Asia 2023

Target Classification on the Edge Using mmWave Radar: A Novel Algorithm and Its Real-Time Implementation on TI’s IWRL6432

Muhammet Yanik, Slobodan Jovanovic, Sandeep Rao

Radar Systems and Algorithms R&D, Texas Instruments

What We'll Cover

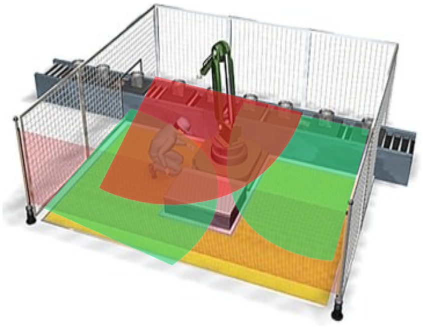
- Introduction
  - What/why is mmWave radar sensing?
  - Why do we need target classification on the edge using mmWave radars?
- The proposed target classification algorithm details. What is the novelty of this work?
- Real-time implementation on TI's 60GHz low-power mmWave radar IWRL6432
- Performance results of the algorithm and the real-time implementation benchmarks
- Summary and conclusions

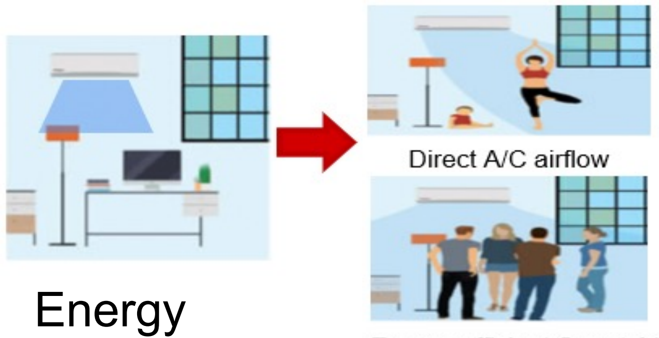
The mmWave Sensing

- The mmWave sensing typically refers to radar applications that operate in the 60GHz and 77GHz frequency bands.

- Why radar now?
  - Technological advances => smaller size, lower cost, higher performance
  - Automotive remains a key market for these sensors
  - New applications: Building surveillance, factory automation, robotics, automotive, in-cabin sensing, and more!

Safety bubbles

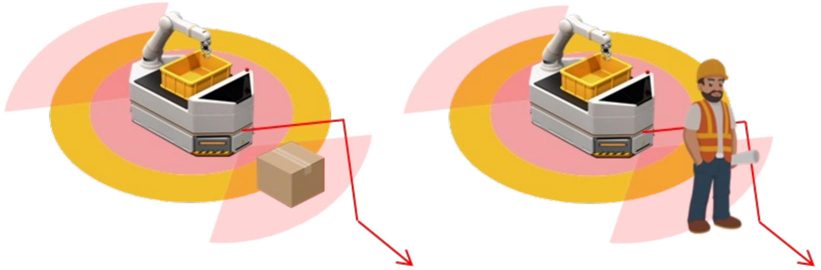




Direct A/C airflow

Energy efficient Smart A/C

Energy efficient units



Mobile robots

In-cabin sensing

Smarter devices

Need for Target Classification

- Traditional applications leverage radar strengths in target detection and tracking.
- Emerging applications require target classification as well:

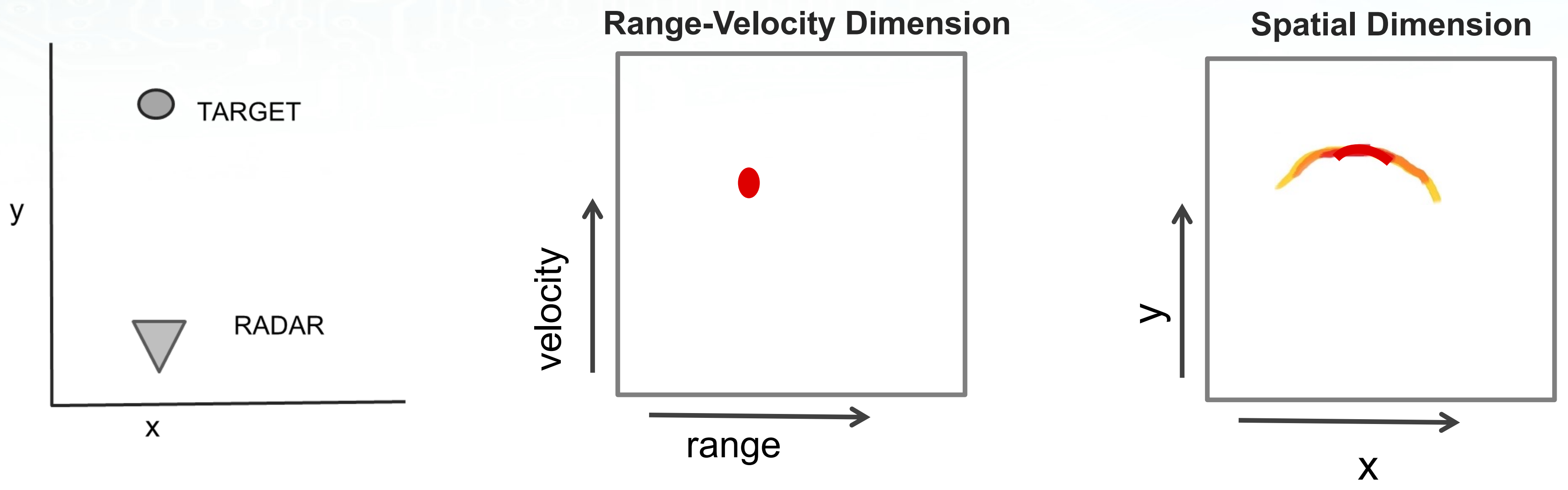
Segment	Use-case	Requirement
Automotive Radar	Road User Classification	High Performance
In-Cabin Sensing	Occupancy Detection and Classification	High Performance, Low Power
Building Automation / Surveillance	Motion Classification (people vs pets, trees etc.)	Low Cost, Low Power, Edge Processing
Elderly Care	Fall Detection, Activity Classification	Low Cost, Low Power, Edge Processing
Gesture Recognition	Automotive Dash Board, Mobile Devices, Household Appliances, Kick-2-Open (Automotive Boot opener)	Low Cost/ Low Power, Edge Processing

Main focus of this work

- Radar classification can leverage the work done in the context of Vision
  - Machine Learning architectures/frameworks can be re-used
- However, the nature of the signal is different
  - Vision: High spatial resolution. None or limited range/velocity resolution
  - Radar: High range/velocity resolution. Limited spatial resolution
  - Hence, intuition on feasibility, performance, and complexity has to be built from the ground up

