Classification of radar targets using micro-doppler signatures
Final Report, EEE 505 Spring, 2022

Shammi Doly, Willie Chang, Abinaya Subramaniam

Abstract:

We study the efficacy of the methods described in [1] for different Time-frequency representation (TFR) and windows functions. We investigate the continuous wavelet transform (CWT) and generate the scalogram for the radar targets. We compare the visual representation of the scalogram with the corresponding spectrogram as a benchmark. We further solve a corner case of overlapping signatures by doing main body return cancellation using two methods - singular value decomposition (SVD) and delayed conjugated multiplication (DCM). We show proof of concept for SVD and using the car-pedestrian. In addition, we also simulate DCM as shown in [2] to understand parameter estimation.

1. Introduction

Detection and classifications of vulnerable road users like pedestrians and bicyclists have become an urgent research problem because of the recent development of autonomous vehicles. Researchers have explored numerous sensing approaches to address the issues, including computer vision, laser scanners, and automotive radar technologies. Radar-based road's vulnerable target classification using micro-Doppler signatures is one of the most reliable approaches because, unlike other sensors, radar sensors aren't affected by weather conditions. With the ability to detect obstructions like glass, they can "see" through walls. Advanced Machine learning (ML) based algorithms might be the best option to classify overlapping target signatures and extract unique features using the spectrogram of radar returns.

The source of micro motion of an object or target can be a rotating propeller of a fixed-wing aircraft, rotating rotor blades of a helicopter, a rotating antenna, flapping wings of birds, the swinging arms and legs of a walking person, or other causes [3]. Micro-Doppler (MD) provides many detailed radar image features in addition to those associated with the translation for example in [3] authors presents a simplified model of dismount micro-doppler and RCS. There have been a number of literature on human movement analysis by using micro-Doppler features [3]. In [2], the authors investigate the effect of sparsity-driven time–frequency analysis on hand gesture classification using the support vector machine (SVM). However, the extraction of useful features form the interested target is always a challenging task. Specifically in a situation where signatures overlaps because of the presence of multiple targets of interest or objects. We extensively study different methods on micro-Doppler signature based target classification, recognition, and identification, especially for vehicle, human, and animal motion cited in [4, 5]. In [5], a frequency domain classification is performed using two distinct approaches: a generative statistical classification algorithm based on Gaussian mixture models (GMMs) and a discriminative statistical classification algorithm based on support vector machines (SVMs). The time domain classification is used in [4] based on using Dynamic Time Warping (DTW), which is comparable of k- Nearest Neighbour (k-NN) algorithm. However, the performance of any ML-based model depends on the quality and quantity of the data used for training purposes. Radar raw data collection is always challenging, especially for vulnerable road users like pedestrians. Because often, it can get confused with roadside clutters. A discussion on measurement and generating radar data cube for training purposes are presented in section 4. The micro-Doppler signatures exhibit reliable distinguishing features for various classes of target's data set.

The main scope of this project is equally split into three parts among the project members. Each member has done a comprehensive literature review and contributed to writing and generating results and analysis in their the assigned task as mentioned below:

1.1. Task Management and Key Contribution

1.1.1 Shammi-

In the first thrust, we present a spectrogram-based classification study of three (e.g., pedestrian, cyclist, and vehicle) representatives from the classes of mobile targets typically encountered on roads using wavelet transformation. We study the efficacy of the methods described in [1] for different Time-frequency representation (TFR) and windows functions. We compare the visual representation of the scalogram with the corresponding spectrogram in [1] as a benchmark. We also propose Wigner-Ville distribution to generate the smoothed and detailed spectrogram for classification purposes as the next step of the project.

1.1.2 Willie-

We elaborately study the radar data cube and dataset pre-processing methods. We write our Matlab scripts to generate our own data set for training and classification purposes. We also study the STFT with different windows and different length with the source codes for better understanding the target and implementations. Furthermore, we study the neural architectures for the (e.g., Convolutional Neural Network Classification (CNN)) classification problem. Because of the tight timing schedule, we could not implement more models. However, we leave this as a promising research problem to consider in the future in Radar-based detection and classification.

1.1.3 Abinaya-

In the final thrust, we present a solution to handle the overlapping signatures in the presence of the Gaussian noise (car noise). We describe the problem shown in [1] and implement two methods to cancel car noise - Singular Value decomposition (SVD) and Delayed Conjugated Multiplication (DCM) [2]. We show proof of concept for DCM by simulating the scenario given in [2]. In [1], car noise cancellation has not been shown. We investigate the usefulness of SVD and DCM for the same.

2. Radar Basic

RAdio **D**etection **A**nd **R**anging (**RADAR**) is an electromagnetic device, used for the detection, locating, tracking, and, recognizing the various kinds of targets form reasonable distances. The targets can be regular or autonomous vehicles, humans, or any vital sign. Besides determining the presence, location, and velocity of such objects, radar can sometimes obtain their size and shape. What distinguishes radar from optical and infrared sensing devices is its ability to detect faraway objects under adverse weather conditions and to determine their range, or distance, with precision [6]. Radars can be divided in two types based on the signals it uses.

- Pulsed Radar: where radar transmits a sequence of pulses of radio frequency energy
- Continuous Wave (CW) Radar: where radar uses the continuous signals as the transmitting signals

CW radars typically use separate transmitter and receiver in the transmission process because it is difficult to achieve the full sensitivity while transmitting the high power signals. The pulsed radars use a single special antenna for both transmitting and receiving purposes. This type of antenna is called the 'Transceiver' (transmitter+receiver) [6]. Based on this Radars can be again classified in two categories.

- Monostatic Radar: In monostatic radar antenna and receiver and associated with other antennas are collocated. Pulsed monostatic radars use transceiver whereas the monostatic CW radars use a shield to seperate the transmitter from the receiver [6].
- Bistatic Radar: In bistatic radars the transmitter and receiver are located in a large distance (more than 1km) [6].

2.1. Radar signals and working principle

Target moving in front of radar introduce a frequency shift in the returned backs scattering signal due to the Doppler effect phenomenon. The radar return is received at the receiver end, and after the demodulation technique, the signal is ready to retrieve the target's information. The resultant signal at the receiver is the baseband signal. The radar signal's waveform is responsible for determining the target's range and radial velocity (range rate) accuracy, resolution, and ambiguity. Range rate is associated with the Doppler shift of the signal received.

2.2. Range-delay & Velocity-Doppler

To determine the distance between the radar and a target, the delay of the echoed pulse is used as shown in Figure 1. When in the free space the target is located as a point object then the relationship between range R and delay τ is simply expressed as the following equation;

$$Range, R = C * \tau/2, \tag{1}$$

where C= 3×108m/s, is the speed of light in the free space. The factor 1/2 is due to the fact that the 2 radar signal traverses the distance R twice (round trip). Doppler effect is used to measure speed in Radar sensors. When the fixed-frequency radio wave sent from the sender continuously strikes an object that is moving towards or away from the sender, the frequency of the reflected radio wave will be changed. This frequency shift is known as Doppler effect [10], as shown in

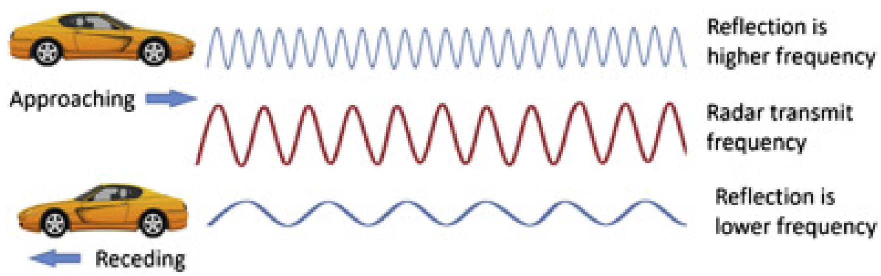


Figure 1: Doppler effect (figure courtesy [10])

Figure 1.

2.3. Micro-Doppler Signatures

The radar micro-Doppler effect is a phenomenon caused by the micro-motion of a target or its parts and the bulk translation. The Doppler frequency shift is usually measured in the frequency domain by taking the Fourier transform of the returned signal. In the Fourier spectrum, the peak component indicates the Doppler frequency shift induced by the radial velocity of the object's micro-motion. The micro-Doppler effect can be used to detect the kinematics of an object. For example, the speed of vibration of an engine can be detected by using the surface vibration of the car [8].

