Department of Electrical, Computer, and Software Engineering

Part IV Research Project

Final Report

Project Number: 77

Millimeter Wave Radar based Human Activity Recognition

Samuel Mason

Beck Busch

Kevin I-Kai Wang

Akshat Bisht

13/10/2023

Declaration of Originality

This report is my own unaided work and was not copied from nor written in collaboration with any other person.

Name: Samuel Mason

ABSTRACT: Human activity recognition (HAR) is an area of research that has received increasing attention in recent years. HAR can be applied in many areas, such as human computer interaction, smart homes, internet of things, detection of abnormal or suspicious behaviours, fitness trackers and much more. This research explores alternative methods of activity segmentation to allow for the detection of activity ends. This is in an effort to improve the recognition of complex activity sequences, which consist of multiple activities in succession, and may constitute a higher level behaviour. Existing research fails to cover the recognition of longer, more complex activity sequences and assumes activities fit into a window of a fixed length. We propose a novel activity recognition method using millimetre wave (mmWave) radar, one that selects salient activity data frames, and allows for the recognition of activity sequences using machine learning.

Acknowledgements: I would like to thank my supervisor, Kevin Wang for his guidance in the project, as well as my co-supervisor Akshat Bisht. Additional thanks to my project partner Beck Busch for his efforts over the course of this project.

Table of Contents

Tables and Figures

1. Introduction

Broadly speaking, human activity can be grouped into the categories of full-body motion, as explored in [1-13], typically involving the movement of multiple limbs, and gestural activity [13, 14-19], which is on a smaller scale and involves arm or hand motions. Posture estimation [20, 21] can also be used to discern human activity by positioning of the joints. For this literature review, involuntary activities are not being considered. For example, monitoring of heart and breathing rate as part of vital signs detection. For our research we only consider line of sight (LOS) conditions.

2. Literature Review

HAR can be realised using numerous technologies, typically consisting of sensing component(s), data processing and pre-processing modules (including activity segmentation), and classification component(s). Classification components are typically realised using machine learning algorithms in existing approaches. Machine learning approaches typically differ based on the specific implementation of HAR systems and can be tailored based on the sensing method. It is for this reason that determining the sensing method of choice is of importance. Some of the different sensing technologies commonly used for HAR are cameras [10], WiFi channel state information (CSI) [6, 7, 9], Lidar [8], sensors that attach to the body, such as accelerometers or gyroscopes [3, 11] and millimetre wave (mmWave) [1, 2, 4, 5, 13-21].

Camera footage was used in [10] for segmenting activity repetitions. However, there has been a shift away from using cameras in HAR applications due to privacy concerns. The advent of newer, commercially available sensing technologies has made HAR possible without the need for cameras. Camera based methods suffer in performance when lighting conditions are poor, and cannot sense through obstacles, making them unsuitable for a general use case.

Sensors that attach to the body, such as in [3, 11] have seen applications in HAR. However, these types of sensors can be seen as inconvenient and potentially uncomfortable [7], as HAR is not possible unless the user carries the sensors around. There have been efforts to reduce the number of sensors used, as in [11], where accelerometer data from a mobile phone was used to provide HAR, but if a user is not carrying their phone, then HAR is not possible.

Lidar has been explored for its potential application in HAR, albeit in a limited capacity [8]. Lidar has the advantage of not being sensitive to lighting conditions and can be used both indoors and outdoors. However, a major drawback of lidar systems is the cost, as lidar sensors are typically very expensive when compared to electromagnetic wave-based sensing methods. Lidar also requires LOS and cannot sense over longer distances.

Electromagnetic wave-based sensors include WiFi and radar, which have become popular for use in HAR in recent years. WiFi based recognition systems typically employ channel state information (CSI) data to detect human influences on the environment. However, this sensing method is not very precise, as WiFi usually operates between the 2.4GHz and 5GHz bands, resulting in a wavelength in the order of centimetres.

Radar systems, specifically radar within the millimetre wave (mmWave) range are the sensing method of choice for a lot of very recent research. Unlike WiFi CSI, as the name implies, wavelengths are in the order of millimetres. This allows for more fine-grained recognition of HAR tasks. The use of mmWave radar in HAR is a recent development, which has been made possible by the advent of single chip radar.

Both WiFi and radar-based systems are insensitive to lighting conditions, making them preferable over cameras for HAR. Unlike wearable sensor-based recognition systems, only one or two (as in [4]) radars are required for the task of HAR. Radar does not need to be attached to the body, but in applications where system mobility is of importance, radar has been shown to be able to perform HAR tasks. [14] references a radar system embedded within a smartphone, showing that radar-based HAR systems have the ability to be portable.

2.1. Segmentation methods

Many of the proposed solutions for HAR in existing research have fixed assumptions about the length of activities being performed. In [14], gesture activities are assumed to fit within 64 frames, amounting to 2.6s. Whereas in [7], researchers limited activity data collection by asking volunteers to perform activities within a specific timeframe. Activity samples are 10s long, with activities being performed between the 4th and 7th second. Volunteers are asked to stay still the rest of the time to prevent overlapping of activity data. [16] made use of temporal sampling to align gesture durations, each of which contained 30 frames (1.5s) of heatmaps. [4] performed an analysis on different sample lengths, and found that for their application, 40 frames (3.7s) provided the optimal classification result. Padding is used where necessary, to bring sample length up to 40 frames, by padding with the last frame of an activity.

There are examples of research where activity segmentation is not mentioned at all. [20] and [21] feed single frames into a CNN network, because temporal dependency analysis is not required for posture estimation. [5], which explores full-body activity recognition, decides to buffer data frames for recognition, but makes no mention of specific segmentation methods.

More commonly, however, a sliding window approach is taken for segmentation of activity data. [1] employs a sliding window of fixed length, at 1.2s, and a sliding factor of 0.3s. 12 frames are used in total to make up a feature of dimension 12 x 50 x 50 x 30, which is then fed into a machine learning model for classification. A sliding window of size (64, 3) is used in [11], which amounts to 3.2s at the chosen 20Hz. An overlap of 50% was used here. [3] involved multiple sliding windows, starting with a 6s window. Data rebalance, cluster analysis and activity segmentation are followed by a sliding window of (2, 3, 4)s with 50% overlap to segment long period activities.

A more complex approach to segmentation is made in [15], where gesture samples, performed in a window of 125 frames (~5s) are segmented with an activity detection module (ADM). The ADM consists of a binary classifier and accumulator, which is trained to detect gesture ends. However, gestures are considered to not exceed 50 frames (~2s). [9] uses a similar approach to detection and introduces an AACA algorithm to segment periods of activity from periods of inactivity. A sliding window related to the WiFi packet transmission rate is used (50 frames in this work) and has a step length of 1. A large CSI variance is taken to be indicative of periods of activity.

2.2. Project Scope

From the above discussion, it can be seen that there is a gap in existing research with regard to the recognition of activity sequences in more complex scenarios. Recognising activities of different lengths and in different contexts is of interest, especially complex activity sequences, which are sequences consisting of multiple activities, and may constitute higher level behaviours. The recognition of complex activity sequences is not currently possible, or at the very least, has not been explored in existing literature, to the best of our knowledge.

Applications of smooth HAR recognition systems include human computer interaction, monitoring and surveillance systems, fitness trackers, smart homes, and lab safety. The improved ability to recognise complex activity sequences allows for the seamless operation of these systems, where complex sequences can arise.

3. Design Methodology

After having identified a gap in the existing literature, consideration of the components needed for the intended application was made. This application requires that a recognition method be adopted, and in addition, the problem of dealing with complex activity sequences must be faced. These are explored further in the below sections of the report.

3.1. Exploration of System Architecture

In examining the options for potential system architectures, a number of classification algorithms were considered. These included CNN, CNN-LSTM, GMM, CGAN, and TCN. Potential types of data format that can be used in this application are the raw radar data cube, DFT over one or more dimensions of the data cube (range-doppler, range-angle image), an end-to-end model or the use of a meta-heuristic for activity segmentation.

3.2. System Architecture Design

Prior to designing the system architecture, it was necessary to relate a detailed problem description. It was noted that in existing literature dealing with transitions between activities, these activities are explicitly treated as their own activity classes. For example, activities such as walking to sitting, or sitting to standing. However, the inclusion of activity transitions does not scale well in a general case. For example, if a system is able to recognise 10 continuous activities, then in order to train on every possible activity transition as well, an additional 45 classes are needed. It is for this reason that a different approach was taken to this problem.