Resolution in Radar and µ-Doppler Signatures

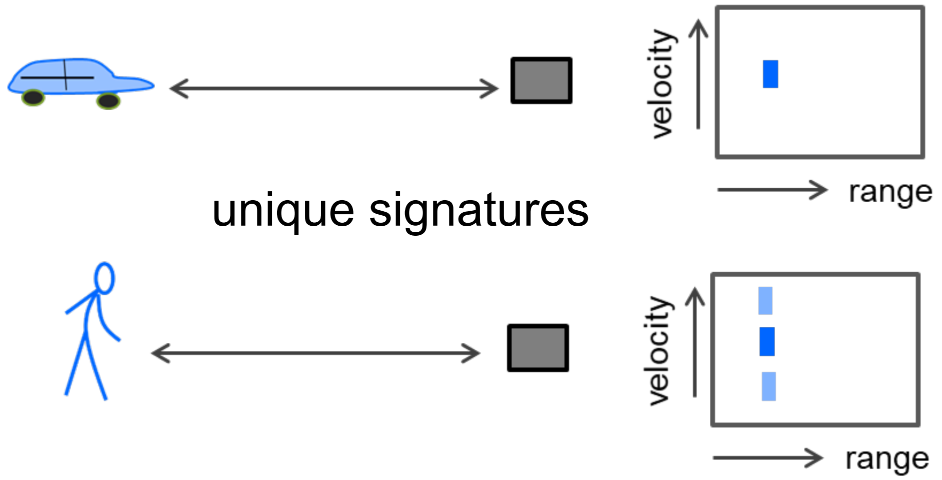
Radar images are very sharp in the range-velocity domain but blurred in the spatial domain



	Radar	Vision
Range resolution	🟩	
Velocity resolution	🟩	
Spatial resolution		🟩
All Weather	🟩	
Privacy	🟩	

- Radars' native good velocity resolution can be used to capture the unique "velocity signature" of moving targets
- Different targets (bike, pedestrian, pet, drone, bird, car, etc.) can have different "signatures."
- Can be used in target classification/ target filtering.

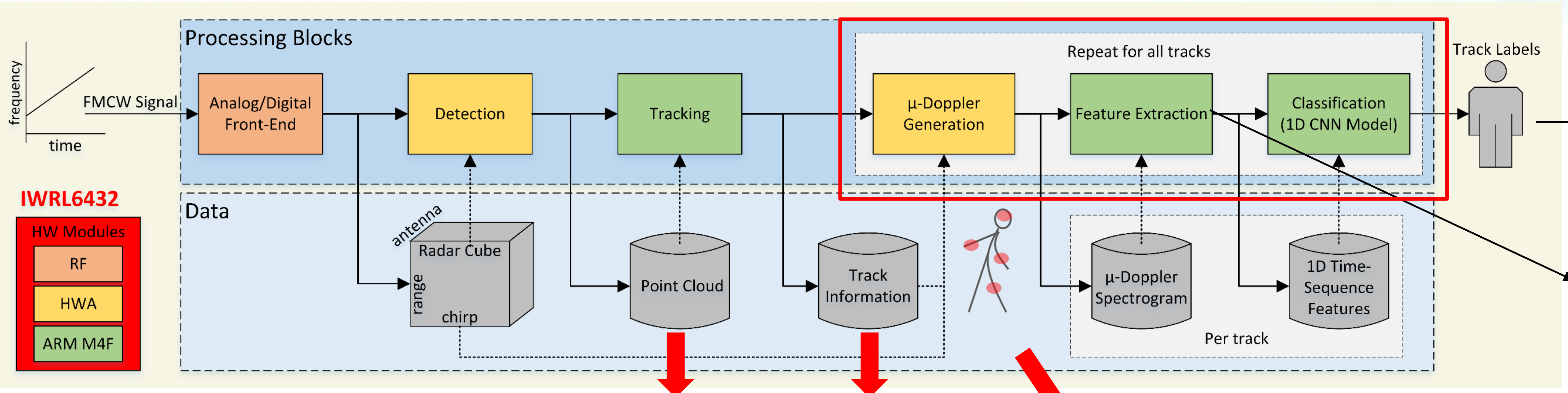
The work in this tutorial relies heavily on these Doppler signatures

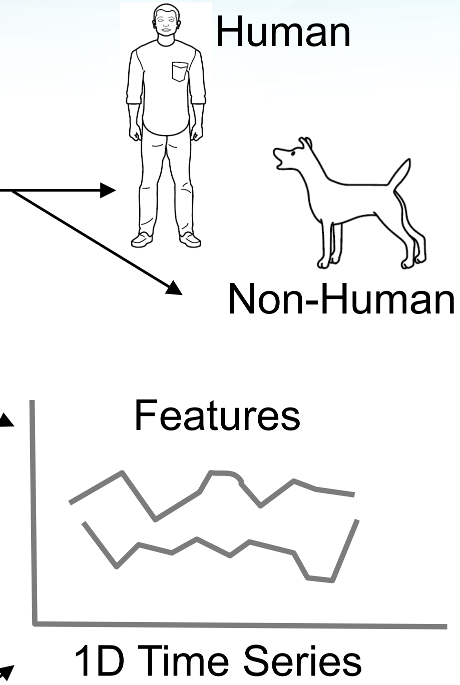


Main challenge: How do we create these signatures for multiple objects concurrently, and how do we process them?

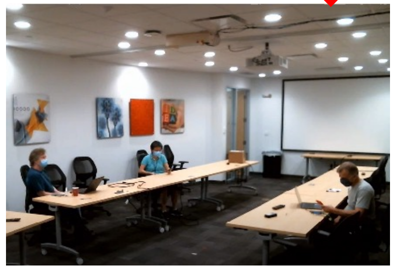
Proposed Solution for Target Classification

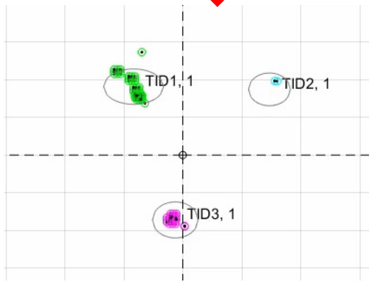
- The end-to-end processing chain to classify multiple target objects: The detection layer generates the point cloud. The tracker layer is run to track multiple objects. Required features are created, and the classifier is run per track.