3. Time-frequency analysis of Radar signals

Different joint time frequency transformations are used to understand the time-varying frequency contents of the signals. Transformations are divided into two classes:

- Linear Time Frequency Transforms: the most well-known transformations belong to this class are The most-known are Short Time Fourier Transform (STFT), Continuous Wavelet Transform (CWT) and Adaptive Time-Frequency Representation.
- Quadratic (Bilinear) Transforms (QTFRs): the Wigner-Ville Distribution (WVD).

3.1. Short-Time Fourier Transformation

One of the best-known time-frequency representations of a time signal is known as " Short Time Fourier Transform - STFT ". SFT is a set of localized FTs obtained by sliding a window in time where the signal assumed stationary over duration of the window. The window is moving over the time domain and the examination of the frequency content of the signal generates a 2-D time-frequency distribution called "spectrogram". The general expression of STFT is expressed as,

$$STFT_x(t,f) = \int (x(\tau)h(\tau - t)e^{-j2f\pi})d\tau, \tag{2}$$

where, $h(\tau - t)$ is the moving window.

Uncertainty Principle:

Another well-known principle in FT is the uncertainty principle or Heisenberg inequality. According to the uncertainty principle, the frequency function and the time function are inversely proportional and their value is equal to the width of the moving window and the frequency bandwidth. If the time duration of a signal $x(t)$ is Δ_t and the frequency bandwidth is Δ_ω then the Fourier Transform $S(\omega)$ is related by:

$$\Delta_t \Delta_\omega \geq 1/2. \tag{3}$$

The major advantages of the STFT are, firstly it is simple and computationally less burdensome as it is equal with the computation of multiple FTs. Secondly, the absence of the 'Cross-Terms(CT)' unlike the bilinear transforms (i.e. Wigner-Ville distribution). One of the major disadvantages of STFT is the processing results are not good in both time and frequency domain. The processed signal can be either analyzed with good time resolution or frequency resolution. To overcome these limitations of the STFT, in order to obtain a multi-resolution analysis, wavelet transforms are used.

3.2. Wavelet Transformation

The concept of Wavelets Transformation (WT) is derived from the Morlet and Grossman, 1984 to analyze short duration waveforms at high frequencies. This is also linear time-scale representation (TSR) $T_x(t,\alpha)$ where, the scale parameter α is important for analyzing signals with different size components. The general expression of the WT is:

$$WT_x(t,\alpha;\Psi) = \int (x(\tau)1/\sqrt{\alpha}\Psi^*((\tau - t)/\alpha))d\tau, \tag{4}$$

where, the scale parameter $\alpha \geq 0$. The WT is classified in two categories.

- Continuous Wavelet Transform,
- Discrete Wavelet Transform

Contrary to the fixed resolution of the STFT, the Continuous Wavelet Transform – CWT is a time frequency representation capable of achieving variable resolution in one domain (time or frequency) and multi resolution in the other domain.

3.3. Wigner-Ville distribution

Wigner-Ville distribution (WVD) is a branch of quadratic or bilinear transformation. Quadratic time–frequency distributions (TFDs) are commonly used for the analysis of time varying non-stationary signals because of their simple interpretation as the distribution of the signal energy in a time–frequency (t–f) plane. Any quadratic representation can be represented as:

$$T_x(\underline{\Theta}) = \int\int K_T(\underline{\Theta}; t_1, t_2) x(t_1) x^*(t_2) dt_1 dt_2; \tag{5}$$

where, kernel $K_T(\underline{\Theta}; t_1, t_2)$ uniquely characterizes $T_x(\underline{\Theta})$. The main reason of introducing the uadratic or bilinear transformation is to address the TF resolution trade-off issues faced in linear TFRs. The main issue with QTFRs is cross terms due to bilinearity. Moreover, Wigner distributions preserve the energy spres E_x as,

$$\int\int W_x(t, f) dt df = E_x. \tag{6}$$

A image enhancement method for suppressing interference terms in the Wigner–Ville distribution is presented in In [7]. The proposed technique adapts the direction of the smoothing kernel locally at each t–f point, so that the smoothing kernel remains aligned with the ridges of the auto-terms.

4. Radar Data Cube

The radar data cube is one of the most convenient ways to present the radar signal in both time and space. It is a three-dimensional block with fast time samples, spatial samples, and slow time samples as different axes. It can also be considered as an extension of a two-dimensional data matrix, including spatial sampling. Figure 2 shows the structure of the radar data cube. The radar signal processing operations are in the Phased Array System Toolbox, which is used to process lower-dimensional subsets of the radar data cube. This lower-dimensional could be either a one-dimensional subvector, for instance, a fast time subvector, or a two-dimensional submatrix, i.e., a data matrix with fast time samples and slow time samples [9].

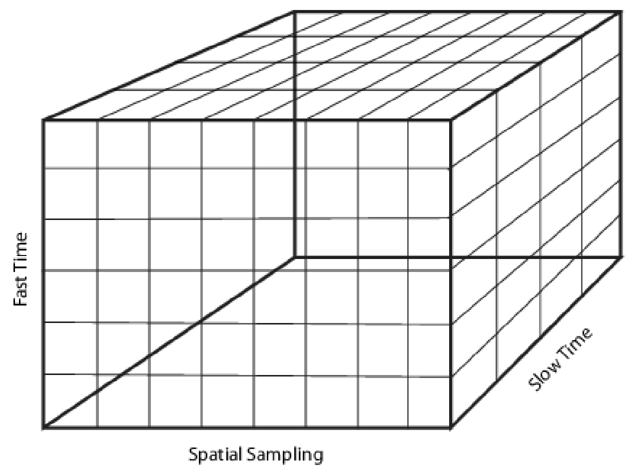


Figure 2: Radar data cube

4.1. Fast time Samples

The fast time data are a set of complex basedband values sampled within a single pulse. Consider the sampling rate as f_s, it should be large enough to avoid aliasing, and the corresponding sampling interval is $T_s = 1/f_s$. Since this is the highest sampling rate in the system, it leads to be given as *fast time*. Fast time sample intervals can also be converted to distance using the propagation speed of the transmitted signal, which is the speed of EM wave, are therefore often referred as *range bins* [9].

4.2. Slow time Samples

The slow time data are a set of complex baseband values as well. However, considering multiple pulses are transmitted from the radar, the slow time samples are sampled within the time interval between the contiguous transmitted signals. This sampled time unit is defined as *pulse repetition interval* (**PRI**), while the sampled frequency unit *pulse repetition frequency* (**PRF**) is given as $PRF = 1/PRI$, where PRF simply gives the unambiguous Doppler spectrum. PRIs are much longer than the fast-time sampling interval, sampling across multiple pulses. They are then referred to as *slow time*. The Nyquist criteria are necessary to avoid aliasing for slow-time sampling [9].

4.3. Spatial Sampling

Phased arrays generally consist of multiple radars to form a radar array. Consider the matrix within a single pulse; each column vector is the fast-time data sampled by f_s and received by a single radar element, while the row vector represents the sample pulse sampled across the radar array elements. These row vectors are the spatial sampling of the incident/received radar signal.

The Nyquist criterion is necessary to avoid aliasing which the radar elements must be separated by less or equal than one-half of the carrier frequency wavelength [9].

5. Classification of micro-Doppler Signature

Machine learning is a method of optimizing a desired model or target with past experiences or examples. Multiple machine learning models and methods have been widely used in commercial activities and research. Machine learning can be separated into three significant paradigms based on the desired outputs and approaches.

- Supervised learning
- Unsupervised learning
- Reinforcement learning

For supervised learning, which the data is labeled, can also be separated into two different classes.

- Regression
- Classification

In this project, since we want to classify the micro Doppler signatures and compare them with the given labels, we propose and study CNN classification methods for this project.

5.1. Convolutional Neural Network Classification

The convolutional neural network (CNN) is a popular method for classifying two-dimensional data or visual images. Typical CNN structures include: *convolutional layers*, *pooling layers*, *connection layers* with *activation functions*. Convolutional layers extract the features of the two-dimensional input data with a designed filter, known as a kernel, and pass the output to the next layer. Pooling layers reduce the input data size to the next feature map. There are two common methods of pooling: max pooling (select the max value of the local cluster into the new feature map) and average pooling (give the mean value of the local cluster into the new feature map). The connection layers and activation functions are considered conventional multi-perceptron, or external network, which works with the extracted features and separates them into different classes to the next layer or final output. Backpropagation is applied to optimize the weights of CNN for better testing accuracy. Some common applications of CNN are object detection, image segmentation, image recognition, pose estimation, and natural language processing.

