# RF POSE USER GUIDE (I2I-III)

**Arnab Saha**

In I2I-2, our team put effort toward creating a driver monitoring system that utilized RF-based human activity recognition to deter dangerous driving behaviors. It would emit continuous waves from the radar, capture the reflected waves, process the captured signals, and then a neural network would classify whether or not the driver was doing certain bad behaviors and it would have an audio output to notify them of their actions.

This User Guide takes a chapter-wise approach, guiding people step-by-step from the basic fundamentals of radar sensing, then providing context and details on the radar module and data streaming module used, followed by guidelines to aid the set-up of the radar and its software counterpart mmWave Studio before finally moving on to the exacts of our specific implementation, with our specific configuration details, codes used and pipeline. All of this is also being complemented with relevant texts and reading materials, which were used by our team for our implementation.

Therefore, the list of chapters includes:

1. Radar Fundamentals:

   This chapter provides an introduction to the fundamentals behind Radar sensing, the different terminologies one should be familiar with when working with a mmWave radar, and helps one understand how this correlates to sensing human activity. This chapter thus aims at making clear the concepts of FMCW Radars, and terminologies like a chirp, sampling rate, range resolution, and the significance of FFTs, and how it all gels into our implementation.

2. Human Activity Recognition with FMCW Radar:

   This chapter provides the inspiration behind our work, how radar sensing could be exactly be used for Human Activity Recognition, and the range of activities that are plausible to be detected.

3. Radar and Data Streaming Module:

   This chapter aims at familiarizing the reader with the hardware used for the I2I-II implementation, which includes the radar Module used, AWR1843BOOST, and its corresponding data streaming device, DCA1000EVM. With the aim of providing the context of what the functionalities are for both of these devices, we also aim at aiding towards setting up the radar module, and its accompanying GUI app, mmWave studio, and its own set-up.

4. mmWave Studio:

Provides a brief guideline, supported by the user guide for mmWave studio itself, to bring the reader up to speed with setting up the application in sync with the radar module, and provide a context of the configuration our team ended up using.

5. Data Collection:

   After setting up the radar and its GUI, and with the knowledge of the principle behind how mm-wave radar sensing of human activity works, the next step is collecting data for processing. Here, a suitable explanation for the custom automation code used, training set-up and conditions, data collection division amongst teams, and labeling of data is provided, with example images to provide the user a quick guide to being able to record their own dataset.

6. Pre-processing and Denoising:

   The final step, for radar-based Human Activity Recognition, would in fact be the actual processing of the data. Here, sufficient context and guidelines for using FFTs, and their respective codes are explained and essentially explain how to achieve time-doppler and range-angle-doppler plots from the recorded data. This section also deals with the problems faced by our team with Multipath propagation and demonstrates suitable denoising methodologies to deal with the same.

## CHAPTER 1: FUNDAMENTALS OF FMCW RADAR FOR HUMAN ACTIVITY RECOGNITION

Briefing using the excellent reading materials by Texas Instruments, in mmWave sensing, FMCW (Frequency Modulated Continuous Wave) radars are a common choice of radar technology. Using electromagnetic waves in the millimeter-wave frequency range for sensing and detection is known as millimeter-wave (mmWave) sensing. The normal range of mmWave frequencies is 30 to 300 GHz. High resolution, better precision, and the capacity to identify microscopic objects are just a few benefits of mmWave sensing. Continuously broadcasting a signal with a linearly modulated frequency sweep is how FMCW radars work. When a signal is transmitted, it is reflected off of nearby objects, and the receiver then collects these reflected signals for processing.

The FMCW radar equation provides a mathematical relationship between the transmitted power, received power, range to the target, and other parameters. The radar equation is given by:

$$Pr = (Pt * Gt * Gr * \lambda^2 * \sigma * Ae * R^4) / ((4\pi)^3 * L^2)$$

where:

- Pr is the received power at the radar receiver.
- Pt is the transmitted power.
- Gt is the transmit antenna gain.
- Gr is the receive antenna gain.
- $\lambda$ is the wavelength of the radar signal.
- $\sigma$ is the radar cross-section of the target.
- Ae is the effective aperture of the receiving antenna.
- R is the range of the target.
- L is the system loss factor.

The FMCW radar equation helps in understanding the relationship between various parameters and the received power level, which is crucial for signal detection and range estimation. Furthermore, the frequency of the transmitted signal in FMCW radars linearly rises or falls with time. The term "chirp" signal is frequently used to describe this frequency sweep. The resulting signal displays a beat frequency when the sent and received signals are combined in the receiver. The beat frequency, which reveals the target's range and relative velocity, is proportional to the difference in frequency between the emitted and received signals. The beat frequency (fb) can be calculated using the following formula:

$$fb = (2 * B * \Delta f) / c$$

where:

- B is the sweep bandwidth of the chirp signal.
- Δf is the change in frequency between the transmitted and received signals.
- c is the speed of light.

Similarly, FMCW radars can estimate the range of a target based on the time delay between the transmitted and received signals. The range (R) can be calculated using the formula:

$$R = (c * \tau) / (2 * B)$$

where:

- c is the speed of light.
- τ is the time delay between the transmitted and received signals.
- B is the sweep bandwidth of the chirp signal.

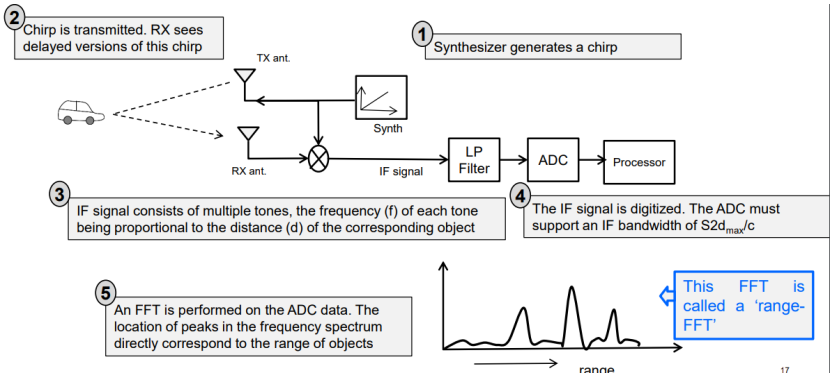
By measuring the time delay, FMCW radars can determine the distance between the radar and the target accurately.FMCW radars can also estimate the velocity of a target based on the Doppler effect. The Doppler frequency shift (fd) is proportional to the relative velocity between the radar and the target. The Doppler frequency shift can be calculated using the formula:

$$fd = (2 * v * f0) / c$$

where:

- v is the relative velocity between the radar and the target.
- f0 is the center frequency of the transmitted signal.
- c is the speed of light.

By measuring the Doppler frequency shift, FMCW radars can determine the velocity of the target.




Understanding the specific terminologies, the above figure essentially provides a perspective of how all of it comes together for achieving mm-wave sensing.