The system requirements identified were that a detection mechanism is necessary to identify activity transitions. Additional processing logic is also needed to match complex activity sequences. The devised detection mechanism is to identify salient points in the frame data that correspond to the transitional motion. Once these points have been detected, individual activity "fragments" can be identified, and, where applicable, complex activities can be inferred. Fragments of activities can be pieced together to identify when higher level behaviours are being observed by the radar. Thus, recognition of these salient points is a necessary step.

The proposed method of detecting activity transitions is to segment the activity data stream into multiple passes, with each pass containing slightly more frames than the last. From these frames, a constant number of frames will be selected such that the input to the classification algorithm(s) is uniform in size, even for passes of different sizes. When the classifier can no longer identify a pass as an activity with high confidence, then this is indicative that there is some unrecognised activity occurring, which can be assumed to be transitional motion.

4. Radar Model

Following the analysis of different options for the system architecture, an investigation into the different commercially available radar models was conducted. The chosen radar model for this application is the IWR1843 [22], made available by Texas Instruments. This was included in the development board IWR1843BOOST [23], which provides functionality for interfacing with the radar chip, and to other boards provided by Texas Instruments. The evaluation module DCA1000EVM [24] was additionally chosen for its ability to capture and stream raw ADC data as it is captured from the radar. The added streaming capability provided by this board allows for a higher temporal resolution than the BOOST can provide on its own.

The relevant specifications for the IWR1843 are provided in the table below.

Table 1: IWR1843 Specifications

Frequency range	76 - 81 GHz
Number of receivers	4
Number of transmitters	3
ADC sampling rate (ksps)	25000
TX power (dBm)	12
DSP type	1 C67x DSP @ 600MHz
Interface type	CAN, CAN-FD, I2C, LVDS, QSPI, SPI, UART

Particularly, the virtual antenna array for this radar has size 12, which allows for a higher angular resolution than some of the other available radar options, and makes it desirable for this application, where spatial and temporal resolution are desirable.

4.1. Radar Environment Setup

The IWR1843BOOST and DCA1000EVM were both set up according to the user documentation available from and provided by Texas Instruments [23, 24]. The ethernet cable used for streaming data from the DCA1000EVM, both micro-USB cables connecting to the DCA1000 and the IWR1843BOOST as well as the power cable are all connected to a laptop running millimetre wave studio (mmWave Studio). The settings used in mmWave Studio are detailed in section 6. The two boards are interfaced using a 60-pin SAMTEC connector cable.

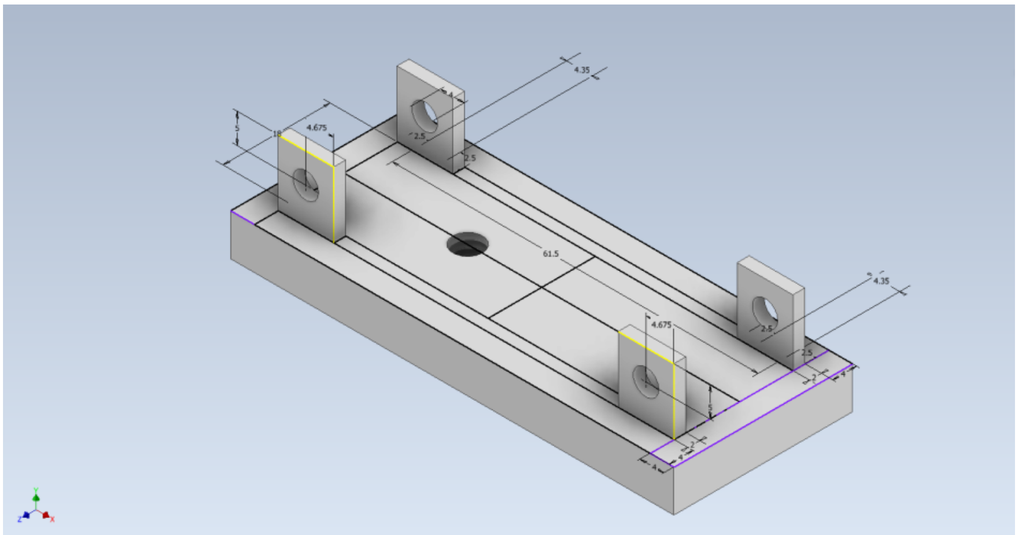


Figure 1: CAD Model of Radar Mount

The provided mounting brackets were discarded in favour of a custom mounting component, which allows for the radar boards to remain stable atop a tripod, allowing for seamless data collection in the chosen collection environment. The radar board dimensions were measured, then CAD was used to design the custom mounting component according to these dimensions. This was then 3D printed and attached to both the radar boards and to the tripod.

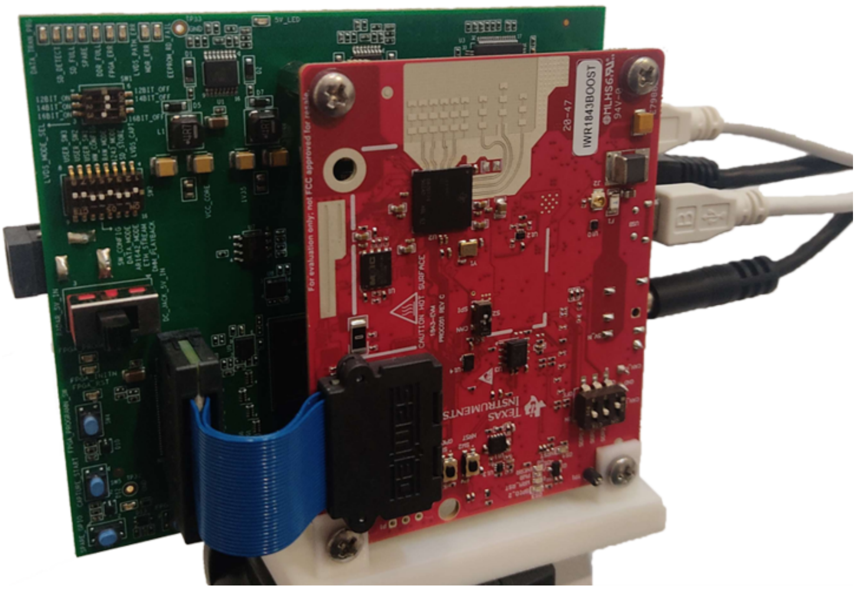


Figure 2: Radar Setup

5. Process Flow

The devised process flow for capturing human activity data using the radar and eventually classifying the result is detailed in the diagrams below. Further explanation of each component is provided in this section of the report.