6. Motion compensation

The goal of the motion compensation is to reduce any Doppler spread from the primary body return so that micro doppler features can be extracted easily. One standard way is to develop a delay alignment algorithm to organize the data in fast and slow times properly. A crude matched filtering will enhance the SINR of all radar pulse returns, which will facilitate good alignment of all pulses. The peak main body returns will be used to estimate the phase due to translation motion, and then this phase will be removed from the data. Another way to cancel the primary body return is by singular value decomposition, where the returns from micro-motion and the main body are separated. Then the principal body return is canceled. One other way to do this is delayed conjugated multiplication, similar to matched filtering. By choosing the suitable delay, the motion parameters like velocity and acceleration of the main body can be estimated and used for cancellation [2]. In this project, the main body is a car, and we present ways to cancel car noise.

6.1. Singular Value Decomposition (SVD)

Singular value decomposition is the basis for principal component analysis, a standard technique used in machine learning that takes high dimensional data and distil it into key features or key correlations so that data that can be used to interpret and understand.

$$\mathbf{A} = \begin{bmatrix} \mathbf{u}_1 & \mathbf{u}_2 & \dots & \mathbf{u_m} \end{bmatrix} \begin{bmatrix} \sigma_1 & 0 & \dots & 0 & 0 & \dots & 0 \\ 0 & \sigma_2 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \dots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & \sigma_r & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 \end{bmatrix} \begin{bmatrix} \mathbf{v_1}^T \\ \mathbf{v_2}^T \\ \vdots \\ \mathbf{v_n}^T \end{bmatrix} \tag{7}$$

Where u_1 is more important than u_2 in terms of its ability to differentiate the variants in X. It is the basis through which the original columns of X can be represented. Every column of V is the Eigen mixture of all the columns of U concerning the corresponding column of X. In this project, SVD is applied on the radar return signal to differentiate the returns from the car, and pedestrian [13].

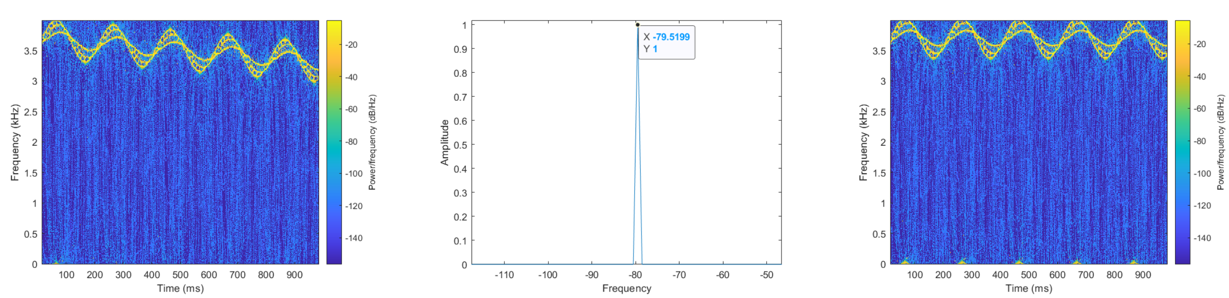
6.2. Delayed Conjugated Multiplication (DCM)

Parameter estimation is done for the problem scenario described in [2]. A space spinning target is considered to perform DCM on the echo signals of the target and the value of translation acceleration is estimated. After estimating acceleration using center frequency in and the derived equation described in [2], an exponential signal is applied to the echo signal to compensate the translational acceleration and the motion compensated echo signal's TFR. The process followed in [2] is as followd - the total radar return signal is converted to baseband and a multiplied with a conjugated delayed version of itself. Delay is chosen in such a way that the microdoppler part of the return signal is cancelled and translation part of the return signal remains. Now, DFT of this output is done to see the peak frequency. With this peak frequency, translation acceleration is derived as:

$$\hat{a} = -\frac{\hat{f}_a \cdot c}{2 f_0 \tau}$$

where τ is the delay, f0 is PRF, $\hat{f}_a$ is peak frequency frm DFT and c is speed of light.

As given in [2], we consider a spinning target with 3 rotational scatterers. The rotational amplitude and initial phase are (0.2 m, 30°), (0.4 m, 45°), and (0.5 m, 60 °), respectively. The residual translational acceleration is assumed to be 20 m/s^2. Figure 3(a) shows the TF representation of return signal before motion compensation. Three sinewaves represent the 3 rotational scatterers and the plot is moving towards right bottom direction because of the translation motion. Figure 3(b) shows DFT of DCM processing and Figure 3(c) shows motion compensated TFR.



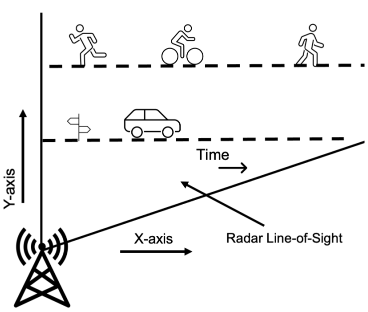
(a) Before motion compensation (b) DFT of DCM processing (c) After motion compensation

Figure 3: Translational Motion Compensation using DCM

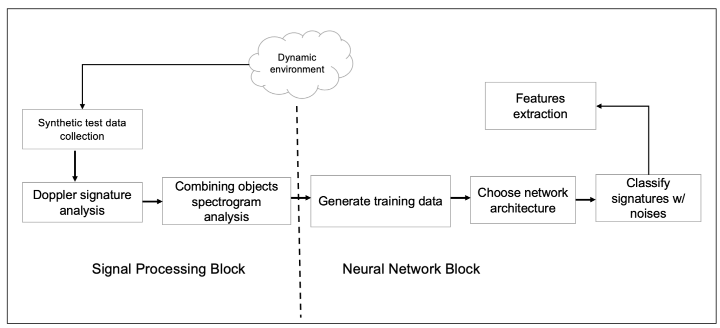
7. Pedestrian and Bicyclist Classification Using Deep Learning

7.1. Problem Specification

This project designs and implements a method to identify and classify targets based on their micro-Doppler signatures using a deep learning network and time-frequency analysis. We consider the example cited in [1] as a baseline experiment. As a case study, we consider a simple problem scenario 4(a), where three different targets (e.g., a pedestrian, a biker, and a car) are present in the radar line-of-sight(LOS). The experiment aims to separate and classify the vulnerable roadside users (e.g., a pedestrian, a biker) from the other road objects (car data) using the micro-Doppler signatures. We consider car data as the noises.



(a) Problem scenario



(b) System block diagram

Figure 4: Pedestrian and bicyclist classification system

There are two major sections in this project as shown in Figure 4(b). One is to generate the micro-Doppler spectrograms from the radar returns using the time-frequency signal analysis. Then feed them to use CNN for target classification. The embedded micro-Doppler information in the radar signal needs to be presented in a format that CNNs can use.

8. Simulation and Results

We implement the pedestrians and bicyclists classification problem described in Section 7.1 based on their micro-Doppler characteristics using a deep learning network and time-frequency analysis methods in MATLAB, [1]. The dataset we use are generated by ourselves, using MATLAB's in built function, **backscatterPedestrian** [11] and **backscatterBicyclist** [11] from Radar Toolbox™. Each of the array size of the data is 500×7104, where the former is the number of fast-time samples with the sampling frequency $f_s = 500MHz$ and the latter is the number of slow-time samples with the slow-time sampling interval $T_{samp} = 2.8153 \times 10^{-4}s$. Since we only simulate one radar element,

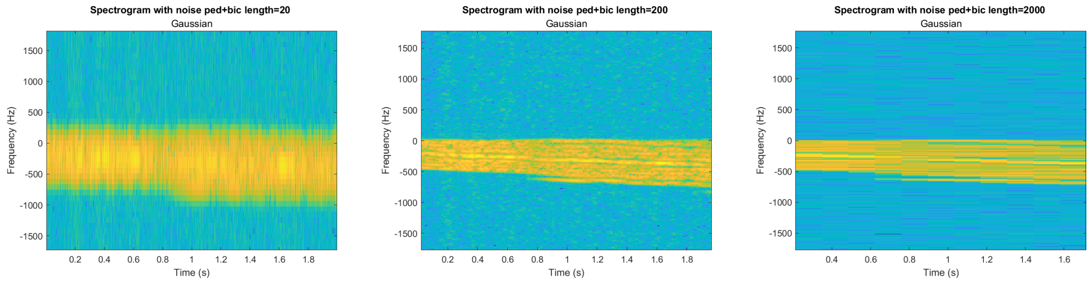
Table 1: Radar parameters for simulation setup

Parameter	**Value**
Bandwidth (B)	25 MHz
Sampling frequency	50 MHz
Carrier frequency	24 GHz
Waveform repetition time	1e-6 sec
Effective temperature (T_{temp})	1000 K
Radar antenna gain	25 dBi
Noise Figure	10
Target cross section	10 m^2
Target process standard deviation (σ_{proc})	100 m
Radar duty factor (δ)	0.01
Simulation time duration	2 sec

the number of the spatial samples is 1. Each of the radar signal spends 245s to generate. Therefore, due to the lack of computation and time resources, we decided to randomly generate 1000 samples from the given labels in [1]. These labels include: one pedestrian, one bicyclist, and one pedestrian + one bicyclist. The helper function **helperDopplerSignatures** [1] computes the short-time Fourier transform (STFT) with Kaiser window to generate the micro-Doppler signature. In order to generate different signatures, we modify this function to implement CWT and nine different windows for STFT. The parameters used in this study are shown in Table 1.