## CHAPTER 2: HUMAN ACTIVITY RECOGNITION WITH FMCW RADAR

The method of identifying and categorizing human actions using sensor data is known as human activity recognition (HAR). Frequency Modulated Continuous Wave Radar, sometimes known as FMCW Radar, is a potential HAR technology. It tracks and detects human movement using radar waves, which enables the identification of a variety of activities.

FMCW Radar is a good fit for HAR due to its many benefits. Non-contact sensing is made possible, protecting user privacy and convenience. FMCW Radar works effectively in a variety of settings and is resistant to obstructions and poor light. It offers great precision and resolution for in-depth analysis and a wide sensing range that covers large areas. Additionally, FMCW Radar can track numerous people at once, making it easier to recognize activity in group settings.

Based on the concepts of radar signal processing, FMCW Radar can identify human movement. To locate the target (a person), it measures the distance between the radar and the subject. Doppler processing determines the target's velocity by analyzing the frequency shift in the reflected signals brought on by movement. Real-time monitoring of multiple targets is possible thanks to Doppler-ambiguity resolution and multi-target tracking techniques.

HAR relies heavily on features collected from FMCW Radar data. Doppler frequency, which discloses the target's velocity, and time-of-flight, which establishes the range or distance, are important characteristics. Additionally, statistical metrics capture the temporal and spatial aspects of activities, whereas micro-Doppler fingerprints record distinctive patterns brought on by particular human movements.

FMCW Machine learning algorithms for HAR can be utilized with radar data and the retrieved features. Collecting training data entails acquiring tagged radar datasets of various human activities. The most essential features for precise activity recognition are found through feature selection. To categorize activities based on the chosen features, classification algorithms like Support Vector Machines (SVM), Random Forests, or Neural Networks are used. Metrics like accuracy, precision, recall, and F1-score are used to assess the HAR system's performance.

Setting up an FMCW Radar system for HAR requires the appropriate hardware, software, and configurations. The hardware includes the FMCW Radar sensor, antennas, and necessary peripherals. The software and programming environment should be installed and configured for data acquisition and processing. Configuration parameters, such as frequency sweep range, transmit power

and modulation, need adjustment to optimize the system for HAR. Proper radar system setup, including mounting and positioning, ensures effective coverage of the monitoring area. Visualization and interpretation are used to analyze the results of HAR with FMCW Radar. To get insights into human activity, radar data is represented using range-Doppler maps, spectrograms, or time-frequency representations. The results of activity recognition are interpreted, including the recognized activities and their labels. By comparing the HAR system's accuracy and reliability to ground truth labels, taking false positives or false negatives into account, and assessing performance measures.

Although FMCW Radar is a promising technology for HAR, there are several drawbacks and difficulties to take into account. It might have trouble differentiating apart fine-grained activities with minute variations. The accuracy of HAR employing FMCW Radar can be impacted by environmental factors such as noise, clutter, and multipath propagation. When using FMCW Radar for HAR, it's essential to protect user privacy by following the right procedures. Because of its ruggedness, multi-person tracking abilities, and lack of touch, FMCW Radar has a lot of potential for detecting human activity. The FMCW Radar is able to precisely detect and identify a variety of human behaviors by utilizing radar signal processing techniques, feature extraction, and machine learning algorithms. The construction of effective and efficient systems for activity monitoring and analysis is made possible by understanding the foundations of FMCW Radar for HAR.

A detailed dive into the concept, with in-depth examples and a suitable explanation of Radar Human Activity recognition, is provided by the paper titled 'RadHAR: Human Activity Recognition from Point Clouds Generated through a Millimeter-wave Radar'. The paper demonstrates how even activities like boxing and jumping jacks are distinguishable using radar sensing, realizing the concept of our project and making it a certain possibility. A flow similar to them is employed for our prototype implementation, even for the LSTMs used for our neural network, making the paper a strong background study for this project.

## CHAPTER 3: RADAR AND STREAMING DEVICE

The two main pieces of hardware for our implementation consist of the radar module used, AWR1843BOOST, and its corresponding data streaming device, DCA1000EVM. We chose AWR1843BOOST manufactured by TI as our radar transceiver which generates electromagnetic signals to detect the driver. It has 3 transmitters and 4 receivers so we can record Range, Velocity, and Angle at the same time, and we chose DCA1000EVM manufactured by TI as our Data Streaming Device since the data obtained by our Radar Transceiver need to be captured and streamed to our software end for further processing progress.

The AWR1843BOOST is an evaluation module that incorporates the xWR1843 mmWave sensing device, which operates in the 76-81 GHz frequency range. It offers advanced sensing capabilities, making it suitable for a wide range of applications, including automotive radar systems, industrial automation, robotics, and surveillance. Some of its key characteristics are:

- The AWR1843BOOST uses mmWave technology to enable precise and high-resolution measurements of range and velocity.
- It allows the functioning of multi-chirp frequency modulated continuous wave (FMCW) radar, enabling improved target identification and tracking.
- The radar data is handled effectively thanks to the onboard Digital Signal Processor's (DSP) real-time data processing capabilities.
- The evaluation module has integrated antennae that are tuned for mmWave sensing, making setup and operation simple.
- It has a variety of connectivity options, such as USB, UART, and GPIO, allowing for easy interaction with external systems and devices. The AWR1843BOOST has the ability to be expanded by connecting to additional evaluation modules or unique hardware thanks to its high-speed connector.

As for the hardware elements present within the module, it includes:

- The assessment module's brain, the xWR1843 mmWave Sensing Device offers cutting-edge radar capabilities for precise sensing and detection.
- To ensure optimal performance, the module has an integrated antenna array for sending and receiving mmWave signals.
- A microcontroller unit (MCU) is a part of the system that performs a variety of control and communication functions.
- The onboard power management circuitry makes sure that all of the module's components receive effective power delivery. The AWR1843BOOST has a variety of connections, including USB, UART, and GPIO headers, for attaching to external devices and peripherals.

Further in-detail report for each component and its functionality can be found in its official user guide by Texas Instruments for AWR1843BOOST.

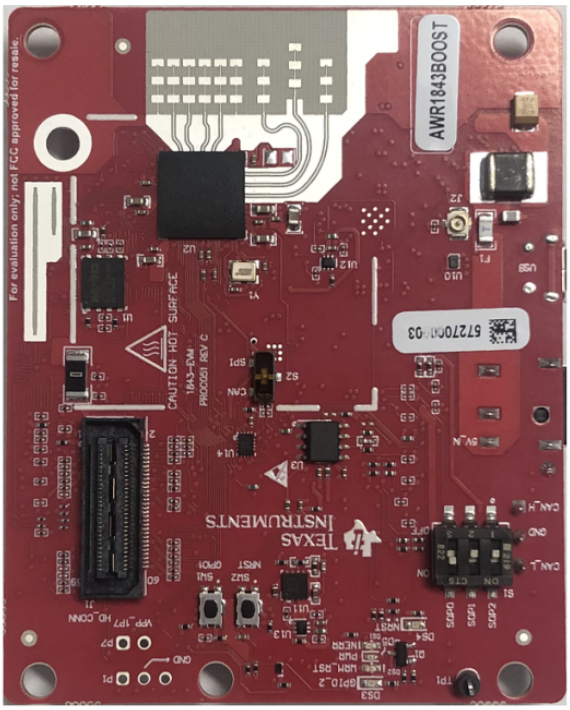



Figure 1. EVM (Front)