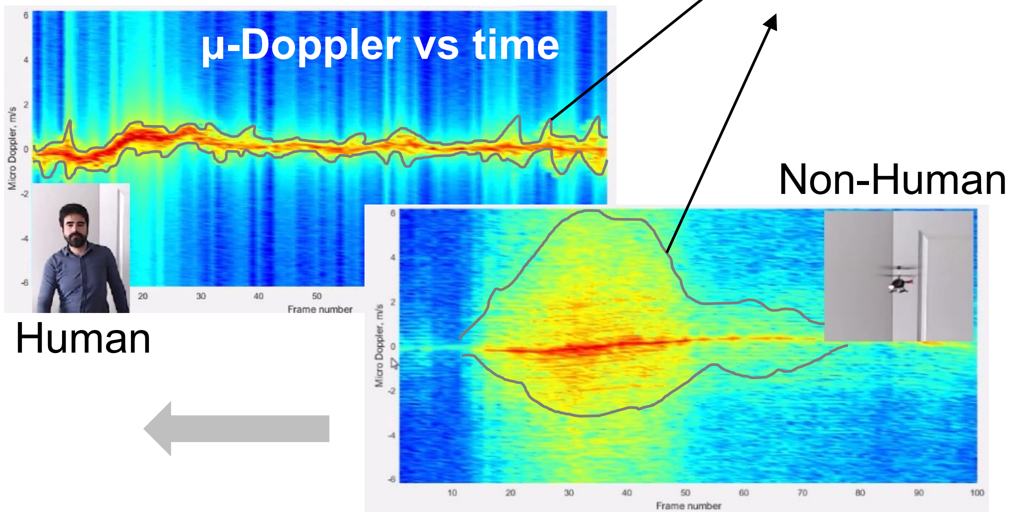




Cluster & Track





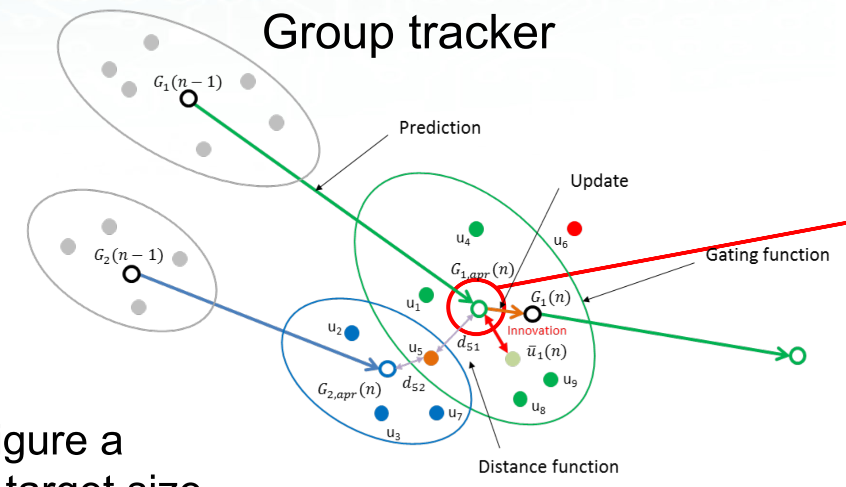


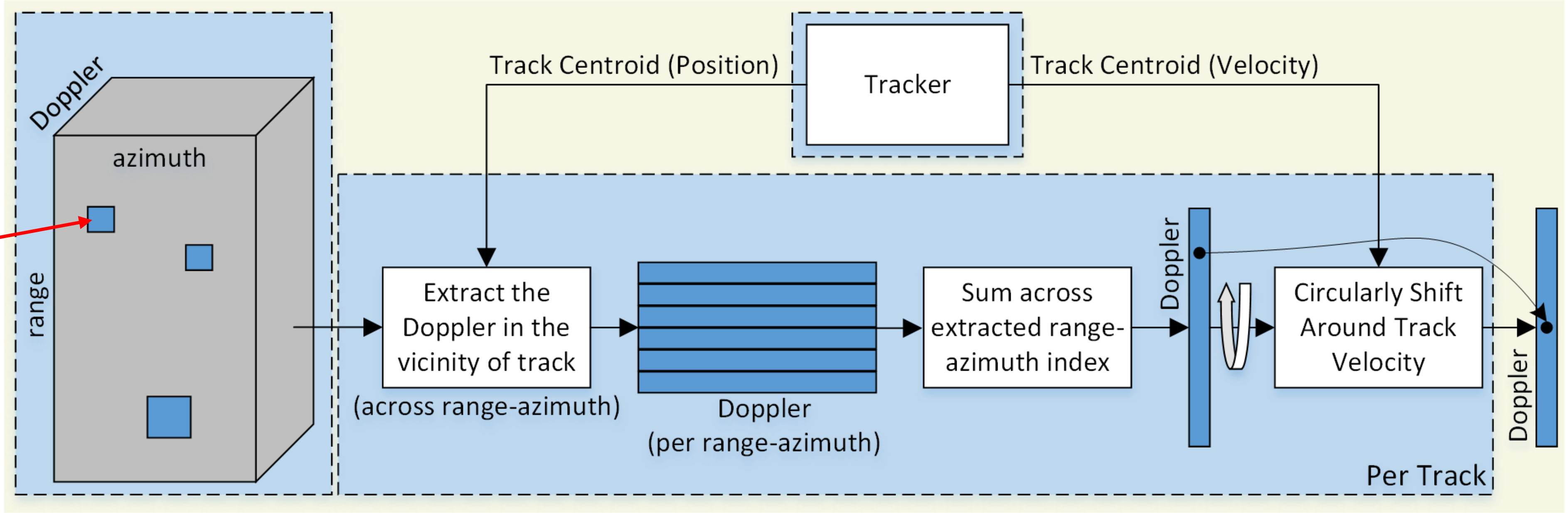
Distinguishable features can be extracted for different target classes. Here: upper and lower envelopes are given as examples.

μ-Doppler Generation

- Tracking and spectrogram generation based on referencing the radar cube by the track centroids are central to our focus.

Group tracker

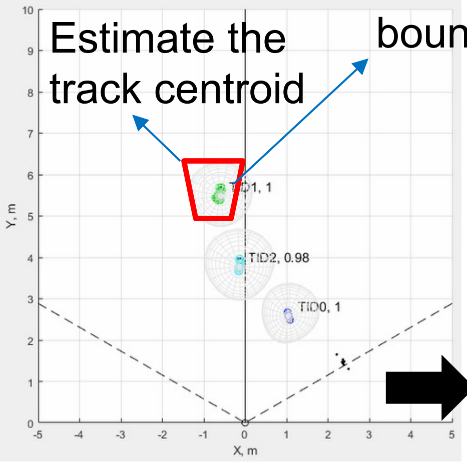




Doppler
azimuth
range
Track Centroid (Position)
Tracker
Track Centroid (Velocity)
Extract the Doppler in the vicinity of track
(across range-azimuth)
Doppler
(per range-azimuth)
Sum across extracted range-azimuth index
Doppler
Circularly Shift Around Track Velocity
Doppler
Per Track

Configure a fixed target size

Main approach:

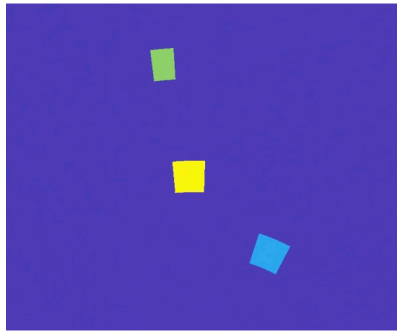


Estimate the track centroid

Define the boundary box

Extract spectrums

Scenario, 3 targets



4 features are then extracted:

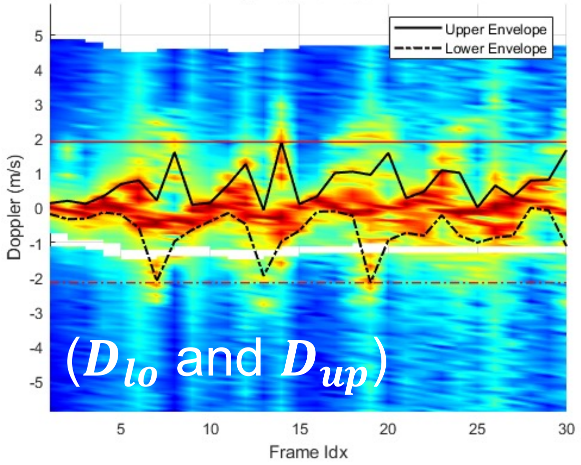
(1) Normalized bandwidth power ($\boldsymbol{P_{BW}}$)

(2) Lower Envelope ($\boldsymbol{D_{lo}}$)

(3) Upper Envelope ($\boldsymbol{D_{up}}$)