8.1. Performance Comparison STFT vs. CWT

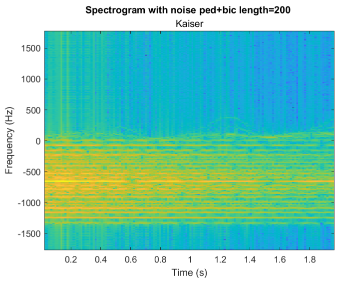
In real-life signal analysis, time-domain signal truncation is always necessary to reduce the spectral leakage and for the modest frequency resolution. Therefore, we perform short-time Fourier transform (STFT) for different types of window functions: *rectangular, triangular, Bartlett, Blackman, Chebyshev, Gaussian, Hamming, Hann, and Kaiser.* We also evaluate the performance of the algorithm for these windows. We measure the algorithm's ability to suppress the scaling of the computational effort with the different window lengths as $w = 20$, $w = 200$, and $w = 2000$, which the window time duration is 40ns, 400ns, and 4μ s.



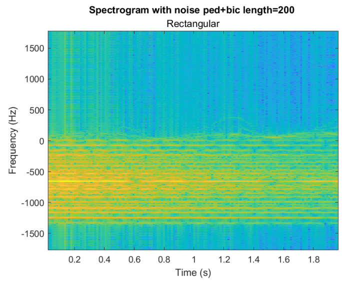
(a) Gaussian window, $w = 20$ (b) Gaussian window, $w = 200$ (c) Gaussian window, $w = 2000$

Figure 5: Spectrogram of a Pedestrian and a bike with different window lengths at sample #851.

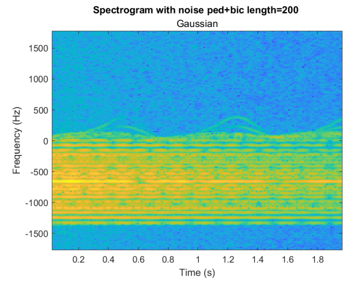
As we discussed in Section 3.1, the STFT has a fixed resolution no matter how the window lengths are. Figure 5 shows the comparison of different window time duration. For $w = 20$, the signature has better time resolution but worse frequency resolution, while for $w = 2000$, the signature has better frequency resolution but worse time resolution. These kind of distortions could lose some important features of the micro-Doppler signature and is hard to recognize and classify them for training and testing process. Proper window length selection, for example, in figure 5(b), could provide clear information both in time and frequency and thus be selected as the signatures for further classification processes.



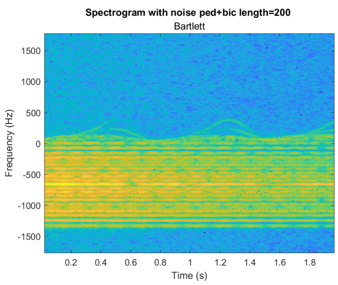
(a) Kaiser window(ref)



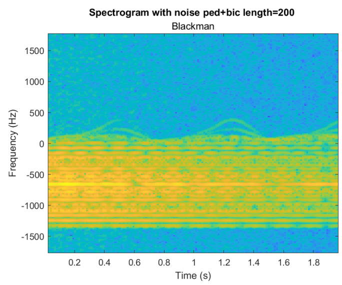
(b) Rectangular window



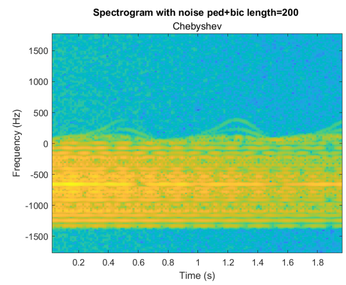
(c) Gaussian window



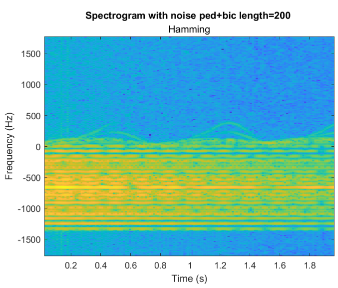
(d) Bartlett window



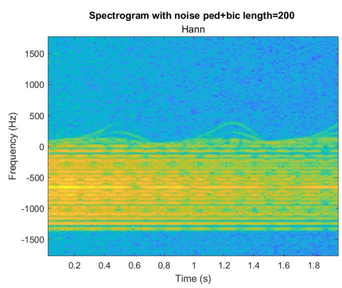
(e) Blackman window



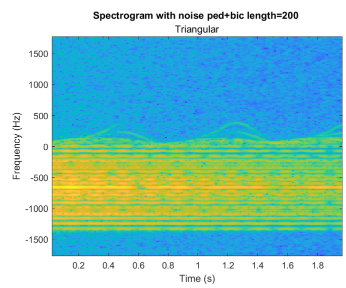
(f) Chebyshev window



(g) Hamming window



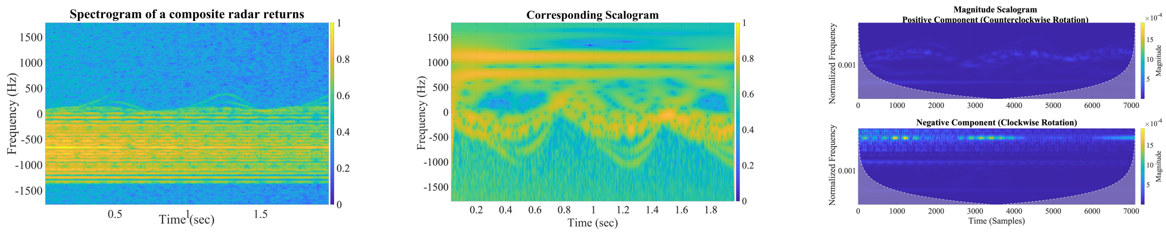
(h) Hann window



(i) Triangular window

Figure 6: Spectrogram of a Pedestrian and a bike with window length = 200 samples.

Aside from the window length, we also compare different windows in STFT to see how the spectral leakage on each column affects the classification results. It is evident that Kaiser, Chebyshev, and rectangular windows have explicit spectral leakage on each column (fast-time samples) and thus affect the feature extractions for classification. Also, we can observe that Gaussian, Bartlett, Hamming, Hanning, and Triangular windows significantly reduce spectral leakage and preserver the features of Pedestrians and bicyclists. CWT [12] uses the inner products to measure the similarity between the interested signal and the analyzing function like the Fourier Transform (FT) does. The best part of CWT is no longer the need to trade-off in the choice of window size or compromise the resolution. In Figure 7(a) the frequency content of the radar return is more accurately captured than the STFT techniques as shown in Figure 7(b). We use MATLAB's CWT [12] and Signal



(a) Because the pedestrian and bicyclist signatures overlap in time and frequency, differentiating between the two objects is difficult.

(b) Distinguishable micro-Doppler signatures of the received signal by using the CWT.

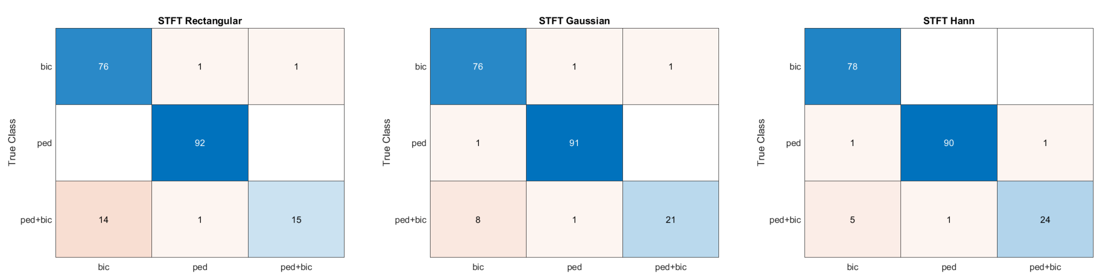
(c) Scalogram for the composite returns for slow-time samples considering the fast- time samples at $N_{fast} = 400$.