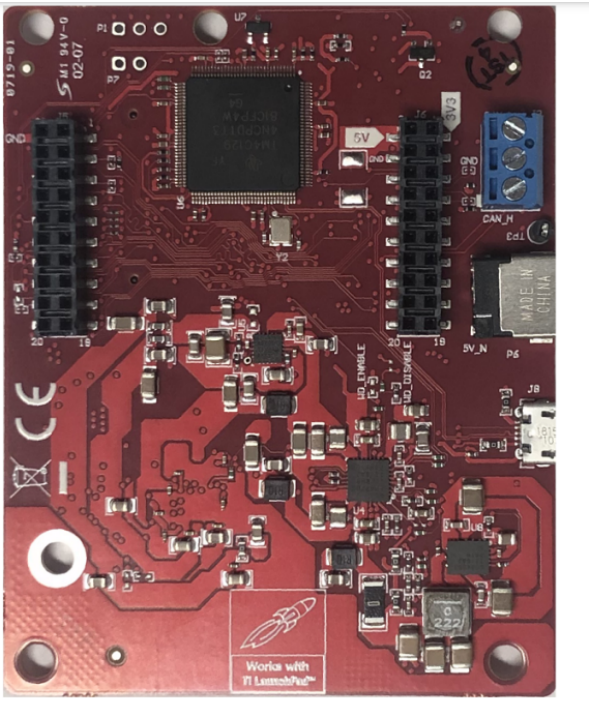



Figure 2. EVM (Rear)

Fig. Images of the AWR1843BOOST radar module by itself.

With the radar module, the DCA1000EVM Data Capture card was used as the streaming device for the radar. For mmWave radar systems, the DCA1000EVM Data Capture Card is an essential link in the data processing and acquisition chain. It makes it possible to capture raw radar data in real time, giving programmers insightful information for subsequent analysis and algorithm development. Some features are as follows:

- The DCA1000EVM provides high-speed data acquisition up to 6.25 Gbps, guaranteeing the recording of real-time radar data with the least amount of lag.
- With up to four input channels available, it allows for the simultaneous capture of data from numerous radar sensors or antenna arrays.
- The card supports both internal and external clocking settings, giving you the freedom to synchronize data capture with other system elements.
- The DCA1000EVM's enormous onboard memory allows it to retain a sizable amount of unprocessed radar data for later processing and analysis.

- Real-time data streaming enables the instant viewing, analysis, and development of algorithms from the collected data on a host computer.

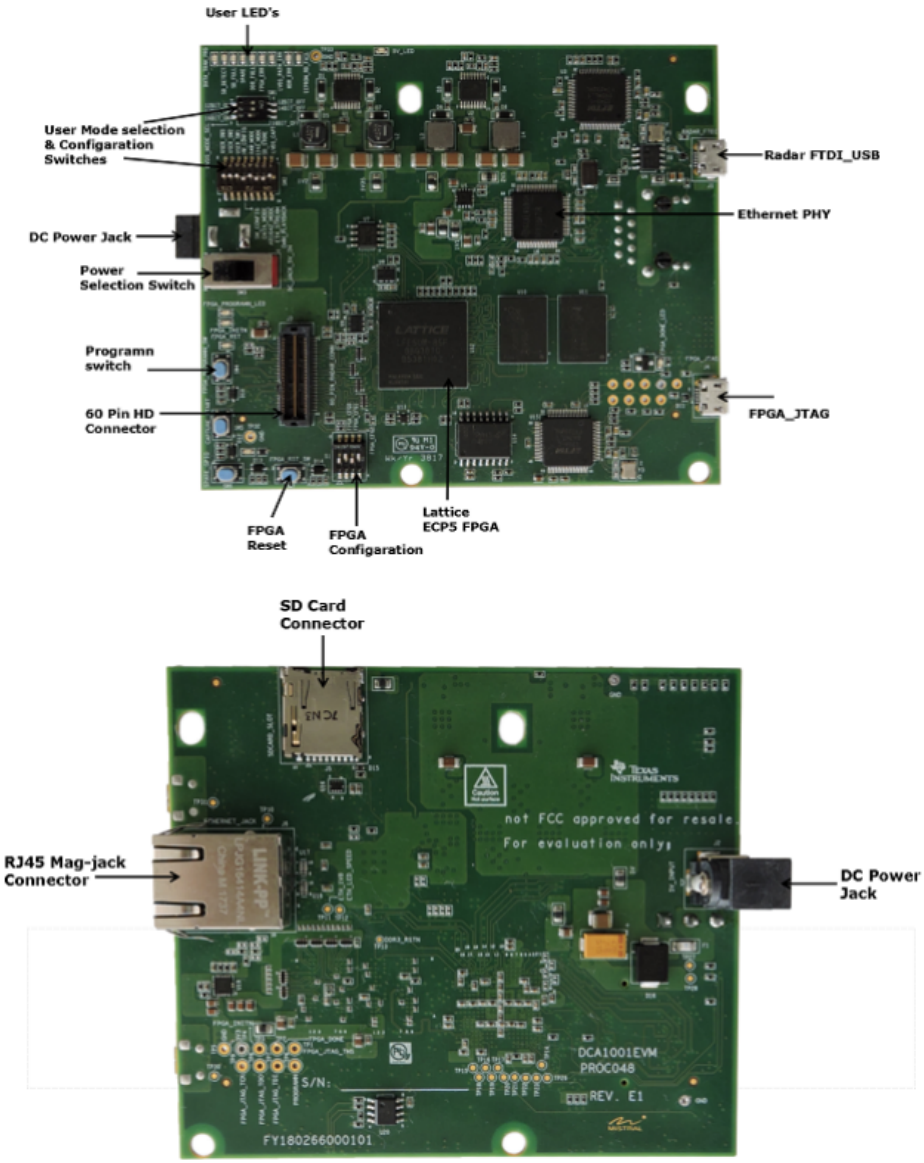



Fig. Images showing DCA1000EVM capture card by itself.

Using [the official user guide for DCA1000], a brief listing of its hardware components for some extra context includes:

- The DCA1000EVM's main component, the data capture board, is in charge of capturing and buffering unprocessed radar data.
- To transform analog radar signals into digital data, the card incorporates high-speed analog-to-digital converters (ADCs) and data converters.

- Real-time data processing, synchronization, and control activities are handled by a field-programmable gate array (FPGA).
- Before transferring the collected radar data to the host computer, a sizable buffer is provided by the onboard memory.
- The DCA1000EVM has connectors for input that can be used to attach radar sensors or antenna arrays, as well as connectors for output that can be used to interface with the host computer.

With the use of each of the two devices explained, they are set up together in the manner shown below with a proper ethernet cord connecting to a PC for using mmWave studio.

Fig. Demonstration of how the complete module should look after successfully setting it up

Upon establishing a connection between the two cards, with proper ribbon wires, the radar module requires a 5V supply using the power connector from the PC as well, along with the ethernet cable. More in-depth setup guides if required, are available in the User Guides for each respective component.