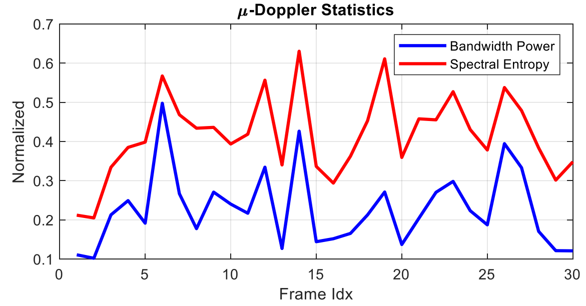
(4) Spectral entropy ($\boldsymbol{H}$)

The spectrum is normalized (between [0 1]) (to eliminate the effect of SNR)



($\boldsymbol{D_{lo}}$ and $\boldsymbol{D_{up}}$)

($\boldsymbol{P_{BW}}$, and $\boldsymbol{H}$)

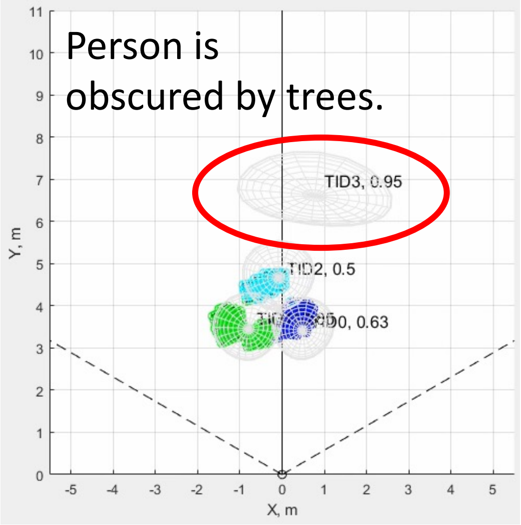


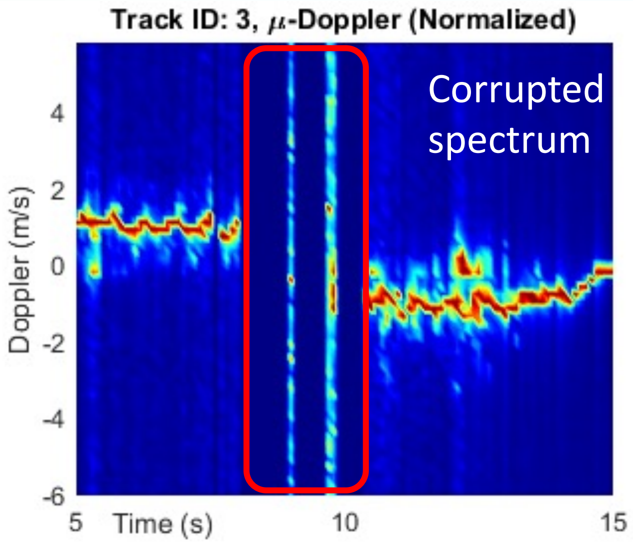
For details: M. E. Yanik and S. Rao, "Radar-Based Multiple Target Classification in Complex Environments Using 1D-CNN Models," *IEEE Radar Conf.*, San Antonio, TX, USA, 2023, pp. 1-6

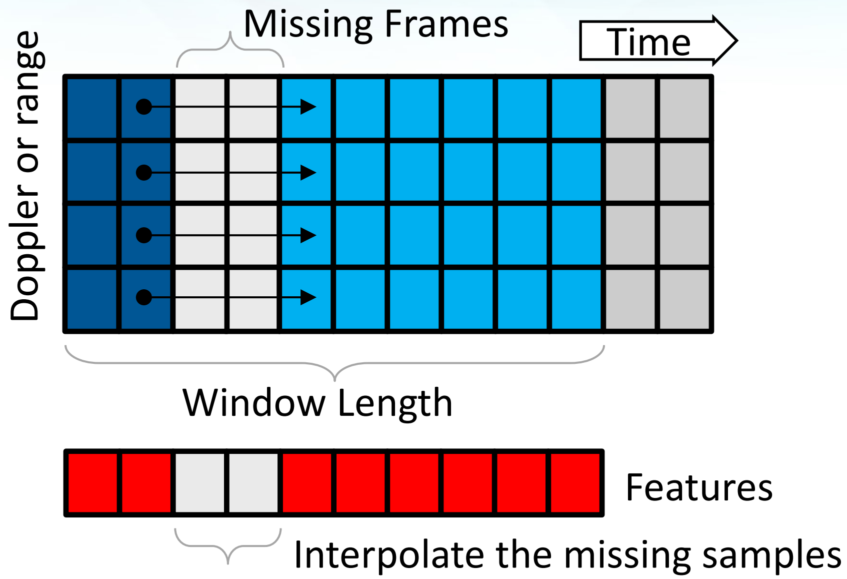
Feature Extraction – Improve Robustness

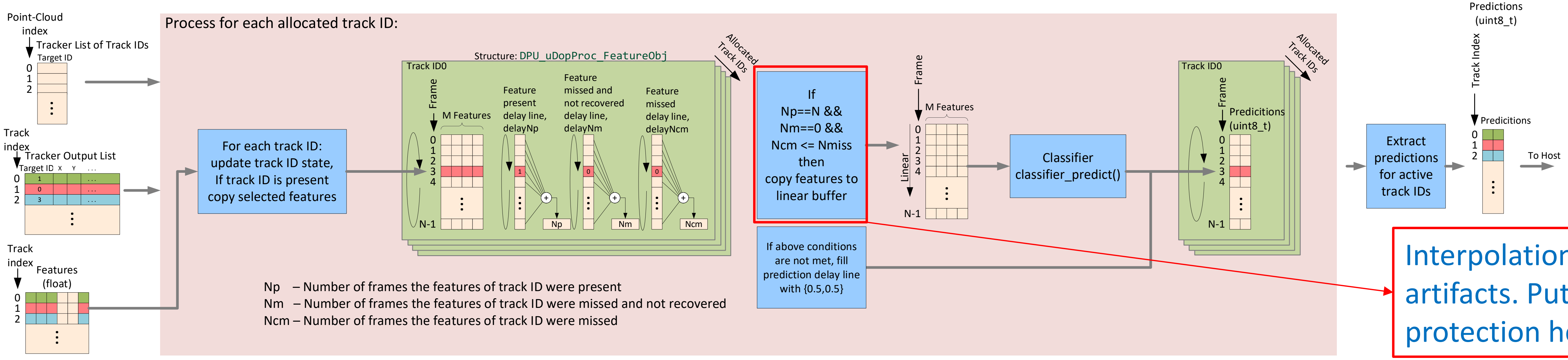
- If a certain track has no associated points at some frames (it may exit from the scene or obscured by another track), the tracker may still continue to track. In such scenarios, we should avoid updating the feature set because the extracted information **will not be reliable**.

An example scenario


Person is obscured by trees.