Figure 7: Visual comparison of spectrogram vs. scalogram results for the composite radar returns

Processing Toolbox to perform the experiment.

8.2. CNN Model Implementation for classifications

After generating the micro-Doppler signatures, we implement the CNN model for classifications. The CNN structure and training options for STFT are referred to in the documentation from Matlab [1]. We modified some parameters of the training option. For instance, we change the batch size from 128 to 8 and the maximum iterations from 30 to 200 to fit the smaller size of datasets, 1/20 of the given datasets from Matlab [1]. We classify the datasets of nine windows with window length $w = 200$, so that feature extractions in CNN could preserve the features of pedestrians (arms and legs swinging) and bicyclists (feet and wheels rotating at specific frequencies). The datasets are separated 80% into training datasets and 20% into testing datasets. Furthermore, the classification results are shown in table 2.



(a) Rectangular window

(b) Gaussian window

(c) Hann window

Figure 8: Confusion matrix of the classification with 9 windows.

We can also observe the testing results from the confusion matrix in Figure 8.

Table 2: Classification results of STFT

Window	**Accuracy**
Rectangular	91.5%
Triangular	95.5%
Bartlett	93.0%
Blackman	93.0%
Chebyshev	91.5%
Gaussian	94.0%
Hamming	93.0%
Kaiser	92.0%
Hann	96.0%

8.2.1 Challenges in classification

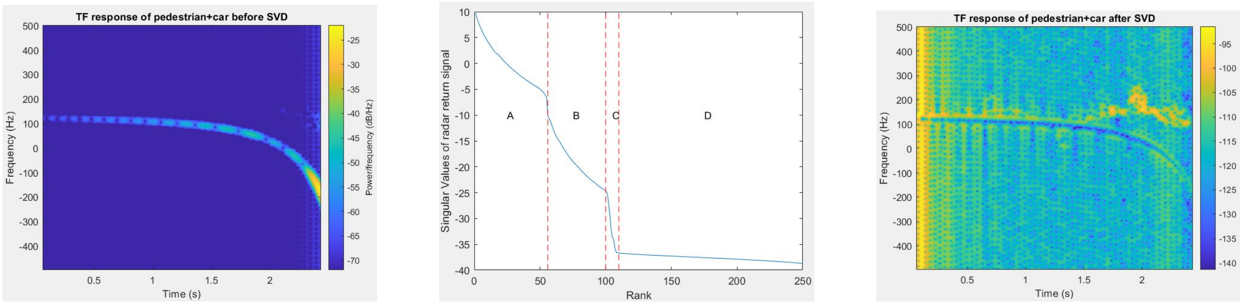
We have the false classified rates at the down-left corner, which has the ground-truth value of pedestrians and bicyclists being recognized as bicyclists only. This is due to the mixture of micro-Doppler signatures of pedestrians covered by the bicyclist. Therefore, during the feature extraction process, the CNN model can not get the features of pedestrians and can cause this misclassification. This also links to the motivation of why we want to apply CWT or other TF representations for MD classification, to preserve the features of the objects better. Due to the time limitation, we could not figure out a proper model for CWT classifications. However, this does give us an idea of implementing different time-frequency representations, and we will work on tuning a proper model for CWT classifications.

8.3. Car Noise Cancellation

So far in our analysis, only pedestrian and bicyclist data were used in the network for classification, but that is not the case in a real-life scenario. Large rigid body like a car is always in the picture. We measure the performance of car noise cancellation in two methods as mention in Section 6. We use Kaiser window and STFT to generate the spectrogram.

8.3.1 Car Noise cancellation using the SVD

When car noise dominates, the deep learning network classifies it as a bicyclist, as shown in Figure 9(a) because the signatures are similar and the bicyclist signature is much weaker. As seen in Figure 9(a), car noise domination is visible, and that is the reason for inefficient classification. To solve this problem, the main body translation motion has to be compensated before the data is fed into the network.



(a) Before SVD

(b) Singular Values of Radar Return

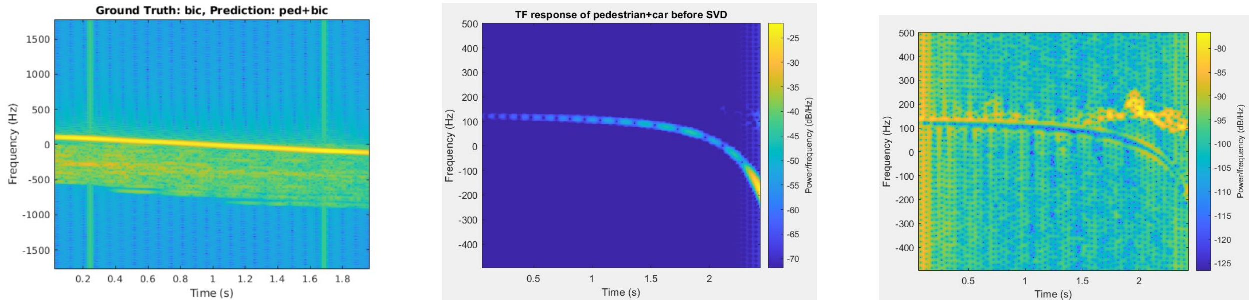
(c) After SVD

Figure 9: Car noise cancellation using SVD

Figure 9(a) shows the time-frequency response of radar return signal of parked car and pedestrian. We see that the pedestrian signature is not visible at all due to string car returns. After applying SVD on total radar return, the singular values over rank are as shown in Figure 9(b). The values are divided into 4 regions A,B,C,D. Since car return is much stronger, region A is classified as car. Region D is extra noise since it is too weak. Regions B and C are a mix of car and noise. Region C is chosen as we intend to look at the weaker pedestrian return clearly. Finally time-frequency response of region C is shown in Figure 9(c).

8.3.2 Car noise Cancellation using DCM approach

To apply DCM on the car pedestrian example, the total radar return was taken and multiplied with a delayed version of the same. This output was matched filtered using a taylor spectrum window and the result is as shown in Figure 10. Comparing Figure 9 and Figure 10, we ee that both the methods show almost similar performance with DCM being slightly better at cancelling car noise and enhancing pedestrian signature.



(a) Time-Frequency response of overlapping signatures with car noise

(b) Before DCM

(c) After DCM

Figure 10: Car noise cancellation using Delayed Conjugated Multiplication

9. Conclusions and Future Work

This project brought us great opportunities to comprehensively study the source project provided in [1]. We explored the idea of micro-Doppler signature classifications using CNN incorporating different types of TFR in the picture. We have faced challenges in generating radar data, spectrogram analysis in CWT, and motion compensation methods. We have successfully debugged some of the challenges. In our future work, we hopefully study those problems more precisely. We noticed that the authors of the source we considered for this project [1] have mentioned a case of overlapping signatures where the classification does not work. It is essential to cancel the main body returns to classify targets using their micro-Doppler signatures. We have discussed two methods to do this, namely singular value decomposition and delayed conjugated multiplication. A proof of concept for both methods has been shown.

The next step would be to implement these methods in the large dataset using more advanced neural network model. We also propose Wigner-Ville distribution to generate the smoothed and detailed spectrogram for classification purposes as the next step of the project.

10. Acknowledgements

We want to express my heartfelt appreciation to our course teacher, **Dr. Antonia Papandreou-Suppappola** for giving us this opportunity to work on a project together and for all the knowledge imparted through this course. We are grateful to her for occasional discussion on the project and course-related problems. We have solved several significant problems in the EEE 505: Time-Frequency Signal Processcourse, for which we are truly indebted to Dr. Antonia.

References

[1] Matlab documentation of Pedestrian and Bicyclist Classification Using Deep Learning ©.

[2] Gu, Fu-Fei, B.Liang, K.Li et al. "Translational Motion Compensation and Micro-Doppler Feature Extraction of Space Spinning Targets." IEEE Geoscience and Remote Sensing Letters, vol. 15, no. 10, pp. 1550–54, 2018

[3] D. Tahmoush and J. Silvious, "Simplified model of dismount microDoppler and RCS," 2010 IEEE Radar Conference, 2010, pp. 31-34, doi: 10.1109/RADAR.2010.5494657.