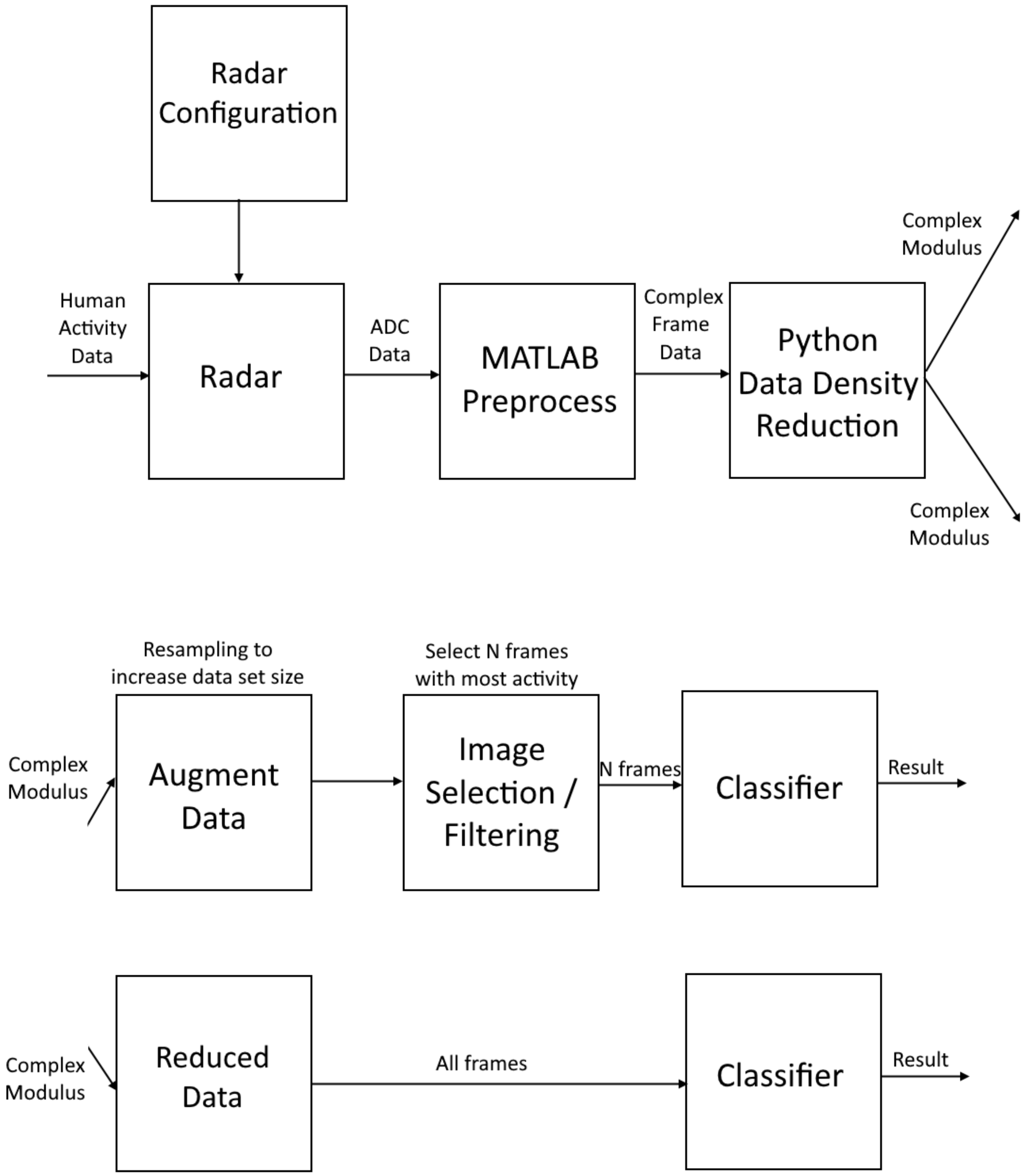


Figure 3 & 4: Process Flow Diagram

- **Radar + Radar Configuration**

  The radar configuration includes all of the settings in mmWave Studio, as well as the hardware settings for the boards.

- **MATLAB Preprocess**

  The MATLAB preprocessing step involves refactoring of the ADC data, and producing a data format that can be worked with later.

- **Python Data Density Reduction**

  Although increased spatial and temporal resolution can contribute to improved classification results, computational requirements for classification of captured data samples with the original resolution proves infeasible for this application. Hence data density reduction is performed.

- **Augment Data**

  Data augmentation can be utilised to increase the size of the training data set. This reduces the time required for data collection and can contribute to increasing the performance of the trained model(s).

- **Image Selection/Filtering**

  For the development of the novel application, frame selection is required to ensure that the input size to the classifier is consistent for samples of differing lengths.

- **Classifier**

  Classification algorithm(s) for making inferences based on frame data. Determining the activity class that an activity sample belongs to.

6. Radar Settings

The radar settings used in mmWave Studio are detailed in the table below.

Table 2: Radar Settings

Setting	**Value**
ADC Samples Per Chirp	256

Sampling Rate	10000 kS/s
Frame Count	150
Periodicity	40ms (25fps)
RX Antennas	4
ADC Data Format	16 bit, 1x Complex
Packet Delay	25µs

Radar data captured using mmWave Studio is saved as a raw data binary. The PostProc tool is used after each data capture for zero filling and packet reordering, to ensure that the saved data is consistent with measurements as they were taken.

During the development process, the interception of incoming ethernet data to allow for real-time recognition was investigated, but was not carried forward, to prevent the project scope from becoming too large.

6.1. Preliminary Data Collection

Following the formulation of the process flow and setting up of the radar, a preliminary data collection step was carried out. The purpose of this data collection was twofold. The first was to assess the suitability of the chosen radar settings for the target application. The second was to aid in the development of classification algorithm(s), to provide a (limited) data set to train algorithms on.

This data collection step involved collecting 4 second samples of activity data from both researchers. Three types of activities were recorded during this data collection, with 10 samples collected for each researcher. The activities selected were clapping, jumping and walking, totalling 60 activity samples.

7. Data Processing

7.1 Preprocessing

7.1.1. MATLAB Preprocess

The captured ADC data is first processed using MATLAB, before being sent to subsequent stages. The MATLAB script takes the raw ADC values and converts them into doppler strength on the range and angle axes. This involves creating a radar data cube for each captured frame, where the dimensions of the cube are radar chirps (slow time), antennas, and radar samples (fast time). Slices of the radar data cube are taken in order to produce the desired format.

Input data to the MATLAB script is in the form of ADC binaries, and the script outputs the complex frame data to a csv file. The csv contains one row per captured activity, where the first column contains the name of the activity according to the naming convention detailed further in section 7.1.3. The rest of the row contains the flattened frame data, which can be read in row-major order.

The data outputted to the csv does not represent the entire range of data captured by the radar. This is because the maximum radar range exceeds the vicinity of where the human activities have been recorded. As part of the MATLAB processing, the maximum range was clipped to 5m, and the maximum angle was clipped to +/- 50 degrees. This helps to reduce processing time in later steps. This results in a reduced frame size of 143x35 values, or 5005 when flattened.

7.1.2 Python Data Density Reduction

The purpose of the data density reduction step is largely to reduce the time taken to train the machine learning algorithm(s), and to make predictions on activity sequences, using the trained models. As determined by the MATLAB processing, the input size to this stage is 143x35 for each frame. These are both semiprimes with factors (11, 13) and (5, 7), so there are four possible ways frames can be reduced using integer divisors. For this application, each frame is reduced to 7x11 values. The reduction is performed as an average pooling operation.

In addition to this reduction, the complex data is discarded in favour of the complex modulus, which can be handled by conventional machine learning algorithms. Data density reduction is a necessary step in the processing pipeline, due to the sheer size of the input. Without sufficient computational resources, the time required to process undiluted radar data becomes unreasonable.

7.1.3 Naming Convention

After each activity is captured by the radar, it is saved under the directory "..\mmWaveStudio\postProc\adc_data_0.bin" by default. Therefore it is necessary to rename the captured data file to distinguish between captures. To facilitate the renaming process, a script was written to automate renaming and moving the file out of the default save directory and into another folder.

The naming convention that was adopted is outlined below. In the case where multiple activities are captured in a single recording, they are simply listed in the order that they were performed.

Naming convention:

Files should be named according to:

- The activity being performed (all lowercase)
- The participant number
- The total activity duration
- The activity number according to the activity only (starts at 1)

Some example file names could look like this:

- boxing_01_6s_1
- sitting_standing_walking_03_15s_2
- clapping_11_6s_1

7.2. Data collection

For the collection of data to be used in training the CNN-LSTM algorithms, activity samples were collected in a single indoor environment, with background objects remaining consistent between activity captures.

The activity classes that were selected for the data collection process were intended to encompass a wide range of possible scenarios. Five activities were selected, and included standing, walking, clapping, jumping jacks and waving, as well as an empty class where no human activity was recorded, to serve as a counterfactual. Detailed descriptions for each activity can be found below:

- Empty - No human present in the scene, the only activity recorded here is ambient environmental noise.
- Standing - Remain standing stationary in front of the radar.
- Walking - Walk back and forth orthogonal to the radar direction.
- Clapping - Clap both hands together slightly below shoulder level, in front of the body.
- Jumping Jacks - Starting from a neutral standing position, simultaneously jump, widening the legs and raise the arms above the head, before returning to the starting position and repeat.
- Waving - Holding the elbow slightly below shoulder height, wave the right arm back and forth from left to right.

Activity data was collected from three individual participants in front of the radar setup. The radar was set up mounted on a tripod as previously described, and participants performed activities in front of the radar at a distance of 1.4m. Collected activity samples were a uniform 6s in length, and following the collection of each sample, participants were allowed to rest before the next sample was collected. 20 samples were collected for each activity, totalling 100 samples for the five activities per participant, plus an additional 20 samples for the empty class. 20 samples of the empty class were collected per participant to ensure consistent class sizes.

The settings used in mmWave Studio are consistent with those in the Radar Settings section, with the exception of the packet delay. It was observed that the packet reorder and zero fill was occurring semi-frequently during the preliminary data collection. An updated delay of 50µs was used here. Following each data capture, the recorded binary file was saved, renamed using an external script, then moved to an external solid state drive for storage. The renaming procedure was carried out according to the designed naming scheme detailed in the previous section.

7.3. Algorithms

7.3.1 Classifier

For the purpose of evaluation of the segmentation method used, the same type of model was used for both the baseline and the novel architecture. The network used for both the baseline and novel model was CNN-LSTM. This model has seen quite a lot of use in the field of HAR, and has been shown to produce good results. Here a 2D CNN-LSTM is being used, with the 7x11 reduced frames being used as the "image" inputs to the CNN.