## CHAPTER 4: MM-WAVE STUDIO

Based on the original user guide by Texas Instruments themselves on mmWave studios, a brief overview would be that Millimeter-wave (mmWave) radar sensor data can be configured, calibrated, and analyzed using mmWave Studio, a potent software development environment. Texas Instruments (TI) has created mmWave Studio, which enables users to take advantage of the advanced mmWave sensor technology. This technology enables a variety of applications in automotive, industrial automation, robotics, and other industries.

The utilization of millimeter-wave frequencies in the radar system, which typically ranges from 30 GHz to 300 GHz, is referred to as "mmWave" technology. These high-frequency waves have a remarkable level of resolution, making it possible to identify, image, and sense objects with great accuracy. This technology is used by TI's mmWave sensors to provide cutting-edge solutions for a range of sensing applications, including long-range object identification, gesture recognition, robotics navigation, and industrial automation. Rapid prototyping, system evaluation, and algorithm development are made possible by TI's mmWave radar sensors thanks to the comprehensive software suite known as mmWave Studio. It offers an intuitive graphical user interface that makes configuration easier and promotes data visualization, freeing developers to concentrate on deriving valuable insights from mmWave sensor data.

Important mmWave Studio Features, that are relevant to the project in hand include:

- Sensor Configuration: TI's mmWave sensors may be easily set up using mmWave Studio, which gives users access to settings for radar range, resolution, power, and data interface. This enables users to tailor the sensor's performance to meet the needs of their particular applications.
- Real-time Data Visualization: The software allows for the live display of radar data, including measures of range, speed, and angle. This makes it possible for users to quickly understand the surroundings and things that the mmWave sensor is picking up, which helps with algorithm development and system evaluation.
- Signal Processing Capabilities: mmWave Studio includes sophisticated signal processing features that let users process radar data using digital signal processing algorithms. This creates possibilities for putting into practice innovative algorithms, extracting features, and creating sensing solutions unique to certain application requirements.
- Testing and Calibration: The program offers tools for sensor calibration, enabling exact measurement and calibration of system parameters to guarantee dependable and accurate

performance. Additionally, mmWave Studio provides testing tools to assess the sensor's operation and confirm its performance in various scenarios

- Interface with External Tools: mmWave Studio allows easy integration with third-party applications like MATLAB® and Simulink®, which makes it easier to create intricate algorithms and run system simulations. Because of this interoperability, one can use their current resources and skills to work with mmWave sensors.

For installation of the application, as demonstrated by Texas Instruments, these are the key steps:

1. Install mmWaveStudio from the installer package.
2. Install 32-bit Matlab Runtime Engine (Version 8.5.1): It is used to run the PostProcessing utility within mmWaveStudio.
3. If required(FTDI drivers will be installed automatically at the end of mmwavestudio installation. Step 3 is only required if the automatic FTDI installation fails.), install FTDI Drivers: FTDI USB Driver (mmwave_studio_\mmWaveStudio\ftdi) necessary to work with Radar device is installed. See section 2.3 for FTDI driver installation
4. Install Microsoft Visual C++ 2013 Redistributable package if using a Windows 10 machine
5. After the installation is complete, the GUI executable and associated files will reside in the following directory: C:\ti\mmwave_studio_\mmWaveStudio
6. Power up the xWR1xx DevPack (or DCA1000 EVM) and the xWR1xxx BOOSTEVM. To start the GUI, click on the file called “mmWaveStudio.exe", located under the C:\ti\mmwave_studio_\mmWaveStudio\RunTime folder.

Thus, it requires the availability of MATLAB within the system of use for the initial use of the mm-wave studio. A comprehensive step-by-step guide to connecting the DCA1000 radar with the mm-wave studio is provided by Texas Instruments themselves: https://www.ti.com/video/5827389052001.
Our team made use of the same 17-minute video and is completely appropriate to set-up the radar as required for this implementation, with the only difference here being the exact configuration used by the team was different.

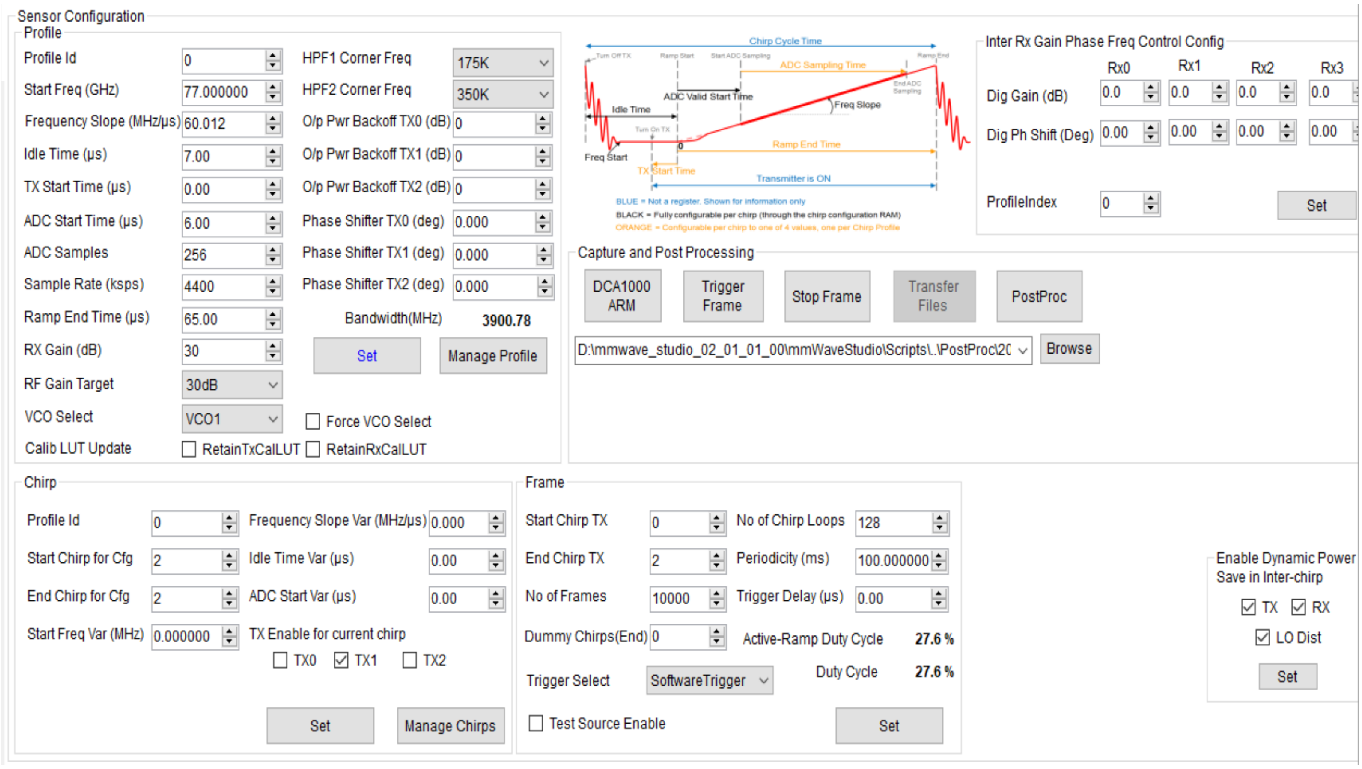



Fig. Screenshot of mmWave Studio showing the exact values used by our team.