[4] Smith, Graeme Edward. Radar Target Micro-Doppler Signature Classification. ProQuest Dissertations Publishing, 2008.

[5] Anderson, Michael Glen. Design of Multiple Frequency Continuous Wave Radar Hardware and Micro-Doppler Based Detection and Classification Algorithms. ProQuest Dissertations Publishing, 2008.

[6] Budge, Mervin C., and Shawn R. German. Basic Radar Analysis. Artech House, 2015.

[7] Khan, Nabeel Ali, and Maria Sandsten. "Time–frequency Image Enhancement Based on Interference Suppression in Wigner–Ville Distribution." SIGNAL PROCESSING, vol. 127, Elsevier B.V, 2016, pp. 80–85

[8] Chen, Victor C., et al. Radar Micro-Doppler Signatures: Processing and Applications: Processing and Applications. The Institution of Engineering and Technology, 2014

[9] Matlab documentation of Radar Data Cube. ©.

[10] Dario Camuffo, Chapter 13 - Measuring Wind and Indoor Air Motions, Editor(s): Dario Camuffo,Microclimate for Cultural Heritage (Second Edition),Elsevier,2014

[11] Matlab documentation of Backscatter radar signals from pedestrian and bicyclist ©.

[12] Mathworks toolbox of Continuous Wavelet Transform and Scale-Based Analysis ©

[13] Matlab documentation of Introduction to Microdoppler effects ©

11. Appendix

A. Matlab Simulation Packages on Google Drive

The simulations we implement in this project can be found in the following google drive link. Please use ASU account to get access in it.

https://drive.google.com/drive/folders/1Md7Q4Hj40CP1TcCRsdrFrx6_odHmNkRi?usp=sharing

B. Matlab Code - DCM

Car noise cancellation using matched filtering and DCM is done using the below code. The code for the scenario was taken from [13]. Only the DCM part, which is our contribution has been shown here

```
%waveform for matched filter
waveform = phased.LinearFMWaveform('PulseWidth',1e-4,'PRF',5e3,...
    'SampleRate',1e6,'OutputFormat','Pulses','NumPulses',1,...
    'SweepBandwidth',1e5);
%function to get matched filter using taylor spectrum window
wav1 = getMatchedFilter(waveform);
taylorfilter = phased.MatchedFilter('Coefficients',wav1,...
    'SpectrumWindow','Kaiser');
%delay radar return signal
delay_xd=delayseq(xd,2000);
%multiply delayed signal with original
y=xd.*delay_xd;
%filter the output using designed matched filter
y_out=taylorfilter(y);

%plot TF response
figure
spectrogram(sum(y_out),kaiser(128,10),120,256,1/Tsamp,'centered','yaxis');
clim = get(gca,'CLim');
set(gca,'CLim',clim(2)+[-50 0])
```

We simulated the results shown in[2] and the the following is the code for that.

```
clc
clear all
close all
```

```
a=10; v=15;
f0=3e9;

A=[0.2, 0.4, 0.5];
sig=[1, 1, 1];
theta=[30*pi/180, 45*pi/180, 60*pi/180];

Tc=1;
fs1=4000;
tm=0:1/fs1:Tc-1/fs1;
c=3e8;
fc=5;

for j=1:length(tm)
    s(j)=0;
    for i=1:3
        s(j)=s(j)+exp((-1i*4*pi*f0/c)*(v*tm(j)+a*tm(j)*tm(j)/2))*(sig(i)*exp((-1i
       *4*pi*f0/c)*A(i)*sin(2*pi*fc*tm(j)+theta(i))));
    end
end

figure(1)
spectrogram(s,kaiser(128,10),120,256,fs1,'reassigned','yaxis');

tau=1/fc;
for j=1:length(tm)
    Phat(j)=exp((-1i*4*pi*f0/c)*(a*tau*tm(j)+v*tau-(a*tau*tau/2)));
    f=1/tm(j);
    Phat_dft(j)=exp((-1i*4*pi*f0/c)*(v*tau-(a*tau*tau/2)))*sinc(f+(2*f0*a*tau/c));
end

figure(2)
spectrogram(Phat,kaiser(128,10),120,256,fs1,'reassigned','yaxis');
freq = -fs1/2: fs1/(length(Phat)-1): fs1/2 ;
Phat_fft= abs(fftshift(fft(Phat)))/length(Phat);
```

```
figure(3)
plot(freq,Phat_fft);
xlabel('Frequency'); ylabel('Amplitude')
[m,idx]=max(Phat_fft);
fa=freq(idx);
acc=-(fa*c)/(2*f0*tau);

clear Phat
clear Phat_dft

for j=1:length(tm)
    snew(j)=s(j)*exp((1i*4*pi*f0/c)*acc*tm(j)*tm(j)/2);
end

figure(4)
spectrogram(snew,kaiser(128,10),120,256,fs1,'reassigned','yaxis');

S=diag(snew);
```

Matlab

C. Matlab Code - Radar Signal Generator

We generate the radar signals by **main_signal.m**. It is referring to Matlab source code in [1] by modifying and adjusting the parameters to fit the requirements of the project. The code is shown below.

```
%% main_signal.m
% remove all previous actions
clear;
close all;
clc;

addpath("./src/");
addpath("./data/");
addpath("./PedestrianAndBicyclistClassificationUsingDeepLearningExample/");
addpath("PedBicCarData");

%%

% load(fullfile("data","LabelNoCar.mat"))

%% Generate 1000 samples
% for jj = 1:length(testLabelNoCar)
for jj = 1:1000
%     labels = split(string(testLabelNoCar(jj)), "+"); % changed here
    labels = split(string(testLabelCarNoise(jj)), "+"); % changed here
    numPed = 0;
    numBic = 0;
    numCar = 0; % numCar == 1 with car noise; else, numCar == 0
    for ii = 1:length(labels)
        if labels(ii) == "ped"
            numPed = 1;
        elseif labels(ii) == "bic"
            numBic = 1;
        else
            numCar = 1;
        end
```

```
    end
    [xPedRec,xBicRec,xCarRec,Tsamp] = Generator(numPed, numBic, numCar);

%     xSigRec = xPedRec + xBicRec;    % no car signals
    xSigRec = xPedRec + xBicRec + xCarRec;

    if jj == 1
        xSig = xSigRec;
        if (numPed==1) && (numBic==1)
            x = "ped+bic";
        else
            if numPed == 1
                x = "ped";
            else
                x = "bic";
            end
        end
    else
        xSig = cat(3, xSig, xSigRec);
        if (numPed==1) && (numBic==1)
            x(end+1) = "ped+bic";
        else
            if numPed == 1
                x(end+1) = "ped" ;
            else
                x(end+1) = "bic";
            end
        end
    end
    fprintf("Loop number %d completed\n", jj);
end

save("generated data_2.mat","xSig","x","-v7.3");
```