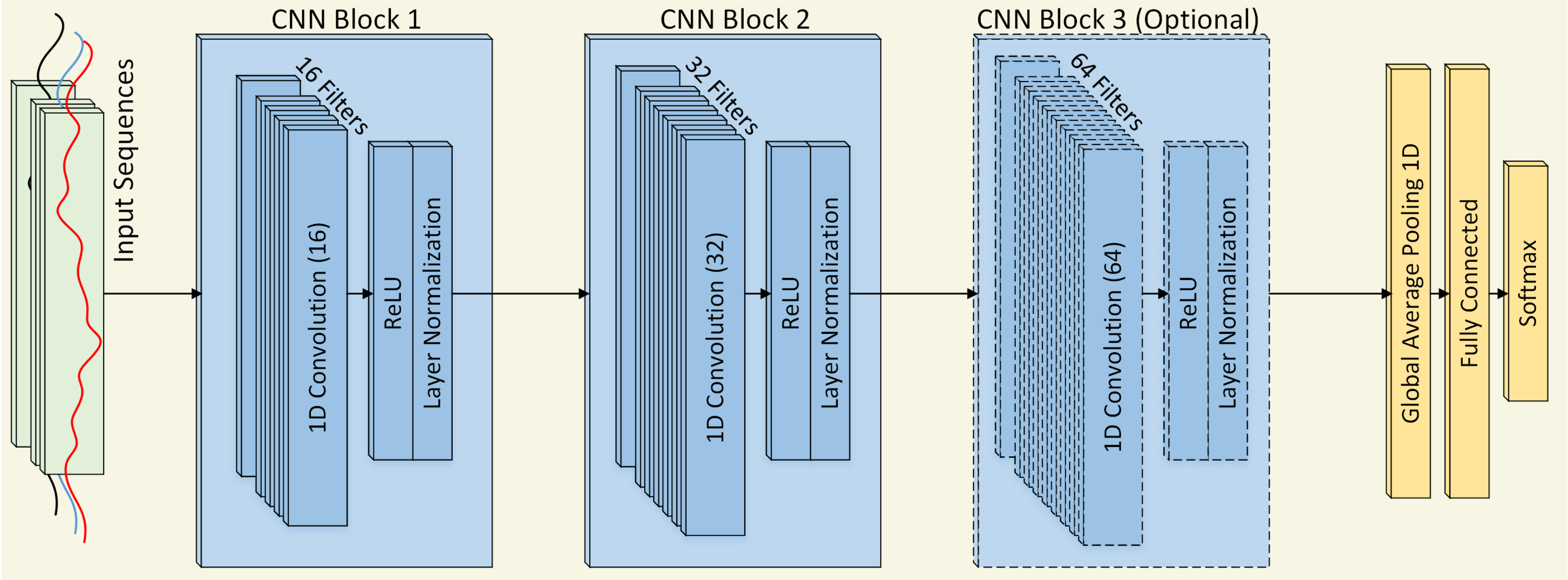

Point-Cloud index
Tracker List of Track IDs
Target ID
Track index
Tracker Output List
Target ID X Y ...
Track index
Features (float)
Process for each allocated track ID:
For each track ID: update track ID state, If track ID is present copy selected features
Structure: DPU_uDopProc_FeatureObj
Track ID0
Frame
M Features
Feature present delay line, delayNp
Feature missed and not recovered delay line, delayNm
Feature missed delay line, delayNcm
Np
Nm
Ncm
N-1
Allocated Track IDs
If Np==N && Nm==0 && Ncm <= Nmiss then copy features to linear buffer
If above conditions are not met, fill prediction delay line with {0.5,0.5}
Linear
Classifier classifier_predict()
Predicitions (uint8_t)
Extract predictions for active track IDs
Predictions (uint8_t)
Track Index
Predicitions
To Host

Np – Number of frames the features of track ID were present
Nm – Number of frames the features of track ID were missed and not recovered
Ncm – Number of frames the features of track ID were missed

Interpolation has some artifacts. Put some protection here!

Classification Model

- In the proposed 1D-CNN architecture:
  - 2 blocks of 1D convolution, ReLU, and layer normalization layers are specified.
  - We create 16 and 32 filters for the first and second convolutional layers (with a filter size of 3), respectively.
  - To reduce the output of the convolutional layers to a single vector, a 1D global average pooling layer is added.
  - To map the output to a vector of probabilities, a fully connected layer is specified with an output size of two (matching the number of classes), followed by a softmax layer.



Processing time of the classifier

- **On ARM M4F core @160MHz**
- **Per prediction call (i.e., per track)**
- **30-frame window**

- 2-layer CNN: **3ms**
- 3-layer CNN: **15ms**

Memory required by the classifier

- **Internal buffer for computations + weights**

- 2-layer CNN: 5.7KB + 7.5KB = **13.2KB**
- 3-layer CNN: 13.4KB + 32.5KB = **45.9KB**

- An optional block of 1D convolution with 64 filters, ReLU, and layer normalization layers (following the first two blocks) is also added to compare the performance of 2 vs. 3 layers CNN models.

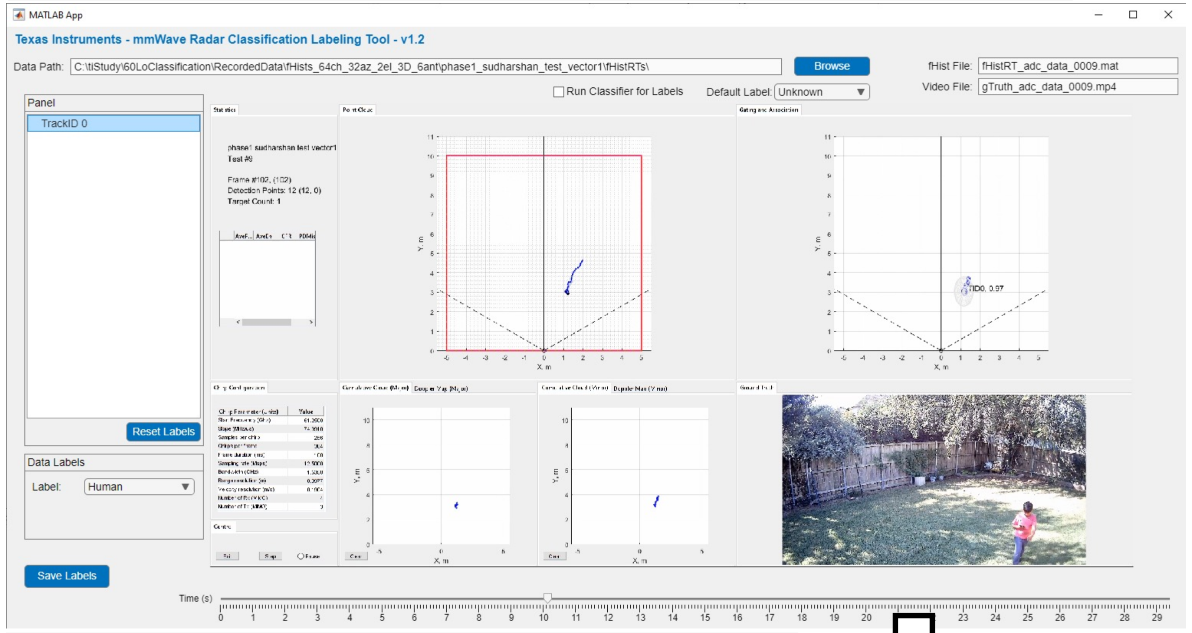
Data Capture and Feature Set Summary

Input the ADC raw data

Point cloud: range, angle, Doppler
Tracker output: position, velocity.

μ-Doppler generation.

Data labeling (with synchronized videos).



MATLAB App

Texas Instruments - mmWave Radar Classification Labeling Tool - v1.2

Data Path: C:\tiStudy\60LoClassification\RecordedData\fHists_64ch_32az_2el_3D_6ant\phase1_sudharshan_test_vector1\fHistRTs\

fHist File: fHistRT_adc_data_0009.mat

Video File: gTruth_adc_data_0009.mp4

□ Run Classifier for Labels

Default Label: Unknown

phase1 sudharshan test vector1
Test #9
Frame #102, (102)
Detection Points: 12 (12, 0)
Target Count: 1

Data Labels

Label: Human

A labeling GUI is built in Matlab.