The above figure is a screenshot documenting the exact configuration used by our team for our implementation when recording data. The team found recorded data in this configuration to be suitably scaled to notice movements by the driver and were accurate enough that neural networks were able to read the distinct bad behaviors. Following the video tutorial, if these exact values for sensor configuration are made use of, the FFT plots down the line would also be of a similar scale and value as achieved by our I2I-II team.

For in-depth guides or troubleshooting, the official guide of mm-wave studio GUI by Texas Instruments would serve the exact purpose.

## CHAPTER 5: DATA COLLECTION

Other than providing a interactive GUI, mm-wave studio also has the very useful features of a Lua Shell and automation scripting capabilities that enhance the flexibility and automation of tasks within the software. These features enable users to streamline their workflows, automate repetitive tasks, and customize the behavior of mmWave Studio to suit their specific requirements, and is exactly what our team used to automize capturing data using the radar sensor module.

An interactive command-line interface is provided by the Lua Shell in mmWave Studio for running Lua scripts and carrying out ad hoc tasks. The lightweight, potent scripting language Lua is renowned for its clarity and adaptability. Users can interact with mmWave Studio interactively, run commands, and access a variety of programmatic functions using the Lua Shell. The Lua Shell offers an easy approach to automate processes, experiment with sensor setups, and gain access to sophisticated capabilities by enabling dynamic scripting and runtime manipulation of mmWave Studio's internal state. Without the need for a separate programming environment, users may quickly prototype and test ideas by executing Lua code fragments directly in the shell. Lua scripts are supported by mmWave Studio's automation scripting feature, which enables users to design their own automation processes and automate difficult tasks. Automation scripting increases productivity by removing the need for monotonous manual activities and enabling bulk processing of several configurations or datasets.
mmWave Studio's features, including sensor configuration, data acquisition, signal processing, and data analysis, can be managed programmatically by users via automation scripting. The users can use mmWave Studio's internal functions and objects by accessing and manipulating them through the power of Lua scripting, which enables the development of sophisticated automation routines.

The advantages of automation scripting in mmWave Studio include:

- Users can write scripts to automatically configure mmWave sensors with particular parameters, making the process of comparing multiple settings and enhancing sensor performance for diverse scenarios more straightforward.
- Batch Processing: Automation scripts allow for the rapid analysis of huge datasets by enabling the batch processing of numerous data files or settings. This is very helpful when working with sensor array data or when conducting parameter sweeps.
- Custom Data Analysis: To draw valuable conclusions from mmWave sensor data, users can incorporate custom data analysis methods and processing routines in Lua scripts. The creation of specialized analysis and visualization techniques is made possible by this flexibility.

To help customers make the most of these functionalities, Texas Instruments also provides documentation and resources that are specifically focused on automation scripting with mmWave Studio, which is available in the official mm-wave studio GUI user guide, all of which was used by our team to gather an understanding for our utilization.

The GitHub repository by HavocFiXer, provides an exact Lua script, along with other Python data capture scripts for the radar sensor our team used. To provide a workflow from setting up the radars, to installation and connecting the mm-wave studio to then automating said application, it would include the following steps.

1. Launch mmWave Studio and open the Lua Shell.
2. Create a new Lua script file or modify an existing one using a text editor or the built-in Lua editor within mmWave Studio, which in this case would be using the Lua script for the repository.
3. Configure the mmWave sensor parameters according to the parameters established in the Lua script, as having dissimilar specifications would lead to errors.
4. Set up data capture settings, using Python in the case of our implementation since our whole pipeline of processing data and neural network were also set up using Python, maintaining a single language environment.
5. Start the data capture.
6. Monitor the progress of data capture, within the console log of mmwave studio.
7. Save the acquired data to files.
8. End the data capture session, giving the user the binary data for the amount of time the sensor was capturing data.

The exact locations of where these windows are found can again be found in the official mm-wave studio user guide, providing context and ability to the user to experiment with their own configurations. But to achieve our exact implementation, using the exact steps and configuration would save a significant amount of time, as figuring out these suitable specifications took our team an extreme amount of time, much more than any of us anticipated, especially because of our strong lack of knowledge in this field. Figuring out the proper combination, using all of the receivers and transmitters available on the device, and understanding mm-wave studio, and LUA scripting, were all very crucial and time-consuming.

After completion of the software side of data capturing, we move on to the physical setup side of data capturing and its corresponding setup. During I2I-II, our team captured 3 minutes of data of each

team member at the configurations established in the last chapter, for each bad behavior we planned to classify, which were six bad behaviors namely:

- Leaning Backwards
- Leaning Forwards
- Looking Left
- Looking Right
- Looking at phone
- Third-person interference.

A set of data was recorded inside a laboratory, inside a car, and outside in an open environment for all six of the predecided bad behaviors. In each of these 3 minutes of data, each frame is essentially a learning parameter leading to 1800 frames of data that needs pre-processing for the neural network to learn from. The Lua script was modified accordingly such that 3 minutes of data was captured for each of the team members. During this is where we first noticed the phenomenon of multipath propagation and Data Loss. We will deal with multipath-propagation in the next chapter, as for data loss, ensuring a stable ethernet connection, and avoiding any loose ports is important as it could lead to frequent loss of data or make the recorded data inconceivable. Hence, extra caution would certainly save unnecessary loss of time.

Demonstrating the exact order in which data was recorded, the device was set up in the following manner for car and outdoor environments.