Generator.m function is called in **main_signal.m** to generate radar signals. The radar parameters are discussed in Table 1

```
function [xPedRecF,xBicRecF,xCarRecF,Tsamp] = Generator(numPed,numBic,numCar)
    % This function is referring to Matlab helperBackScatterSignals function

    %% radar parameters
    % radar operating related parameters
    bw = 250e6; % waveform bandwidth
    fs = bw*2; % waveform sampling frequency - fast time sampling frequency
    c = 3e8;
    fc = 24e9; % waveform carrier frequency
    tm = 1e-6; % waveform repetition time

    % radar plat parameters
    radar_pos = [0;0;0]; % radar position
    radar_vel = [0;0;0]; % radar velocity
    radarplat = phased.Platform('InitialPosition',radar_pos,'Velocity',radar_vel
    ,...
        'OrientationAxesOutputPort',true);

    % radar waveform
    wav = phased.FMCWWaveform('SampleRate',fs,'SweepTime',tm,'SweepBandwidth',bw);
    tx = phased.Transmitter('PeakPower',1,'Gain',25);
    txWave = tx(wav());

    % Simulation setup
    maxbicSpeed = 10; % maximum speed of the Bicyclist
    lambda = c/fc;  % carrier wavelength
    oversamplingFactor = 1.11; % oversampling factor in frequency domain
    fmax = (2*maxbicSpeed/lambda)*oversamplingFactor; % calculate the sampling
    frequency based on the maximum speed of a bicyclist
    Tsamp = 1/(2*fmax); % slow time sampling interval
    timeDuration = 2; % simulation time duration in seconds
    npulse = floor(timeDuration/Tsamp); % number of pulses
    [posr,velr,~] = radarplat(Tsamp); % radar position and velocity



    %%
```

```
    % for jj = 1:length(testLabelNoCar)
    for jj = 1:1
        % signal initialization
        xPedRec = complex(zeros(round(fs*tm),npulse));
        xBicRec = complex(zeros(round(fs*tm),npulse));
        xCarRec = complex(zeros(round(fs*tm),npulse));

        xPedRecF = complex(zeros(size(xPedRec,1),size(xPedRec,2))); % now is 2D
        xBicRecF = complex(zeros(size(xBicRec,1),size(xBicRec,2)));
        xCarRecF = complex(zeros(size(xCarRec,1),size(xCarRec,2)));

        % area of interest
        yLocLimit = [-10,10];
        xLocLimit = [5,45];

        % Generation of pedestrian signals
        if numPed == 1
            % Pedestrian parameters
            ped_pos = [xLocLimit(1) + (xLocLimit(2)-xLocLimit(1))*rand;
                yLocLimit(1) + (yLocLimit(2)-yLocLimit(1))*rand;
                0]; % initial location
            ped_height = 1.5 + (2-1.5)*rand; % height in U[1.5,2] meters
            ped_speed = rand*ped_height*1.4; % speed in U[0,1.4*height] m/s
            ped_heading = -180 + 360*rand; % heading in U[-180,180] degrees

            pedestrian = backscatterPedestrian('InitialPosition',ped_pos,...
                'InitialHeading',ped_heading,'PropagationSpeed',c,...
                'OperatingFrequency',fc,'Height',ped_height,...
                'WalkingSpeed',ped_speed); % pedestrian object
            channel_ped = phased.FreeSpace('PropagationSpeed',c,'
OperatingFrequency',fc,...
                'TwoWayPropagation',true,'SampleRate',fs); % channel

            for m = 1:npulse
                [posPed,velPed,axPed] = move(pedestrian,Tsamp,ped_heading); %
pedestrian moves
```

```
                [~,angrPed] = rangeangle(posr,posPed,axPed); % propagation path direction
                xPedCh = channel_ped(repmat(txWave,1,size(posPed,2)),posr,posPed,velr,velPed); % simulate channel
                xPed = reflect(pedestrian,xPedCh,angrPed); % signal reflection
                xPedRec(:,m) = xPed; % received m-th pulse
            end

            xPedRecF(:,:) = conj(dechirp(xPedRec,txWave)); % convert received signals to baseband

        end

        % Generation of bicyclist signals
        if numBic == 1
            % Bicyclist parameters
            bic_pos = [xLocLimit(1) + (xLocLimit(2)-xLocLimit(1))*rand;
                yLocLimit(1) + (yLocLimit(2)-yLocLimit(1))*rand;
                0]; % initial location
            bicyclistSpeed = 1 + (10-1)*rand; % Speed in U[1,10] meters
            bic_heading = -180 + 360*rand; % heading in U[-180,180] degrees
            GearTransmissionRatio = 0.5 + (6-0.5)*rand; % in U[0.5,6]
            NumWheelSpokes = 36; % number of spokes
            Coast = rand<0.5; % 50% chance to be pedaling or coasting
            bicyclist = backscatterBicyclist('InitialPosition',bic_pos,'InitialHeading',bic_heading,...
                'Speed',bicyclistSpeed,'PropagationSpeed',c,'OperatingFrequency',fc,...
                'GearTransmissionRatio',GearTransmissionRatio,'NumWheelSpokes',NumWheelSpokes,...
                'Coast',Coast); % bicyclist object
            channel_bic = phased.FreeSpace('PropagationSpeed',c,'OperatingFrequency',fc,...
                'TwoWayPropagation',true,'SampleRate',fs);

            for m = 1:npulse
```

```
            [posBic,velBic,axBic] = move(bicyclist,Tsamp,bic_heading); % pedestrian moves
            [~,angrBic] = rangeangle(posr,posBic,axBic); % propagation path direction
            xBicCh = channel_bic(repmat(txWave,1,size(posBic,2)),posr,posBic,velr,velBic); % simulate channel
            xBic = reflect(bicyclist,xBicCh,angrBic); % signal reflection
            xBicRec(:,m) = xBic; % received m-th pulse
        end

        xBicRecF(:,:) = conj(dechirp(xBicRec,txWave)); % convert received signals to baseband
    end

    maxCarSpeed = 10;
    % Generation of car signals
    if numCar == 1
        % Car parameters
        car_pos = [xLocLimit(1) + (xLocLimit(2)-xLocLimit(1))*rand;
            yLocLimit(1) + (yLocLimit(2)-yLocLimit(1))*rand;
            0]; % initial location
        car_vel = [-maxCarSpeed+(maxCarSpeed+maxCarSpeed)*rand;
            -maxCarSpeed+(maxCarSpeed+maxCarSpeed)*rand;
            0]; % car velocity
        car = phased.Platform('InitialPosition',car_pos,'Velocity',car_vel,...
            'OrientationAxesOutputPort',true); % car object
        carTgt = phased.RadarTarget('PropagationSpeed',c,'OperatingFrequency',fc,'MeanRCS',10);
        chan_car = phased.FreeSpace('PropagationSpeed',c,'OperatingFrequency',fc,...
            'TwoWayPropagation',true,'SampleRate',fs);

        for m = 1:npulse
            [posCar,velCar,~] = car(Tsamp); % pedestrian moves
            xCarCh = chan_car(repmat(txWave,1,size(posCar,2)),posr,posCar,velr,velCar); % simulate channel
```

```
                xCar = carTgt(xCarCh); % detection
                xCarRec(:,m) = xCar; % received m-th pulse
            end

            xCarRecF(:,:) = conj(dechirp(xCarRec,txWave)); % convert received signals to baseband
        end
    end
end
```

Matlab

D. Matlab Code - Micro-Doppler Signatures

We also generate the MD signatures to figure out the differences between different TF representations. **main2_STFT.m** & **main2_CWT.m** are the source codes referring to [1] as well. We changed some parameters to generate the corresponding MD signatures. The **helperPreProcess** function is source code provided by Matlab. The codes are shown below.

D.1. main2_STFT.m

```
%% main2_STFT.m
% remove all previous actions
clear;
close all;
clc;

addpath("./src/");

%% Synthetic data generation by simulation
% numPed = 1;
% numBic = 1;
% numCar = 1;
% [xPedRec, xBicRec, xCarRec, Tsamp] = helperBackScatterSignals(numPed, numBic,
    numCar);

%% Load Synthetic data from generated radar signals

load(fullfile("generated data_1.mat"))

%% transpose the label array from row vector to column vector
x = x';

%% STFT of radar to generate the MD signature
M = [20,200,2000];  % FFT window length
w = ["Rectangular", "Triangular", "Bartlett", "Blackman", "Chebyshev", "Gaussian",
     "Hamming", "Kaiser", "Hann"];
Tsamp = 2.815315315315315e-04;  % slow time sampling interval

```

```

for ii = 1:length(M)
% for ii = 3
    for jj = 1:length(w)
%       for jj = 8
        for tt = 1:size(xSig,3)
            [status, msg, msgID] = mkdir(fullfile("fig","STFT",num2str(M(ii)),w(jj)));
            [Sig, T, F] = MDSign_STFT(xSig(:,:,tt), Tsamp, M(ii), w(jj));

            % Concatenation
            if tt == 1
                SigCat = Sig;
            else
                SigCat = cat(3, SigCat, Sig);
            end

            % Plot the realization of objects
            figure(1)
            % subplot(3,1,1)
            imagesc(T,F,Sig(:,:,1))    % SPed(:,:,1) -> (row, col, t) for radar
            xlabel("Time (s)")
            ylabel("Frequency (Hz)")
            title("Spectrogram " + x(tt) + " length=" + num2str(M(ii)) + " ", w(jj))
            axis square xy              % plot box aspect ratio = [1 1 1]

            saveas(gcf,"./fig/STFT/"+ M(ii) + "/" + w(jj) + "/" + num2str(tt) + ", M=" + num2str(M(ii)) + ", " + w(jj) + ".png")


            % Configure Gaussian noise level at the receiver
            rx = phased.ReceiverPreamp("Gain", 25, "NoiseFigure", 10);

            xRadarRec = complex(zeros(size(xSig(:,:,tt))));