7.3.2 Activity Segmentation

The primary contribution of this project is the activity segmentation method. As discussed in the literature review, existing methods of segmentation assume a window of a fixed length, and are unable to adequately handle complex activity sequences that constitute higher level behaviours. The method of activity segmentation here does not assume that activities must be completed within an arbitrary time frame. Every type of activity is assumed to be of a continuous nature, and can therefore extend indefinitely. For activities that are one shot, such as snapping the fingers, the repetition of these activities allows for them to be considered as continuous.

In order for the classification network to be able to handle activities of differing lengths, a segmentation method is required to normalise the size of the input. The segmentation method adopted here selects N frames with the highest variance / overall activity. The overall activity is calculated as the mean average of each frame's values. A maximum of one frame from each T/N frames can be selected, where T is the total number of frames for the activity. Following this, N-1 in-between frames are selected. An in-between frame is a frame that lies directly between (in the middle of) two consecutive frames that have been selected in the previous step. The purpose of this is to retain the key temporal dependencies between frames, as in-between frames do not necessarily contain a little or a lot of activity.

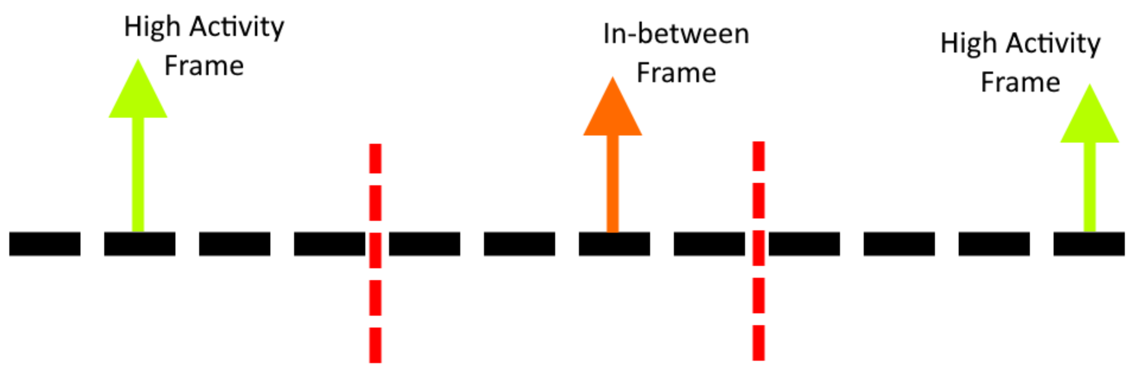


Figure 5: Example of selecting 3 frames from a 12 frame long sequence.

This method of activity segmentation draws from dynamic time warping (DTW) and principal component analysis (PCA). It aims to provide only the necessary information to the classification algorithms, so that activities can be correctly identified. It also allows for activities of different lengths to be classified by the same network. This allows for the end of an activity to be identified, as multiple passes of an activity can be sent to the classifier for classification. Each pass has the same starting point, with the duration gradually increasing to allow more activity frames to be included.

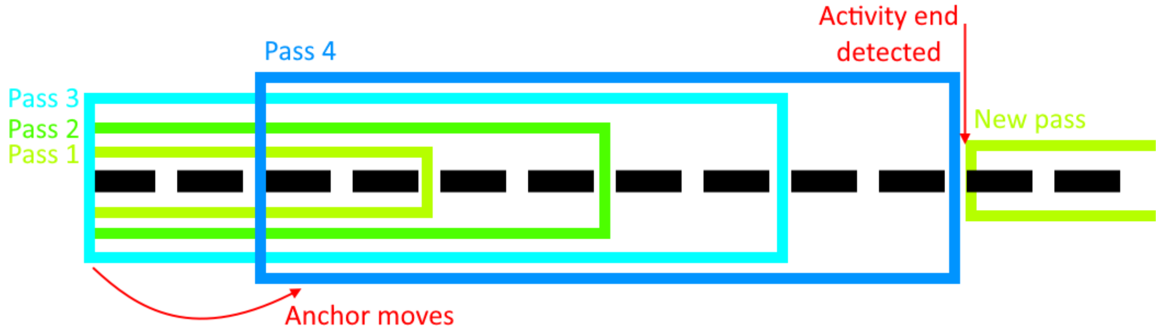


Figure 6: An example of multiple passes, where the maximum number of frames is 8.

Once the classifier can confidently match the selected frames to an activity, the respective pass can be marked as the identified activity. The longest recognised pass therefore marks the end of the activity. Each pass is relatively inexpensive, as a reduced number of frames are sent to the network for classification. If the longest pass exceeds a specified threshold, then the anchor indicating the start of an activity is moved, to prevent deadlocks in cases where recognition fails.

7.4. Data Augmentation

Numerous data augmentation techniques were used in the processing of the collected activity data. For the data being fed into both classification algorithms, clipping of the radar return was performed as described in 7.1.1. Following this, reformatting and reduction of the data was performed, as in 7.1.2. Once the data was in a more workable format, clipping of the collected samples was carried out for the CNN-LSTM+S algorithm, whereas the CNN-LSTM algorithm used the full 6s samples.

Resampling of the collected activity data was performed for the CNN-LSTM+S algorithm, whereby samples of 2, 3, 4, 5 and 6s were produced from the original 6s samples. For each 6s sample, the number of samples produced is shown in the table below.

Table 3: Augmented Data Sample Lengths

Produced sample length (s)	**Number of samples produced**
2	5
3	4

4	3
5	2
6	1

The resampling process starts by reading in all the activity samples from the reduced_data csv, then selecting an activity sample from the list of samples to be processed. The sample is then cut into smaller samples and these are then written into an augmented_data csv. The clipping process begins by selecting the first integer N seconds of the original sample for the first sample of the desired length to be produced. Subsequent samples are taken beginning from an integer representing the index of the next samples, plus some uniformly distributed random number between zero and one second. An example of taking 5 samples of 2s in length from the original 6s sample can be seen in the image below.

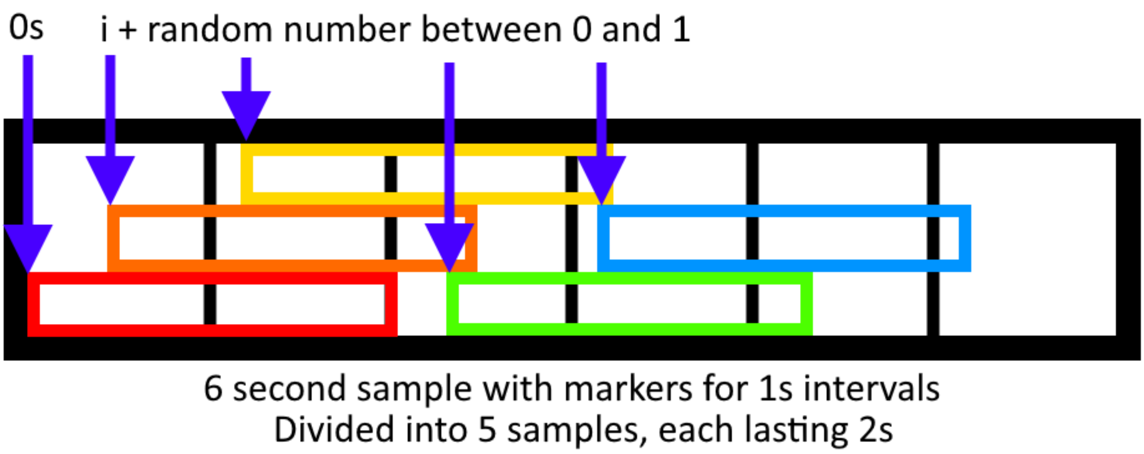


Figure 7: Data Augmentation Example

The above process is repeated for each target duration until all 15 samples have been produced from the original sample. After this process has been completed for every activity sample in the list of samples, the number of samples that can be fed to the CNN-LSTM+S algorithm is 15x the original data set. This is because although the samples are of different lengths, the number of frames selected by the CNN-LSTM+S algorithm is always constant.

The only requirement is that the minimum sample length produced in the augmentation step is greater than two times the number of frames selected in the segmentation step of the CNN-LSTM+S algorithm. This is to ensure that the downsampled activity is able to adequately represent the original activity (Nyquist-Shannon sampling theorem [25]). The size of the data set produced in the augmentation step is thus 20 samples * 5 (+1) activities * 3 participants * 15 = 5400 samples. The original data set is 360 samples by contrast.