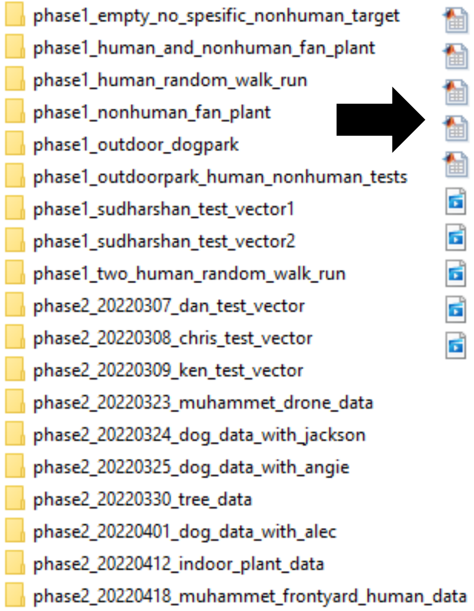
Model/train/evaluate the classifier

Data statistics:

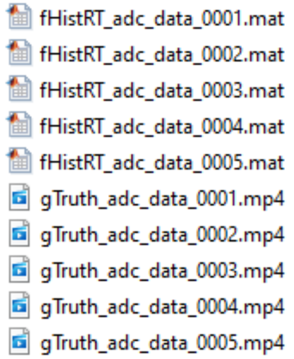
- **310** scenarios in distinct environments.
- Total **125816** frames (**3.5hours**).
- From ~20 different human and numerous non-human targets.

Feature set statistics:

- 2D μ-Doppler maps
- 4 features (time sequence)
- **55079** Observations
  - Block length: 30 frames
  - **25448** human, **29631** non-human

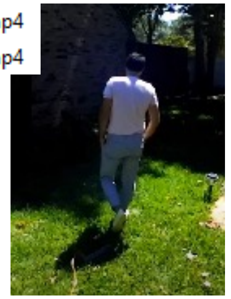


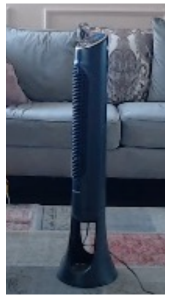
phase1_empty_no_spesific_nonhuman_target
phase1_human_and_nonhuman_fan_plant
phase1_human_random_walk_run
phase1_nonhuman_fan_plant
phase1_outdoor_dogpark
phase1_outdoorpark_human_nonhuman_tests
phase1_sudharshan_test_vector1
phase1_sudharshan_test_vector2
phase1_two_human_random_walk_run
phase2_20220307_dan_test_vector
phase2_20220308_chris_test_vector
phase2_20220309_ken_test_vector
phase2_20220323_muhammet_drone_data
phase2_20220324_dog_data_with_jackson
phase2_20220325_dog_data_with_angie
phase2_20220330_tree_data
phase2_20220401_dog_data_with_alec
phase2_20220412_indoor_plant_data
phase2_20220418_muhammet_frontyard_human_data

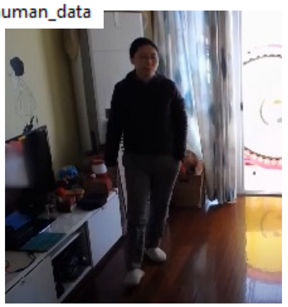


fHistRT_adc_data_0001.mat
fHistRT_adc_data_0002.mat
fHistRT_adc_data_0003.mat
fHistRT_adc_data_0004.mat
fHistRT_adc_data_0005.mat
gTruth_adc_data_0001.mp4
gTruth_adc_data_0002.mp4
gTruth_adc_data_0003.mp4
gTruth_adc_data_0004.mp4
gTruth_adc_data_0005.mp4

Various people and non-human objects (drone, different dogs, fans, trees, plants, etc.) are available in the data.

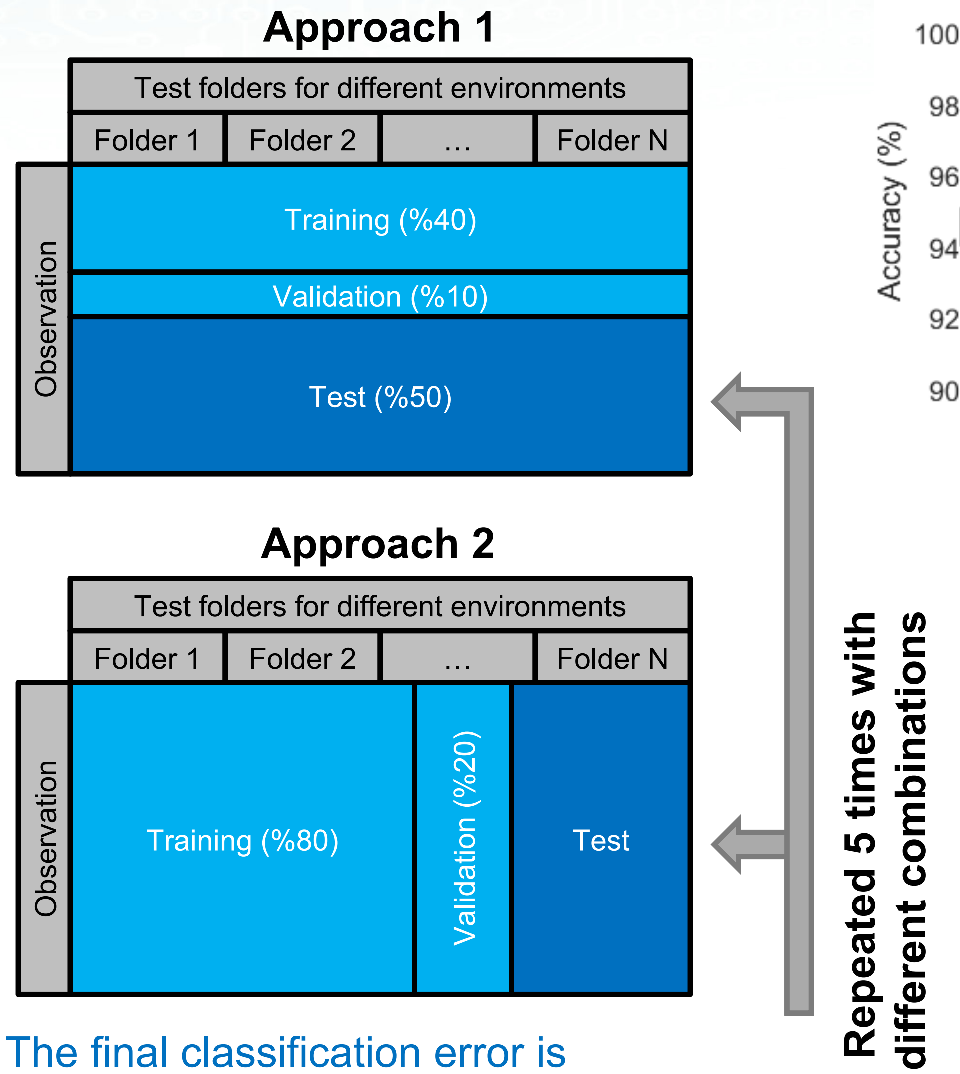






Cross Validation and Performance Results

- The 1st approach uses all the data mixed from different environments. Data split: 40% Train, 10% Validation, **50% Test.**
- The 2nd approach **reserves some folders completely for testing**. The remaining data is used for training + validation