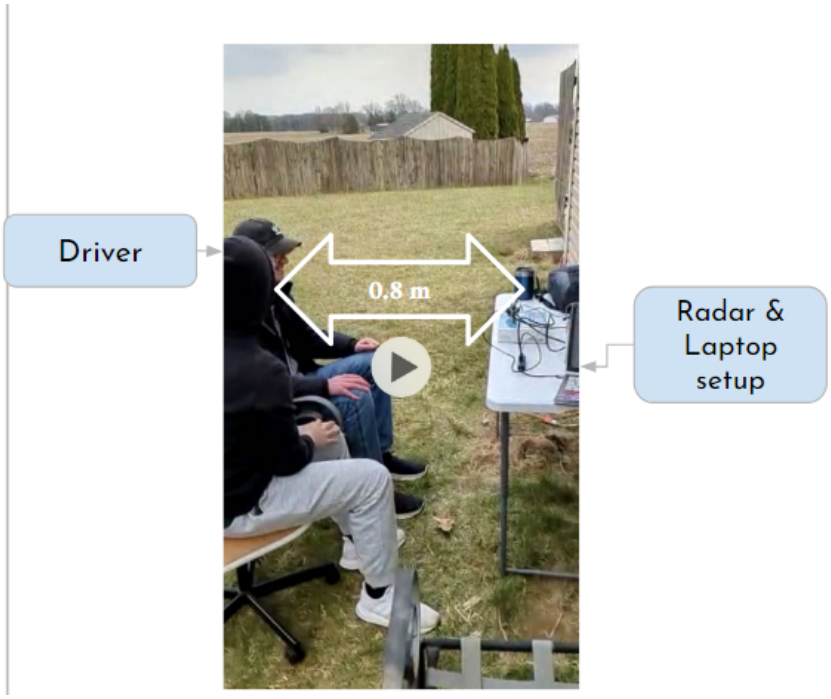


Fig. Outdoor setup when recording third-person interference

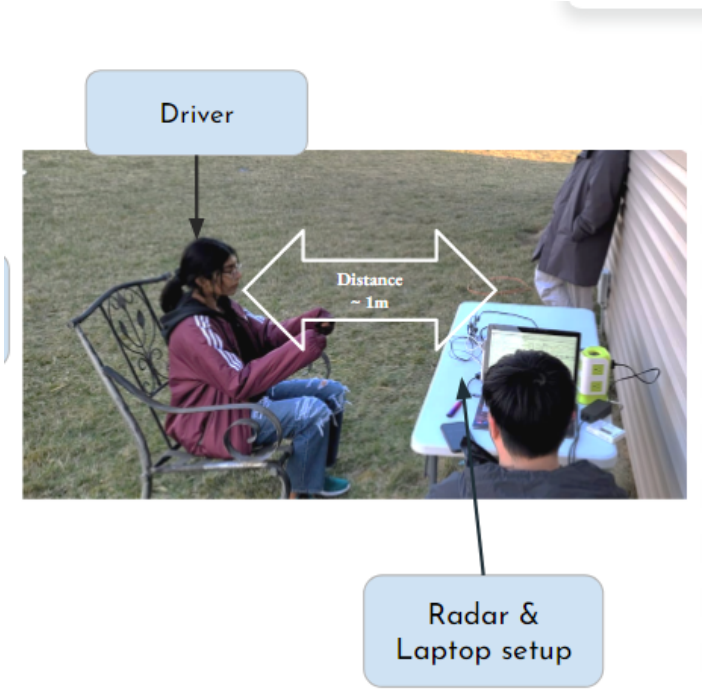



Fig. Outdoor setup when recording other bad behaviors

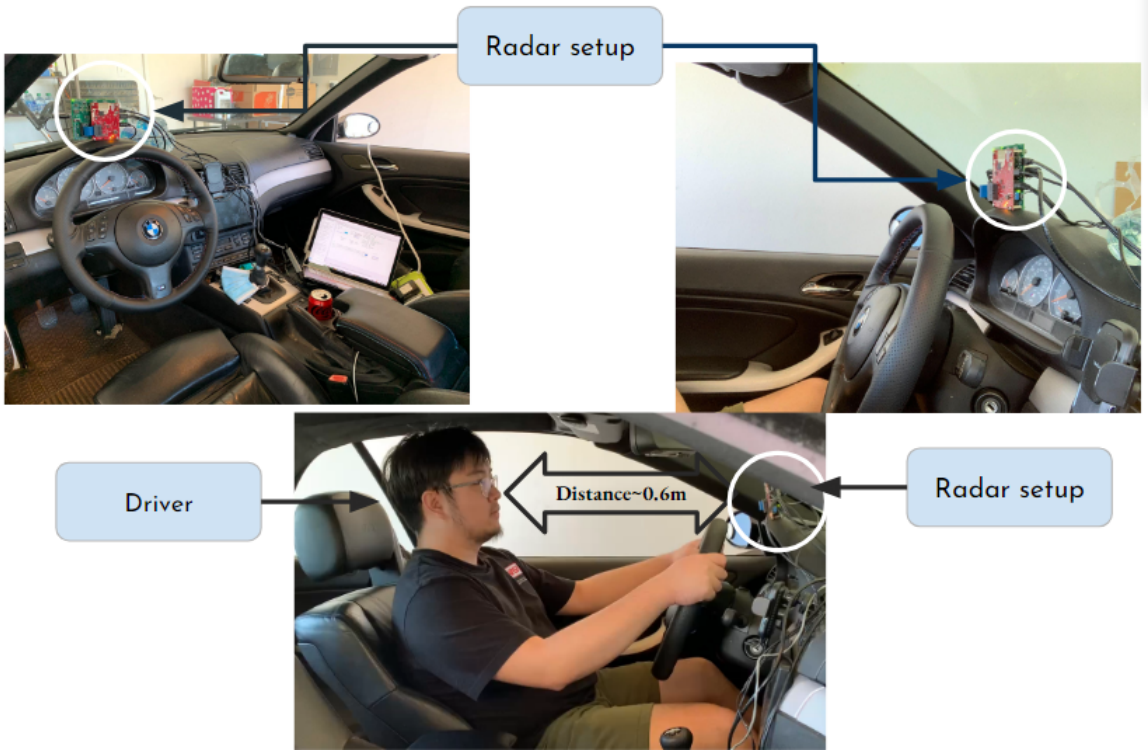



Fig. Showing setup when recording data inside a car

As demonstrated by the images of the setup above, the radar is positioned in front of the person whose data is being recorded with another person triggering the data capture flow using a laptop, and storing the binary data for each bad behavior from each person.

We started to collect the data in the EE lab building. The main purpose is to set up the parameters correctly and test the stability of the communication between the radar device and the software end. The data that we collected in the lab showed that the multipath propagation of the signals caused much noise to the data we collected. After that, we tried to collect data in the hallway of the EE building but from the data plots we collected, the situation had not improved.

After knowing the issue caused by the multipath propagation of the signals, we chose to collect our data in an open area. We moved our device to the backyard of one of our team members' residences so that the tester/driver will be the only object that the radar may detect within 15 meters. The driver was positioned 0.8m ~ 1m away from the radar device and faced directly to the radar

And we customized the parameters on mm-Wave Studio to gain maximum radar resolution based on the limitation of the radar device as below:

- 2 LVDS lanes for receiving reflected signals
- Sampling rate - 4400 ksps
- ADC Samples per chirp - 256
- Slope - 60.012
- Number of chirps per frame - 128
- Number of frames - 1800
- Periodicity - 100 ms
- Format of the received signals - complex

In this way, the noise caused by multipath propagation of signals decreased significantly. By collecting the data in this environment, we collected 6 actions for each of the 5 individuals from our team and stored them separately labeled the raw data files by different classes for further data processing work.

However, we thought that it was necessary to collect some data under a more realistic or practical environment since ideally our prototype should be installed and applied in a vehicle, a set of data was collected inside of a stopped vehicle as well. We placed the radar device above the dashboard of the vehicle and collected 5 actions from 1 individual on the team. With the help of the automation modification, we did not need to set up the parameters one by one every time we reboot the mm-Wave Studio IDE, everything can be done just by one click of running the .lua script we modified.