8. Discussion and Results

The CNN-LSTM and CNN-LSTM+S models were trained on a limited data set containing 278 samples of activity data. The models both used the reduced data format, with a training:validation:testing ratio of 80%:10%:10%. The CNN uses a kernel size of 3x3, with a stride of 2 and the LSTM has a single hidden layer with 8 nodes. The Adam optimiser was used with a learning rate of 0.001. 16 frames were selected in the CNN-LSTM+S model.

Both models were trained until they started to overfit, for the CNN-LSTM model, this happened at 260 epochs, where a training loss of 0.109, a validation loss of 0.213, and an accuracy of 96.4% was achieved. For the CNN-LSTM+S model, after augmentation of the training data, the CNN-LSTM+S model achieved an accuracy of 91.3% after 150 epochs, with training and validation losses of 0.114 and 0.200 respectively.

The CNN-LSTM+S network was also trained in the absence of augmented data, and achieved a similar accuracy, but training was much more unstable and there were large discrepancies in the training and validation losses. This demonstrates the improvements that can be realised as a result of resampling the data for training. Although the CNN-LSTM+S model achieved a lower accuracy, the average inference time is significantly reduced, and allows for the recognition of activity sequences of different lengths.

8.1. Future Work

For future work, further analysis of the efficacy of the designed approach is of priority. A larger data set would prove extremely useful in further validation of the results. Aside from accuracy, other performance metrics could be used here to provide a fuller picture of the advantages and disadvantages of the chosen approach. At present, the research team has applied for ethics approval, which if granted, could allow for collection of much more activity data to aid in building more robust classification models.

Validation of the multiple pass approach for identifying activity ends will be investigated.
This could potentially lead to real time recognition capabilities for complex activity sequences with additional work.

9. Conclusions

Although the data set was limited in size, promising results were generated as part of this research project. This can help to pave the way for future developments in the field of HAR, particularly with regard to activity segmentation methods and the recognition of complex activity sequences.

References

[1] C. Yu, Z. Xu, K. Yan, Y. -R. Chien, S. -H. Fang and H. -C. Wu, "Noninvasive Human Activity Recognition Using Millimeter-Wave Radar," in *IEEE Systems Journal*, vol. 16, no. 2, pp. 3036-3047, June 2022.

[2] J. Wang, Y. Zhao, X. Ma, Q. Gao, M. Pan and H. Wang, "Cross-Scenario Device-Free Activity Recognition Based on Deep Adversarial Networks," in *IEEE Transactions on Vehicular Technology*, vol. 69, no. 5, pp. 5416-5425, May 2020.

[3] J. -H. Li, L. Tian, H. Wang, Y. An, K. Wang and L. Yu, "Segmentation and Recognition of Basic and Transitional Activities for Continuous Physical Human Activity," in *IEEE Access*, vol. 7, pp. 42565-42576, 2019.

[4] Bhavanasi, G., Werthen-Brabants, L., Dhaene, T. *et al.* Patient activity recognition using radar sensors and machine learning. *Neural Comput & Applic* 34, 16033–16048 (2022).

[5] R. Zhang and S. Cao, "Real-Time Human Motion Behavior Detection via CNN Using mmWave Radar," in *IEEE Sensors Letters*, vol. 3, no. 2, pp. 1-4, Feb. 2019, Art no. 3500104.

[6] D. Wang, J. Yang, W. Cui, L. Xie and S. Sun, "Multimodal CSI-Based Human Activity Recognition Using GANs," in *IEEE Internet of Things Journal*, vol. 8, no. 24, pp. 17345-17355, 15 Dec.15, 2021.

[7] X. Cheng, B. Huang and J. Zong, "Device-Free Human Activity Recognition Based on GMM-HMM Using Channel State Information," in *IEEE Access*, vol. 9, pp. 76592-76601, 2021.

[8] F. Luo, S. Poslad and E. Bodanese, "Temporal Convolutional Networks for Multiperson Activity Recognition Using a 2-D LIDAR," in *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7432-7442, Aug. 2020.

[9] H. Yan, Y. Zhang, Y. Wang and K. Xu, "WiAct: A Passive WiFi-Based Human Activity Recognition System," in *IEEE Sensors Journal*, vol. 20, no. 1, pp. 296-305, 1 Jan.1, 2020.

[10] S. -H. Cheng, M. A. Sarwar, Y. -A. Daraghmi, T. -U. İk and Y. -L. Li, "Periodic Physical Activity Information Segmentation, Counting and Recognition From Video," in *IEEE Access*, vol. 11, pp. 23019-23031, 2023.

[11] J. Huang, S. Lin, N. Wang, G. Dai, Y. Xie and J. Zhou, "TSE-CNN: A Two-Stage End-to-End CNN for Human Activity Recognition," in *IEEE Journal of Biomedical and Health Informatics*, vol. 24, no. 1, pp. 292-299, Jan. 2020.

[12] T. Huynh-The, C. -H. Hua, N. A. Tu and D. -S. Kim, "Physical Activity Recognition With Statistical-Deep Fusion Model Using Multiple Sensory Data for Smart Health," in *IEEE Internet of Things Journal*, vol. 8, no. 3, pp. 1533-1543, 1 Feb.1, 2021.

[13] P. Zhao, C. X. Lu, B. Wang, N. Trigoni and A. Markham, "CubeLearn: End-to-end Learning for Human Motion Recognition from Raw mmWave Radar Signals," in *IEEE Internet of Things Journal*.

[14] J. -T. Yu, L. Yen and P. -H. Tseng, "mmWave Radar-based Hand Gesture Recognition using Range-Angle Image," *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*, Antwerp, Belgium, 2020, pp. 1-5.

[15] A. Ali *et al*., "End-to-End Dynamic Gesture Recognition Using MmWave Radar," in *IEEE Access*, vol. 10, pp. 88692-88706, 2022.

[16] Baiju Yan, Peng Wang, Lidong Du, Xianxiang Chen, Zhen Fang, Yirong Wu, mmGesture: Semi-supervised gesture recognition system using mmWave radar, Expert Systems with Applications, Volume 213, Part B, 2023.

[17] A. Ninos, J. Hasch and T. Zwick, "Real-Time Macro Gesture Recognition Using Efficient Empirical Feature Extraction With Millimeter-Wave Technology," in *IEEE Sensors Journal*, vol. 21, no. 13, pp. 15161-15170, 1 July1, 2021.

[18] D. Salami, R. Hasibi, S. Palipana, P. Popovski, T. Michoel and S. Sigg, "Tesla-Rapture: A Lightweight Gesture Recognition System from mmWave Radar Sparse Point Clouds," in *IEEE Transactions on Mobile Computing*.

[19] H. Xie, P. Han, C. Li, Y. Chen and S. Zeng, "Lightweight Midrange Arm-Gesture Recognition System From mmWave Radar Point Clouds," in *IEEE Sensors Journal*, vol. 23, no. 2, pp. 1261-1270, 15 Jan.15, 2023.

[20] A. Sengupta, F. Jin, R. Zhang and S. Cao, "mm-Pose: Real-Time Human Skeletal Posture Estimation Using mmWave Radars and CNNs," in *IEEE Sensors Journal*, vol. 20, no. 17, pp. 10032-10044, 1 Sept.1, 2020.

[21] Cong Shi, Li Lu, Jian Liu, Yan Wang, Yingying Chen, Jiadi Yu, mPose: Environment- and subject-agnostic 3D skeleton posture reconstruction leveraging a single mmWave device, Smart Health, Volume 23, 2022.

[22] T. Instruments, “IWR1843,” IWR1843 data sheet, product information and support | TI.com, https://www.ti.com/product/IWR1843 (accessed Oct. 13, 2023).