Approach 1

Test folders for different environments

Folder 1 | Folder 2 | … | Folder N

Observation

Training (%40)

Validation (%10)

Test (%50)

Approach 2

Test folders for different environments

Folder 1 | Folder 2 | … | Folder N

Observation

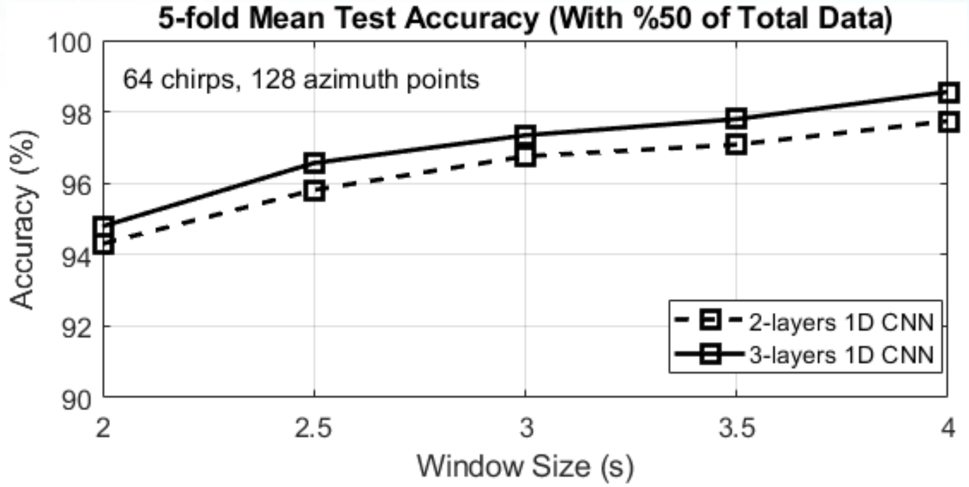
Training (%80)

Validation (%20)

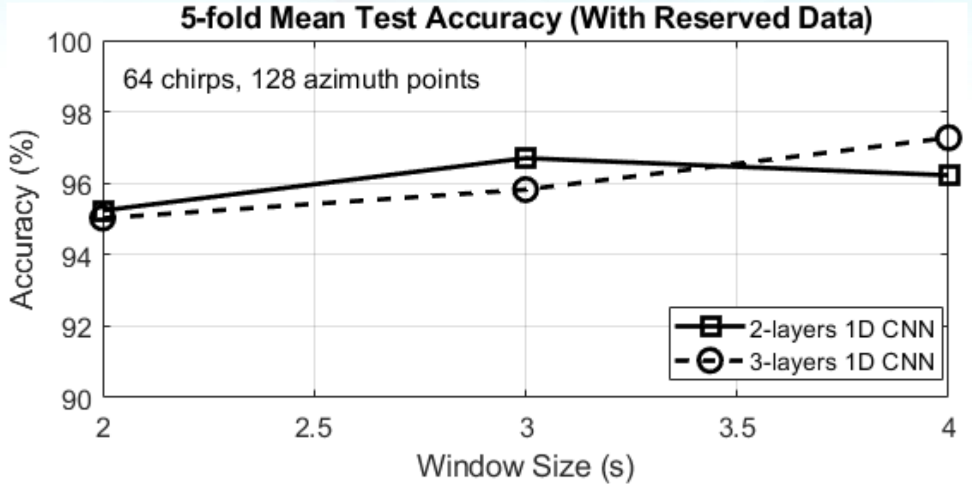
Test

Repeated 5 times with different combinations

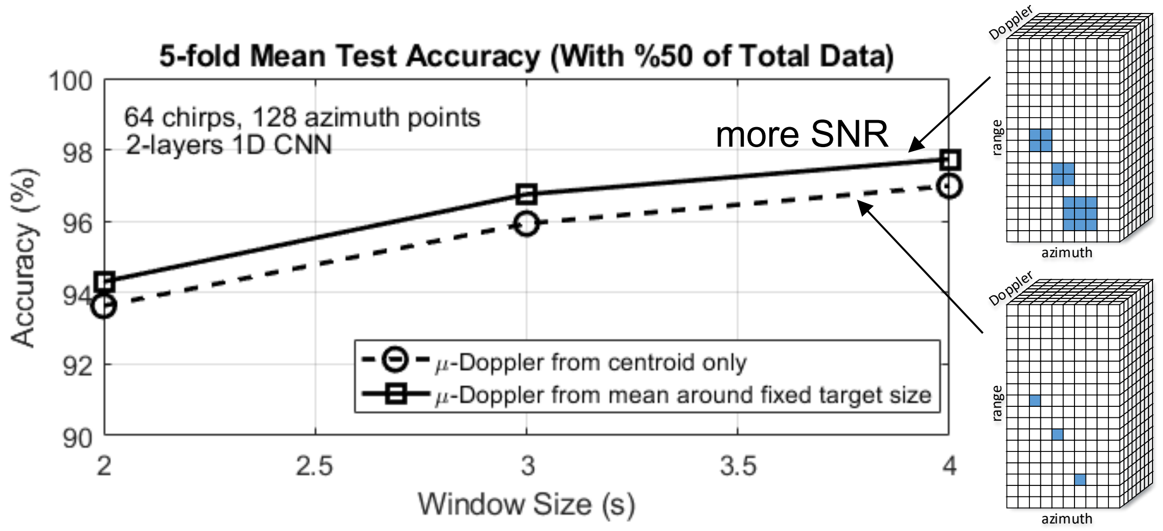
The final classification error is calculated with 5-fold cross-validation.



Approach 1 (All the data is mixed)



Approach 2 (Reserved folders for test)



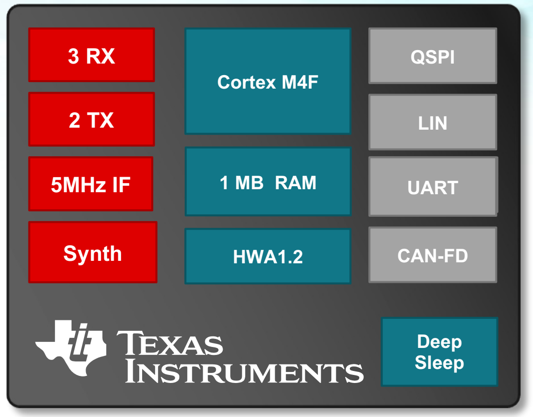
Mean around 0.5m x 0.5m target size)

Centroid only

Summary: ~97% mean classification accuracy with only 3sec latency (2-layers 1D-CNN)