## CHAPTER 6: DATA PROCESSING AND DENOISING

The final chapter talks about the nitty-gritty of how the raw ADC binary files can be used for successful human bad-driving behavior detection. For our specific implementation, after we have collected all the data we need, we organized the data into folders as figures shown below, following this format:

Folder name: [action integer label]_[individual Initial]_[action string name]

File contains: time-doppler : one time_fre.npy file

Range-doppler & Angle: 1800 .npy files

For labeling the actions/bad driving behaviors, we used numbers following the mapping rules:

0 - Head Left(HL)

1 - Head Right(HR)

2 - Leaning Forward(LF)

3 - Leaning Backward(LB)

4 - Phone Usage(PU)

5 - Interference(PI)

After this, we started to pre-process the raw data we collected. Because our raw ADC data was stored in a frame tensor, it has a chirp axis and ADC sample axis. We applied FFT along the sample axis of our frame tensor to obtain the range-time tensor for further transformation. To calculate the range-doppler plot, we applied FFT along the chirp axis of the range-time tensor. To calculate the angle plot, we measure the phase difference using 2 Rx. Short-time Fourier transforms (STFT) is performed on the signal in the range of focus to get the time-Doppler spectrogram. The range of focus here is obtained by finding the maximum range for every chirp axis.

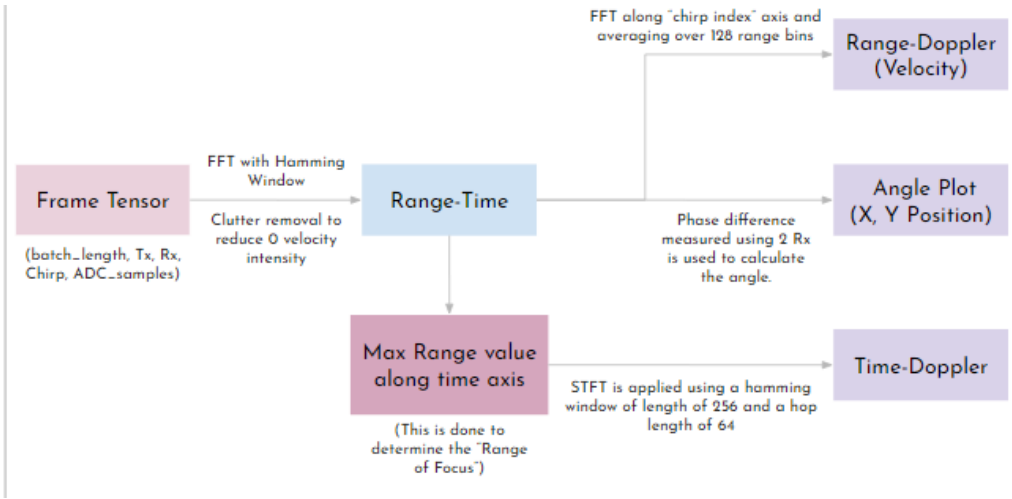
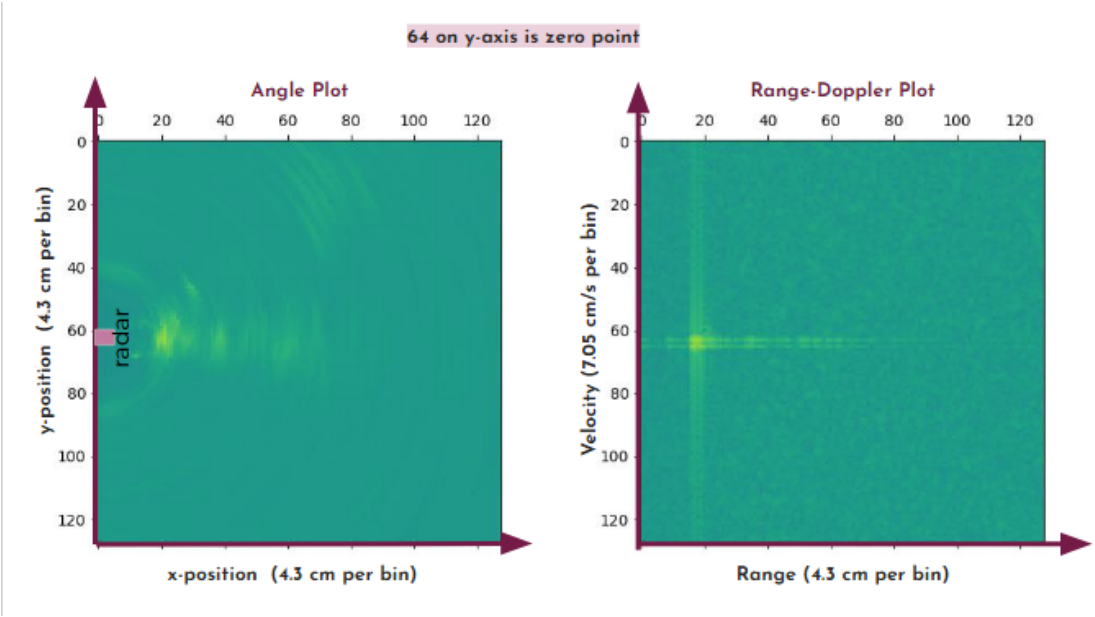
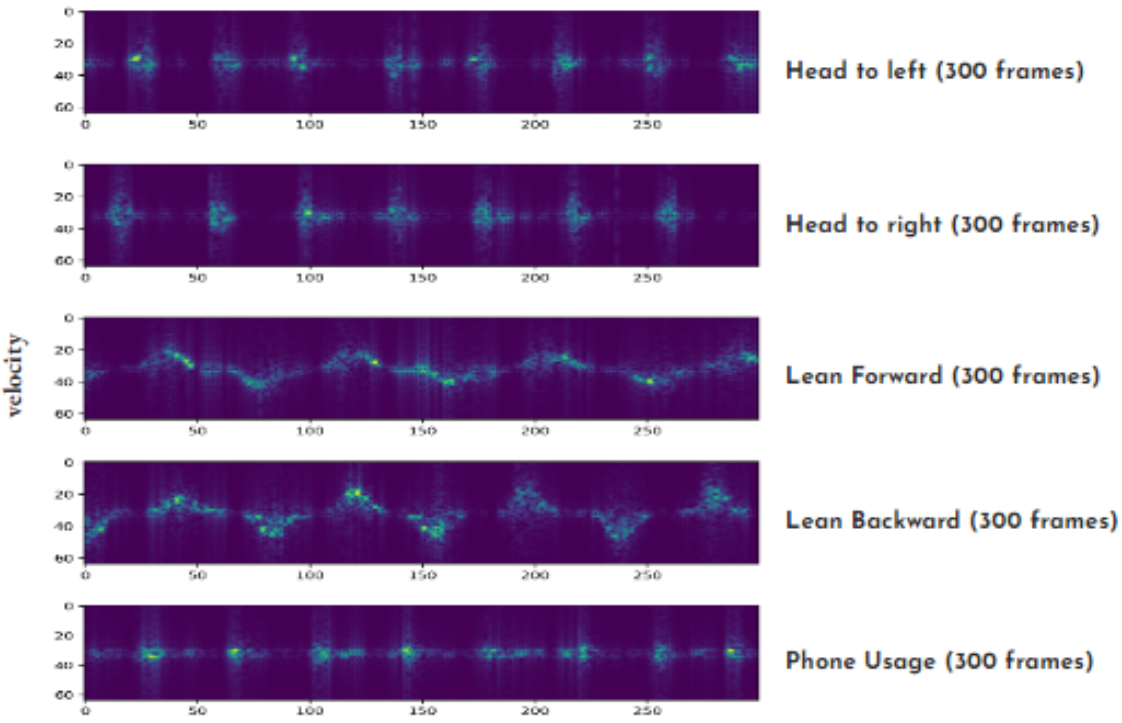



Fig. Flowchart describing the step-by-step approach used by our team for preprocessing captured data




The above figure shows the manner in which the FFT plots, from the ADC Data should be visible in, with two separate plots for Angle Plot and Range-Doppler Plot. Similarly, Time doppler plots should also be realized, with the provided code, as shown below.