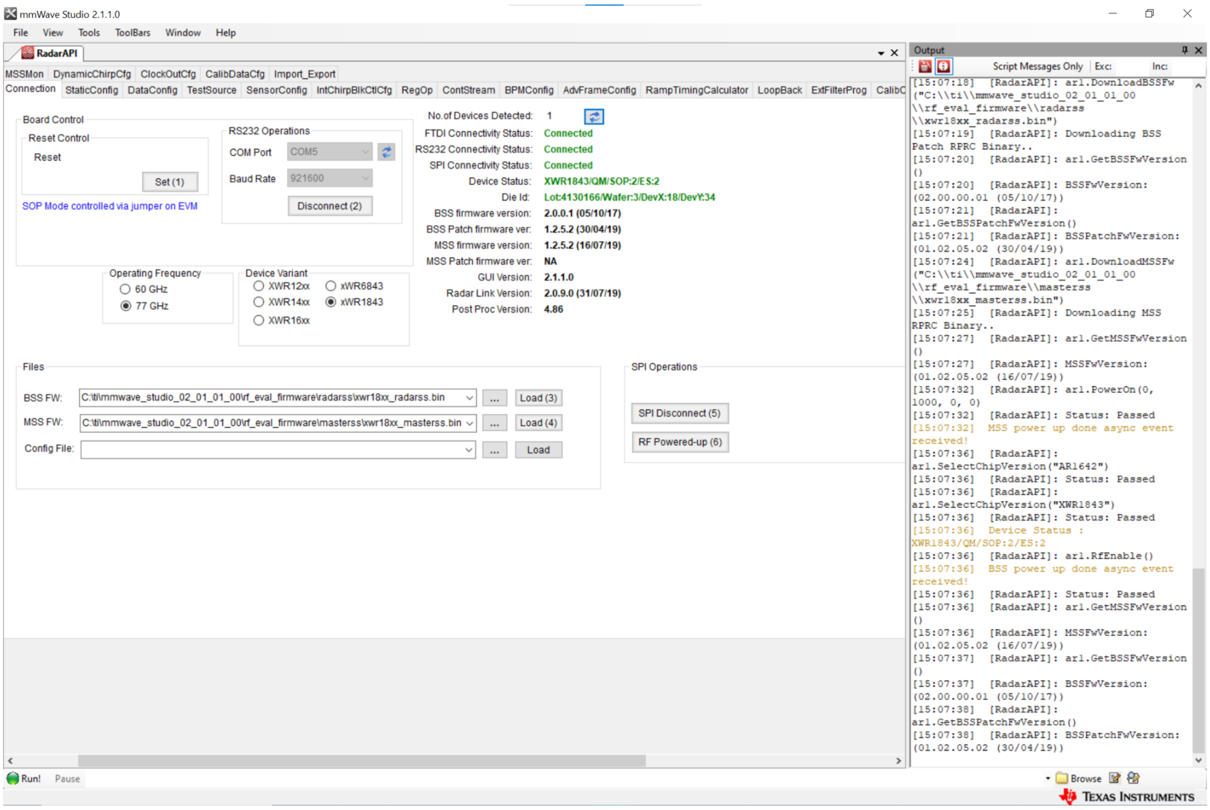
[23] T. Instruments, “IWR1843BOOST,” IWR1843BOOST Evaluation board | TI.com, https://www.ti.com/tool/IWR1843BOOST (accessed Oct. 13, 2023).

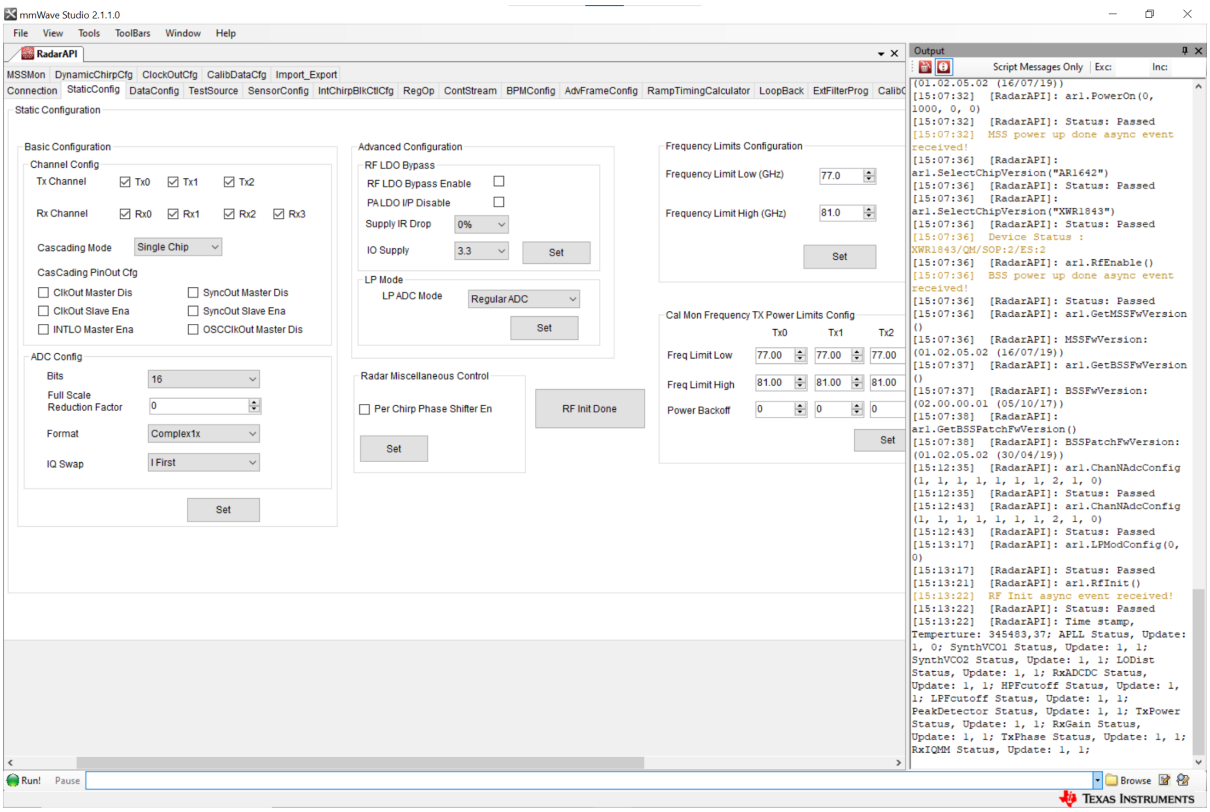
[24] T. Instruments, “DCA1000EVM,” DCA1000EVM Evaluation board | TI.com, https://www.ti.com/tool/DCA1000EVM (accessed Oct. 13, 2023).

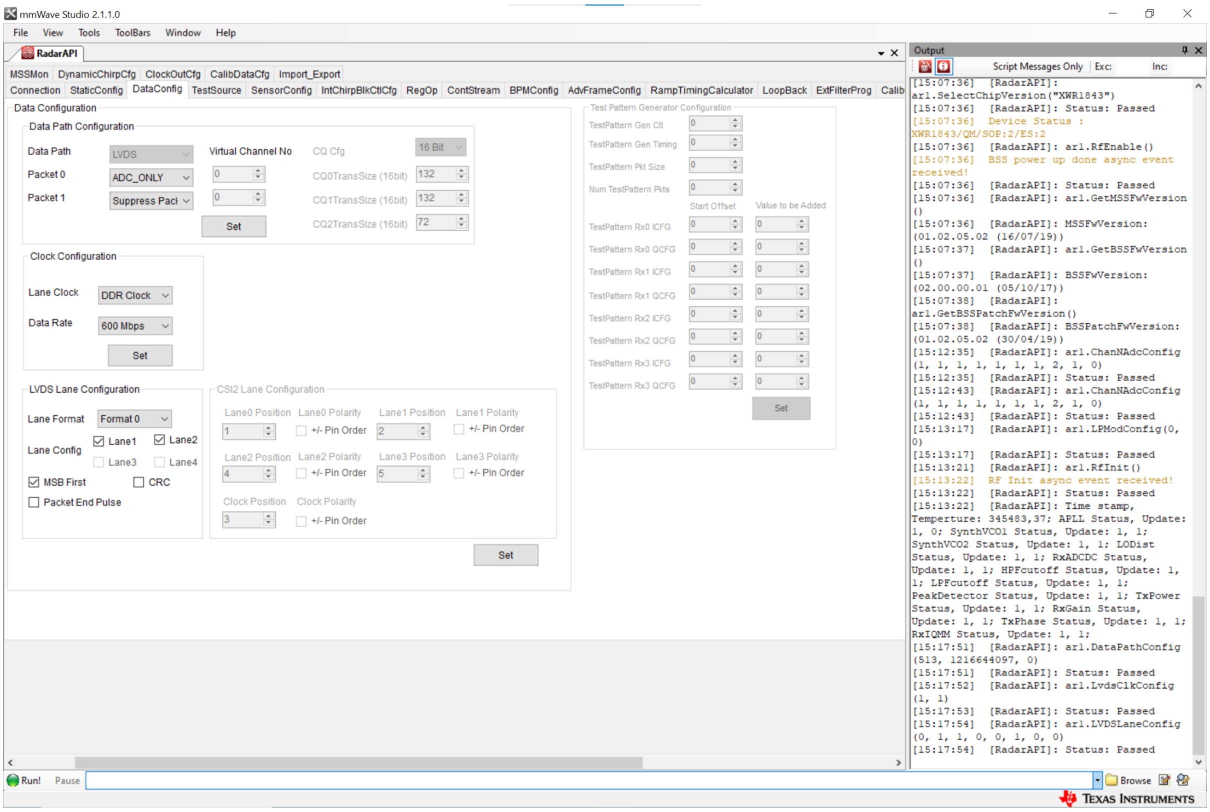
[25] L. Tan and J. Jiang, “Signal sampling and quantization,” Digital Signal Processing, pp. 13–58, 2019. doi:10.1016/b978-0-12-815071-9.00002-6