```

```
            xRadarRec(:,:) = rx(xSig(:,:,tt));


            % obtain micro-Doppler signatures of the received signal by using the STFT
            [S,~,~] = MDSign_STFT(xRadarRec, Tsamp, M(ii), w(jj));

            % Concatenation
            if tt == 1
                SCat = S;
            else
                SCat = cat(3, SCat, S);
            end

            figure(2)
            imagesc(T,F,S(:,:,1));  % plot the realization
            axis xy
            xlabel("Time (s)")
            ylabel("Frequency (Hz)")
            title("Spectrogram with noise " + x(tt) + " length=" + num2str(M(ii)) + " ", w(jj))

            saveas(gcf,"./fig/STFT/"+ M(ii) + "/" + w(jj) + "/" + num2str(tt) + ", M=" + num2str(M(ii)) + ", " + w(jj) + "@ receiver.png")
        end
        close all;
%         filename = "LabelNoCar," + w(jj)+ "," + num2str(M(ii)) + ".mat";
        filename = "LabelWithCar," + w(jj)+ "," + num2str(M(ii)) + ".mat";
        save(fullfile("data",filename),"x","T","F","SigCat","SCat","Tsamp");
    end
end
```

D.2. main2_CWT.m

```
%% main2_CWT.m
% remove all previous actions
clear;
close all;
```

```
clc;

addpath("./src/");

%% Synthetic data generation by simulation
% numPed = 1;
% numBic = 1;
% numCar = 1;
% [xPedRec, xBicRec, xCarRec, Tsamp] = helperBackScatterSignals(numPed, numBic,
    numCar);

%% Load Synthetic data from generated radar signals

load(fullfile("generated data_1.mat"))

%% transpose the label array from row vector to column vector
x = x';

%%  CWT of radar to genearte the MD signature


Tsamp = 2.815315315315315e-04;  % sampling duration
fsamp = 1/Tsamp;                % sampling frequency

for tt = 1:size(xSig,3)
% for tt = 1:1  % for demo
    [status, msg, msgID] = mkdir(fullfile("fig","CWT"));
    [SigP, F] = MDSign_CWT(xSig(:,:,tt), fsamp);

    if tt == 1
        SigCat = SigP;
    else
        SigCat = cat(3, SigCat, SigP);
    end

    sigLen = numel(xSig(:,:,1));
```

```
    T = (0:sigLen-1)/fsamp;
    T = (T*2)/1000; % normalized in 2 secs for x axis

    % plot
    figure(1)
    imagesc(T,F,SigP);
    xlabel('Time (s)');
    ylabel('Frequency (Hz)')
    title('Scalogram ' + x(tt));
    axis square xy

    saveas(gcf,"./fig/CWT/"+ num2str(tt) + ".png")

    % Configure Gaussian noise level at the receiver
    rx = phased.ReceiverPreamp("Gain", 25, "NoiseFigure", 10);

    xRadarRec = complex(zeros(size(xSig(:,:,tt))));

    xRadarRec(:,:) = rx(xSig(:,:,tt));

    % obtain micro-Doppler signatures of the received signal by using the
    % CWT
    [S,~] = MDSign_CWT(xRadarRec, fsamp);

    % Concatenation
    if tt == 1
        SCat = S;
    else
        SCat = cat(3, SCat, S);
    end

    % plot
    figure(2)
    imagesc(T,F,SigP);
    xlabel('Time (s)');
    ylabel('Frequency (Hz)')
```

```
    title('Scalogram ' + x(tt));
    axis square xy

    saveas(gcf,"./fig/CWT/"+ num2str(tt) + "@ receiver.png")

end
close all;
%%

filename = "LabelNoCar,CWT";
save(fullfile("data",filename),"x","T","F","SigCat","SCat","Tsamp","-v7.3");
```

Matlab

E. Matlab Code - CNN

main4_CNN.m is the source code to train and test the datasets. Here we refer the source code provided by Matlab [1], and adjust the CNN structures for this project. The code is shown here.

```
%% main4_CNN.m
% remove all previous actions
clear;
close all;
clc;

addpath("./src/");
addpath("./data/");

%% Classify Signatures without Car Noise
M = 200;
w = ["Rectangular", "Triangular", "Bartlett", "Blackman", "Chebyshev", "Gaussian",
    "Hamming", "Kaiser", "Hann"];

for tt = 1:length(w)
% for tt = 1:1    % for demo

    % load datasets
    filename = "LabelNoCar," + w(tt) + ",200.mat";
    load(fullfile("data", filename))    % load datasets

    % Reshape and separate the datasets
    % 80% for training, 20% for testing
    data_size = size(SCat,3);
    train_size = data_size * 0.8;

    % trained_data = SCat(:,:,1:train_size);
    trained_label = categorical(x(1:train_size));
    % tested_data = SCat(:,:,train_size+1:data_size);
    tested_label = categorical(x(train_size+1:data_size));


```

```
    for ii = 1:train_size
        if ii == 1
            trained_data = SCat(:,:,ii,1);
        else
            trained_data = cat(4,trained_data,SCat(:,:,ii,1));
        end
    end

    for ii = train_size+1:data_size
        if ii == train_size+1
            tested_data = SCat(:,:,ii,1);
        else
            tested_data = cat(4,tested_data,SCat(:,:,ii,1));
        end
    end

    % Network Architecture
    layers = [
        imageInputLayer([size(SCat,1),size(SCat,2),1], "Normalization","none")

        convolution2dLayer(10,16,"Padding","same")
        batchNormalizationLayer
        sigmoidLayer
        maxPooling2dLayer(10, "Stride", 2)

        convolution2dLayer(5,32,"Padding","same")
        batchNormalizationLayer
        sigmoidLayer
        maxPooling2dLayer(10, "Stride", 2)

        convolution2dLayer(5,32,"Padding","same")
        batchNormalizationLayer
        sigmoidLayer
        maxPooling2dLayer(10, "Stride", 2)

        convolution2dLayer(5,32,"Padding","same")
```

```
        batchNormalizationLayer
        sigmoidLayer
        maxPooling2dLayer(5, "Stride", 2)

        convolution2dLayer(5,32,"Padding","same")
        batchNormalizationLayer
        sigmoidLayer
        averagePooling2dLayer(2, "Stride", 2)

        fullyConnectedLayer(3)
        softmaxLayer

        classificationLayer
    ];

    % Specifies optimization solver
    options = trainingOptions("adam", ...
        "ExecutionEnvironment", "GPU", ...
        "MiniBatchSize",30, ...
        "MaxEpochs",60, ...
        "InitialLearnRate",1e-2, ...
        "LearnRateSchedule","piecewise", ...
        "LearnRateDropFactor",0.1, ...
        "LearnRateDropPeriod",10, ...
        "Shuffle","every-epoch", ...
        "Verbose",false, ...
        "Plots","training-progress");

    % Train the CNN
    trainedNetNoCar = trainNetwork(trained_data,trained_label,layers,options);

    saveas(gcf,fullfile("fig", "train_progress", "train_progress_" + w(tt) + "2.
png"))
    % Classification
    predTestLabel = classify(trainedNetNoCar,tested_data);
    testAccuracy = mean(predTestLabel == tested_label)
```

```matlab

    figure
    confusionchart(tested_label,predTestLabel)
    title("STFT " + w(tt))
    saveas(gcf, "test_"+w(tt)+"2.png")

    save(fullfile("saved_weights","STFT_"+ w(tt)+"2.mat"),"trainedNetNoCar","testAccuracy")
    close all;
end

%% batch size = 8, epoch = 100

% Rectangular
% testAccuracy =    0.9150
%
% Triangular
% testAccuracy =    0.9550
%
% Bartlett
% testAccuracy =    0.9300
%
% Blackman
% testAccuracy =    0.9300
%
% Chebyshev
% testAccuracy =    0.9150
%
% Gaussian
% testAccuracy =    0.9400
%
% Hamming
% testAccuracy =    0.9300
%
% Kaiser
% testAccuracy =    0.9200
```

```
%
% Hann
% testAccuracy =    0.9600



%% batch size = 30, epoch = 60

% Rectangular
% testAccuracy =    0.8950
%
% Triangular
% testAccuracy =    0.9350
%
% Bartlett
% testAccuracy =    0.9000
%
% Blackman
% testAccuracy =    0.9350
%
% Chebyshev
% testAccuracy =    0.9000
%
% Gaussian
% testAccuracy =    0.9400
%
% Hamming
% testAccuracy =    0.9200
%
% Kaiser
% testAccuracy =    0.9100
%
% Hann
% testAccuracy =    0.9250
```

Matlab