This was achieved, using codes provided by our Technical Advisor/Teaching Assistant Qiming, we based our own codes to implement data preprocessing, denoising, and pipelining it all into the neural network. For this part of the project, the main steps include:

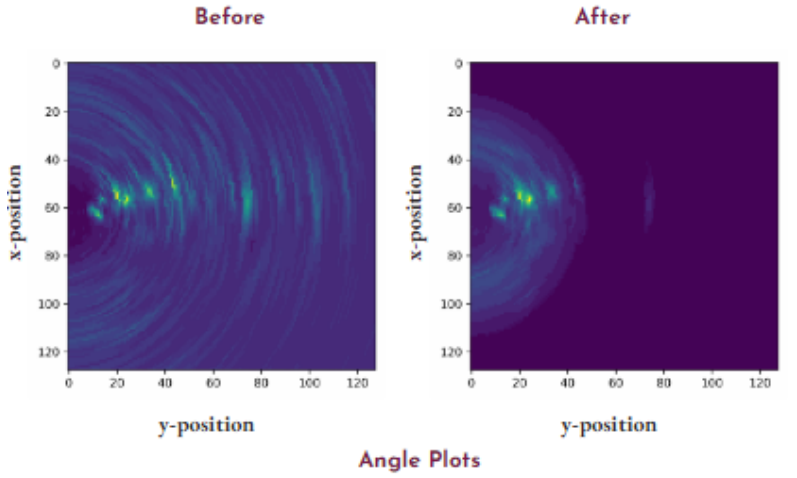
- Perform signal processing techniques to transform into usable data
- Data will be transformed into an angle map and a range-velocity map
- Denoising techniques will be used on the processed usable data.
- A fusion technique will be performed to prepare data as input for Neural Network

Step-wise code is provided for the same here: Jupyter Notebook

These codes essentially explain that we:

- First we are going to see the main approaches for our system.
- For our prototype design, first we need to configure doppler radar in order to capture the motions at a desired range and with desired sensitivity.
- We Convert the binary data obtained from data collection to .npy files and pre-process them to obtain the 3 plots.
- Then we apply denoising to all plots, which will be demonstrated later. Then the plots after denoising will be the train and test data for our Neural Network
- For labeling and creating the dataset, we decide to record 6 different actions for each individual of our team so there will be 5 people with 6 actions each.

With the denoising steps here, we deal with the multipath propagation mentioned earlier. The code provided for denoising takes the processed FFT plots from the raw binary data and puts it through a Savitzky-Golay filter for time-domain denoising and processes the FFT plots through Blackmann Harris Window for denoising within the frequency domain. Results for the same should appear in the manner shown below:



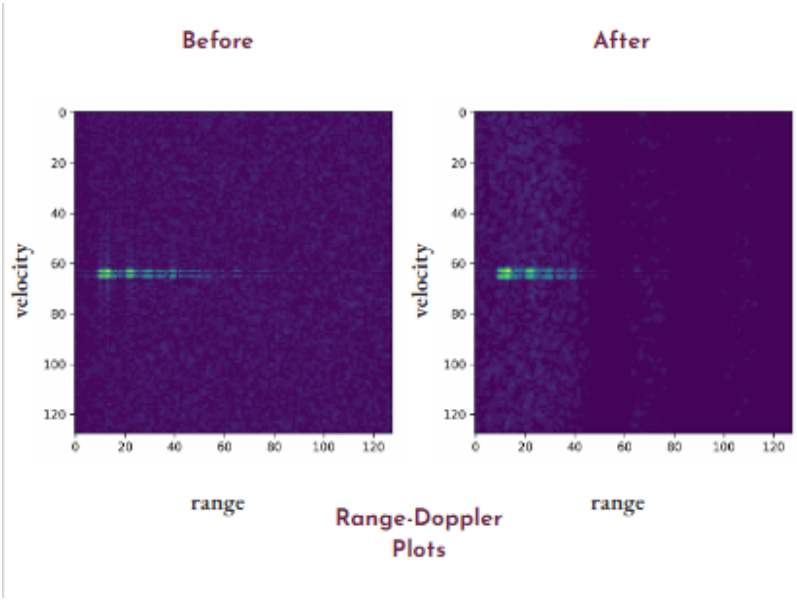




Fig. Shows the results on FFT plots after denoising to get rid of Multipath-Propagation

With appropriate preprocessing done and recorded, the next step would be essentially pipelining these plots through neural networks to achieve the proposed bad behavior detection. A version of this was attempted and is present within the Python notebooks, with proper documentation to attempt the same. But for this user guide, as it was aimed to let a new user understand the concepts of mmWave sensing, setting up the radar, and being able to record and process data from the radar module, without having to waste extra time understanding each and every minute detail of each moving component of

the implementation, with necessary cautions for where the team got stuck for our implementation, the guide should bring the reader to speed with our progress with ease and efficiency.

**REFERENCES/DOCUMENTATION:**

- Link to all of the Reference Codes provided by Qiming, for reading, and processing time-doppler data from the binary files, on which we developed our own codes: https://3.basecamp.com/4975970/buckets/33228437/vaults/6341167742

- Link to User-Guides for the Hardware modules and mmWave Studio by Texas Instruments: https://3.basecamp.com/4975970/buckets/33228437/vaults/6341165812

- Link to all extra reading materials indulged by our team to develop a deeper understanding or to flush our doubts around certain concepts: https://3.basecamp.com/4975970/buckets/33228437/vaults/6341166676

- Link to all code, including denoising, the different preprocessing steps for different FFT plots, pipelining the processed data as well as our attempt of neural networks for Human Activity Recognition: https://3.basecamp.com/4975970/buckets/33228437/vaults/6341168870

- Video Tutorial for Data Capture Setup, using the radar module via the mmWave Studio by Texas Instruments: https://www.ti.com/video/5827389052001

- GitHub Repository used by the team, for automation of Data Capture, bypassing the mmWave studio making using of Python Script and the Lua window within mmWave Studio https://github.com/HavocFiXer/mmMesh/blob/master/1.mmWave_data_capture/DataCaptureDemo_1843new.lua