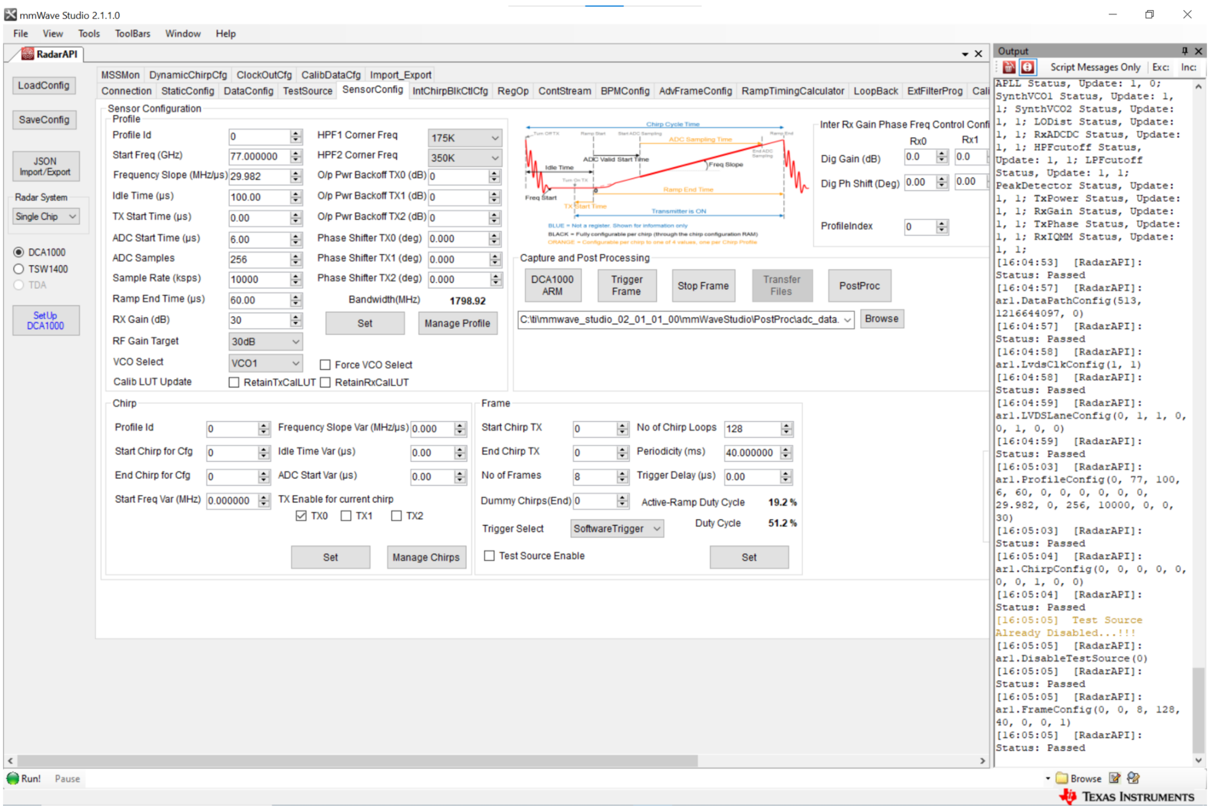
Appendix

mmWave studio settings:






mmWave Studio 2.1.1.0
File View Tools ToolBars Window Help
RadarAPI
MSSMon DynamicChirpCfg ClockOutCfg CalibDataCfg Import_Export
Connection StaticConfig DataConfig TestSource SensorConfig IntChirpBlkCtlCfg RegOp ContStream BPMConfig AdvFrameConfig RampTimingCalculator LoopBack ExtFilterProg Calib
Data Configuration
Data Path Configuration
Data Path LVDS
Virtual Channel No
CQ Cfg 16 Bit
Packet 0 ADC_ONLY
CQ0TransSize (16bit) 132
Packet 1 Suppress Pacl
CQ1TransSize (16bit) 132
Set
CQ2TransSize (16bit) 72
Clock Configuration
Lane Clock DDR Clock
Data Rate 600 Mbps
Set
LVDS Lane Configuration
Lane Format Format 0
Lane Config
Lane1 Lane2 Lane3 Lane4
MSB First CRC
Packet End Pulse
CSI2 Lane Configuration
Lane0 Position Lane0 Polarity Lane1 Position Lane1 Polarity
+/- Pin Order
Lane2 Position Lane2 Polarity Lane3 Position Lane3 Polarity
Clock Position Clock Polarity
Set
Test Pattern Generator Configuration
TestPattern Gen Ctl
TestPattern Gen Timing
TestPattern Pkt Size
Num TestPattern Pkts
Start Offset Value to be Added
TestPattern Rx0 ICFG
TestPattern Rx0 QCFG
TestPattern Rx1 ICFG
TestPattern Rx1 QCFG
TestPattern Rx2 ICFG
TestPattern Rx2 QCFG
TestPattern Rx3 ICFG
TestPattern Rx3 QCFG
Set
Output
Script Messages Only Exc: Inc:
[15:07:36] [RadarAPI]:
ar1.SelectChipVersion("XWR1843")
[15:07:36] [RadarAPI]: Status: Passed
[15:07:36] Device Status :
XWR1843/QM/SOP:2/ES:2
[15:07:36] [RadarAPI]: ar1.RfEnable()
[15:07:36] BSS power up done async event
received!
[15:07:36] [RadarAPI]: Status: Passed
[15:07:36] [RadarAPI]: ar1.GetMSSFwVersion
()
[15:07:36] [RadarAPI]: MSSFwVersion:
(01.02.05.02 (16/07/19))
[15:07:37] [RadarAPI]: ar1.GetBSSFwVersion
()
[15:07:37] [RadarAPI]: BSSFwVersion:
(02.00.00.01 (05/10/17))
[15:07:38] [RadarAPI]:
ar1.GetBSSPatchFwVersion()
[15:07:38] [RadarAPI]: BSSPatchFwVersion:
(01.02.05.02 (30/04/19))
[15:12:35] [RadarAPI]: ar1.ChanNAdcConfig
(1, 1, 1, 1, 1, 1, 1, 2, 1, 0)
[15:12:35] [RadarAPI]: Status: Passed
[15:12:43] [RadarAPI]: ar1.ChanNAdcConfig
(1, 1, 1, 1, 1, 1, 1, 2, 1, 0)
[15:12:43] [RadarAPI]: Status: Passed
[15:13:17] [RadarAPI]: ar1.LPModConfig(0,
0)
[15:13:17] [RadarAPI]: Status: Passed
[15:13:21] [RadarAPI]: ar1.RfInit()
[15:13:22] RF Init async event received!
[15:13:22] [RadarAPI]: Status: Passed
[15:13:22] [RadarAPI]: Time stamp,
Temperture: 345483,37; APLL Status, Update:
1, 0; SynthVCO1 Status, Update: 1, 1;
SynthVCO2 Status, Update: 1, 1; LODist
Status, Update: 1, 1; RxADCDC Status,
Update: 1, 1; HPFcutoff Status, Update: 1,
1; LPFcutoff Status, Update: 1, 1;
PeakDetector Status, Update: 1, 1; TxPower
Status, Update: 1, 1; RxGain Status,
Update: 1, 1; TxPhase Status, Update: 1, 1;
RxIQMM Status, Update: 1, 1;
[15:17:51] [RadarAPI]: ar1.DataPathConfig
(513, 1216644097, 0)
[15:17:51] [RadarAPI]: Status: Passed
[15:17:52] [RadarAPI]: ar1.LvdsClkConfig
(1, 1)
[15:17:53] [RadarAPI]: Status: Passed
[15:17:54] [RadarAPI]: ar1.LVDSLaneConfig
(0, 1, 1, 0, 0, 1, 0, 0)
[15:17:54] [RadarAPI]: Status: Passed
Run! Pause
Browse
Texas Instruments


