wi-fi: A tutorial. *arXiv preprint arXiv:2206.09532*, 2022.
[3] Chien-Cheng Lee and Xiu-Chi Huang. Human activity detection via wifi signals using deep neural networks. In *Proceedings of the IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion)*, pages 3–4, 2018.
[4] First Author and Second Author. Title of the paper. *arXiv*, 2022.
[5] Tara Boroushaki, Laura Dodds, Nazish Naeem, and Fadel Adib. Fusebot: Rf-visual mechanical search. In *Proceedings of the Robotics: Science and Systems (RSS)*, 2022.
[6] Aditya Arun, Roshan Ayyalasomayajula, William Hunter, and Dinesh Bharadia. P2slam: Bearing-based wifi slam for indoor robots. *IEEE Robotics and Automation Letters (RA-L)*, 7(2):3326–3333, 2022.
[7] Daniel Halperin, Wenjun Hu, Anmol Sheth, and David Wetherall. Tool release: Gathering 802.11n traces with channel state information. *ACM SIGCOMM Computer Communication Review (CCR)*, 41(1):53–53, 2011.
[8] Yaxiong Xie, Zhenjiang Li, and Mo Li. Precise power delay profiling with commodity wifi. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking (MobiCom)*, pages 53–64, 2015.
[9] Yingying Ma, Guoliang Zhou, and Shuo Wang. Wifi sensing with channel state information: A survey. *ACM Computing Surveys (CSUR)*, 52(3):1–36, 2019.
[10] Nikhil Jadhav, Wei Wang, Dongming Zhang, Shantanu Kumar, and Santiago Gil. Toolbox release: A wifi-based relative bearing framework for robotics. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 13714–13721, 2022.
[11] Francesco Gringoli, Matthias Schulz, Jan Link, and Matthias 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)*, pages 21–28, 2019.
[12] Quanfu Zheng, Lingen Luo, Hui Song, Gehao Sheng, and Xiuchen Jiang. A rssi-aoa-based uhf partial discharge localization method using music algorithm. *IEEE Transactions on Instrumentation and Measurement*, 70:1–9, 2021.
[13] Manikanta Kotaru, Kiran Joshi, Dinesh Bharadia, and Sachin Katti. Spotfi: Decimeter level localization using wifi. In *Proceedings of the 2015 ACM conference on special interest group on data communication*, pages 269–282, 2015.