mmWave Studio 2.1.1.0
File View Tools ToolBars Window Help
RadarAPI
LoadConfig
SaveConfig
JSON Import/Export
Radar System
Single Chip
DCA1000
TSW1400
TDA
SetUp DCA1000
MSSMon DynamicChirpCfg ClockOutCfg CalibDataCfg Import_Export
Connection StaticConfig DataConfig TestSource SensorConfig IntChirpBlkCtlCfg RegOp ContStream BPMConfig AdvFrameConfig RampTimingCalculator LoopBack ExtFilterProg Cali
Sensor Configuration
Profile
Profile Id 0
Start Freq (GHz) 77.000000
Frequency Slope (MHz/µs) 29.982
Idle Time (µs) 100.00
TX Start Time (µs) 0.00
ADC Start Time (µs) 6.00
ADC Samples 256
Sample Rate (ksps) 10000
Ramp End Time (µs) 60.00
RX Gain (dB) 30
RF Gain Target 30dB
VCO Select VCO1
Calib LUT Update RetainTxCalLUT RetainRxCalLUT
HPF1 Corner Freq 175K
HPF2 Corner Freq 350K
O/p Pwr Backoff TX0 (dB) 0
O/p Pwr Backoff TX1 (dB) 0
O/p Pwr Backoff TX2 (dB) 0
Phase Shifter TX0 (deg) 0.000
Phase Shifter TX1 (deg) 0.000
Phase Shifter TX2 (deg) 0.000
Bandwidth(MHz) 1798.92
Set Manage Profile
Force VCO Select
Inter Rx Gain Phase Freq Control Confi
Rx0 Rx1
Dig Gain (dB) 0.0 0.0
Dig Ph Shift (Deg) 0.00 0.00
ProfileIndex 0
Capture and Post Processing
DCA1000 ARM
Trigger Frame
Stop Frame
Transfer Files
PostProc
C:\ti\mmwave_studio_02_01_01_00\mmWaveStudio\PostProc\adc_data.
Browse
Chirp
Profile Id 0
Start Chirp for Cfg 0
End Chirp for Cfg 0
Start Freq Var (MHz) 0.000000
Frequency Slope Var (MHz/µs) 0.000
Idle Time Var (µs) 0.00
ADC Start Var (µs) 0.00
TX Enable for current chirp
TX0 TX1 TX2
Set Manage Chirps
Frame
Start Chirp TX 0
End Chirp TX 0
No of Frames 8
Dummy Chirps(End) 0
Trigger Select SoftwareTrigger
Test Source Enable
No of Chirp Loops 128
Periodicity (ms) 40.000000
Trigger Delay (µs) 0.00
Active-Ramp Duty Cycle 19.2 %
Duty Cycle 51.2 %
Set
Output
Script Messages Only Exc: Inc:
APLL Status, Update: 1, 0;
SynthVCO1 Status, Update: 1,
1; SynthVCO2 Status, Update:
1, 1; LODist Status, Update:
1, 1; RxADCDC Status, Update:
1, 1; HPFcutoff Status,
Update: 1, 1; LPFcutoff
Status, Update: 1, 1;
PeakDetector Status, Update:
1, 1; TxPower Status, Update:
1, 1; RxGain Status, Update:
1, 1; TxPhase Status, Update:
1, 1; RxIQMM Status, Update:
1, 1;
[16:04:53] [RadarAPI]:
Status: Passed
[16:04:57] [RadarAPI]:
ar1.DataPathConfig(513,
1216644097, 0)
[16:04:57] [RadarAPI]:
Status: Passed
[16:04:58] [RadarAPI]:
ar1.LvdsClkConfig(1, 1)
[16:04:58] [RadarAPI]:
Status: Passed
[16:04:59] [RadarAPI]:
ar1.LVDSLaneConfig(0, 1, 1, 0,
0, 1, 0, 0)
[16:04:59] [RadarAPI]:
Status: Passed
[16:05:03] [RadarAPI]:
ar1.ProfileConfig(0, 77, 100,
6, 60, 0, 0, 0, 0, 0, 0,
29.982, 0, 256, 10000, 0, 0,
30)
[16:05:03] [RadarAPI]:
Status: Passed
[16:05:04] [RadarAPI]:
ar1.ChirpConfig(0, 0, 0, 0, 0,
0, 0, 1, 0, 0)
[16:05:04] [RadarAPI]:
Status: Passed
[16:05:05] Test Source
Already Disabled...!!!
[16:05:05] [RadarAPI]:
ar1.DisableTestSource(0)
[16:05:05] [RadarAPI]:
Status: Passed
[16:05:05] [RadarAPI]:
ar1.FrameConfig(0, 0, 8, 128,
40, 0, 0, 1)
[16:05:05] [RadarAPI]:
Status: Passed
Run! Pause
Browse
Texas Instruments

Research Table

Refs	Activities	Activity Segmentation Method
[1]	walk, fall, get up, stand to sit, sit to stand, sit to lie, lie to sit	Sliding window of 1.2s, with a sliding factor of 3 frames (0.3s). 12 frames are used in total to make up feature of dimension 12 x 50 x 50 x 30.
[3]	walking, walking upstairs, walking downstairs, sitting, standing, laying sit to lie, lie to sit, stand to sit, sit to stand, stand to lie, lie to stand	Sliding window with duration of 6s. Followed by a sliding window of (2, 3, 4)s fixed size with 50% overlap to segment long period activities.
[4]	walk to room, fall on the floor, stand up from the floor, walk to chair, sit down on chair, stand up from chair, walk to bed, sit down on bed, stand up from bed, get in bed, lie in bed, roll in bed, sit in bed, get out bed	k = 40 frames (3.7 seconds) found as optimal length. Padding is used if the sample length is less than k.
[5]	human walking and vanish from radar, human waving hands when standing or sitting, human sitting to standing and walking transition, human walking back and forth, no micro-Doppler detections, complex detections including all behaviours	No mention of any segmentation method being employed, other than buffering frames.
[7]	Collected dataset: calling, squatting, walking, stand-fall, walk-fall Comparison 1: lie down, fall, walk, run, sit down, stand up Comparison 2: boxing, empty, walking, pushing, waving	Activity samples are 10 seconds long. Activities are performed from the 4th to the 7th second, the volunteers are asked to remain still the rest of the time.
[9]	pushing, waving, kicking, running, falling, boxing, sitting, picking, walking, empty	An AACA algorithm is introduced, to segment periods of activity from periods of inactivity. Sliding window of size 50, step 1 is used.
[11]	walking, jogging, standing, sitting, ascending stairs, descending stairs	Sliding window of size (64, 3) is used. Window contains a fixed 64 points per axis, at 20Hz the time scale is 3.2s. Window has overlap of 50%.
[14]	grab, clockwise turning, tilt left and right, draw to the left, falling, arc, clap, counterclockwise turning, draw to the right, lifting, thumb-index finger, thumb-little finger	Gesture measurement length is set to 64 frames, or about 2.6 seconds. Gestures are assumed to fit within this 2.6s.
[15]	radial circle, tangential circle, double pinch, single pinch, left-to-right swipe, right-to-left swipe	Gesture samples are performed in a window of 125 frames (~5s). Gestures length is considered to not exceed 50 frames (~2s).

[16]	Positive: push, pull, left swipe, right swipe, clockwise turning, anticlockwise turning Negative: lifting arms, sitting, standing, walking, waving hands	Temporal sampling is used to align gesture durations. Sampled gesture sequences contain 30 frames (1.5s) of heatmaps.
[17]	random movement/walk, pull, push, swipe up, swipe down, swipe right, swipe left, rotate, wave, push-pull	Minimum duration of 200ms before a gesture is detected, total event duration set equal to 50 frames (1s at 50Hz).
[18]	Used three existing datasets, Pantomime, RadHar, mHomeGes. Included gestures, arm gestures and hand gestures.	Best results were achieved by setting a fixed input of 32 frames for both datasets (~1.07s and 0.533s, 0.8s) for all datasets.
[19]	One arm: draw a circle clockwise lifting the right arm, draw a circle anticlockwise lifting the right arm, lift both arms then lateral down Mirrored for both arms: lift right arm then down, down right arm then lift, push right arm then pull, pull right arm then push, pull right arm twice, push right arm twice, right arm outward clockwise circle, right arm outward anticlockwise circle	Minimum gesture length of 350ms. Point cloud data is aggregated for the entire duration of an activity.
[20]	walking, left-arm swing, right-arm swing, both-arms swing	No temporal dependency for posture estimation, therefore, single frames are fed to the CNN network.
[21]	lifting left/right arm to the front for 45/90/180 degrees, lifting left/right arm from the side for 45/90/180 degrees, lifting left/right leg for 45/90 degrees, waving hands, walking, random moving	No temporal dependency for posture estimation, therefore, single frames are fed to the CNN network.