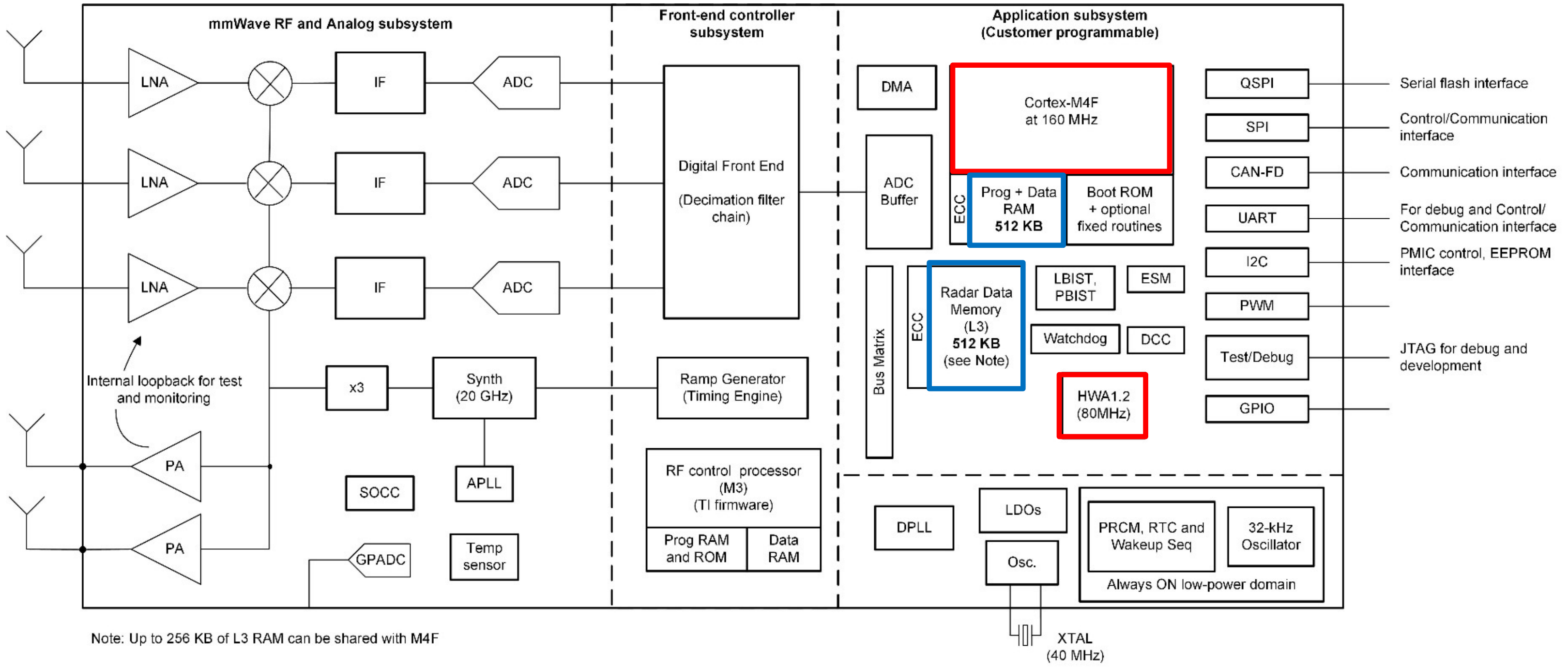
Enable Edge AI – IWRL6432 Overview

IWRL6432: Cost and power-optimized radar on a chip with analog (RF, IF), digital front-end, and processing (M4F+Accelerator) capability in one device.



The target classification algorithm discussed in this work is implemented on this tiny device to create an edge AI solution.

- Integrated transceivers : 3 Rx and 2 Tx.
- Integrated frequency synthesizer: 57GHz to 64GHz
- Processing:
  - ARM Cortex M4F MCU @ 160 MHz
  - Radar hardware accelerator (HWA) @ 80 MHz
- Memory: 1.0 MB RAM
- Interfaces: SPI, UART, I2C, QSPI, GPIOs
- Power: <100mW @ 10Hz (including classifier)
- Package options:
  - ~6.45 x 6.45 mm FCCSP (0.5mm pitch)
  - ~5 x 4.5mm WCSP (0.4mm pitch)



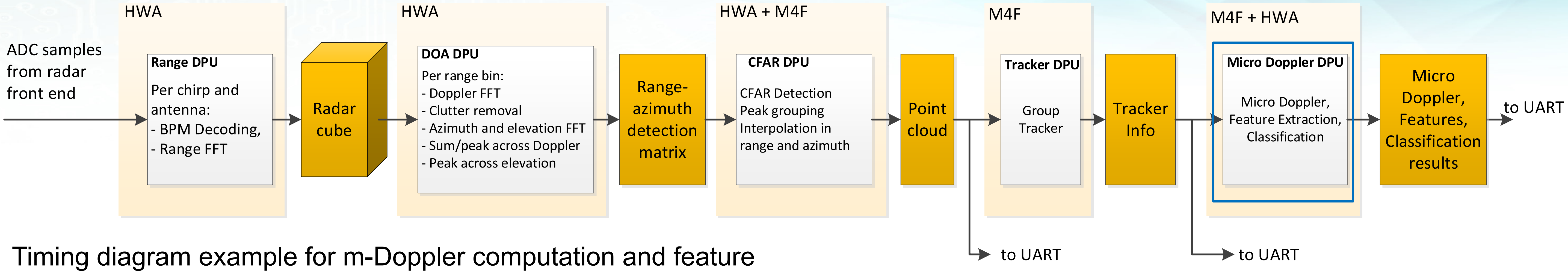
mmWave RF and Analog subsystem
Front-end controller subsystem
Application subsystem (Customer programmable)
LNA
LNA
LNA
IF
IF
IF
ADC
ADC
ADC
Internal loopback for test and monitoring
x3
Synth (20 GHz)
APLL
SOCC
GPADC
Temp sensor
PA
PA
Digital Front End (Decimation filter chain)
Ramp Generator (Timing Engine)
RF control processor (M3) (TI firmware)
Prog RAM and ROM
Data RAM
DMA
ADC Buffer
Bus Matrix
Cortex-M4F at 160 MHz
ECC
Prog + Data RAM 512 KB
Boot ROM + optional fixed routines
ECC
Radar Data Memory (L3) 512 KB (see Note)
LBIST, PBIST
ESM
Watchdog
DCC
HWA1.2 (80MHz)
QSPI
SPI
CAN-FD
UART
I2C
PWM
Test/Debug
GPIO
Serial flash interface
Control/Communication interface
Communication interface
For debug and Control/ Communication interface
PMIC control, EEPROM interface
JTAG for debug and development
DPLL
LDOs
Osc.
PRCM, RTC and Wakeup Seq
32-kHz Oscillator
Always ON low-power domain
XTAL (40 MHz)

Note: Up to 256 KB of L3 RAM can be shared with M4F

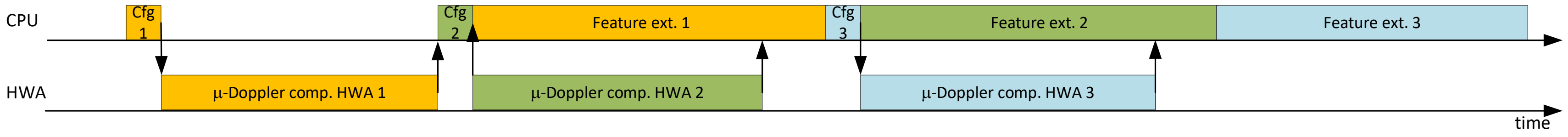
The entire real-time implementation is architected around Cortex M4F and HWA and fits into the 1MB of total memory.

Enable Edge AI – Implementation Details

Detailed view of the signal processing chain on IWRL6432. Each block is implemented as a **DPU** and partitioned across **HWA** and **M4F**.



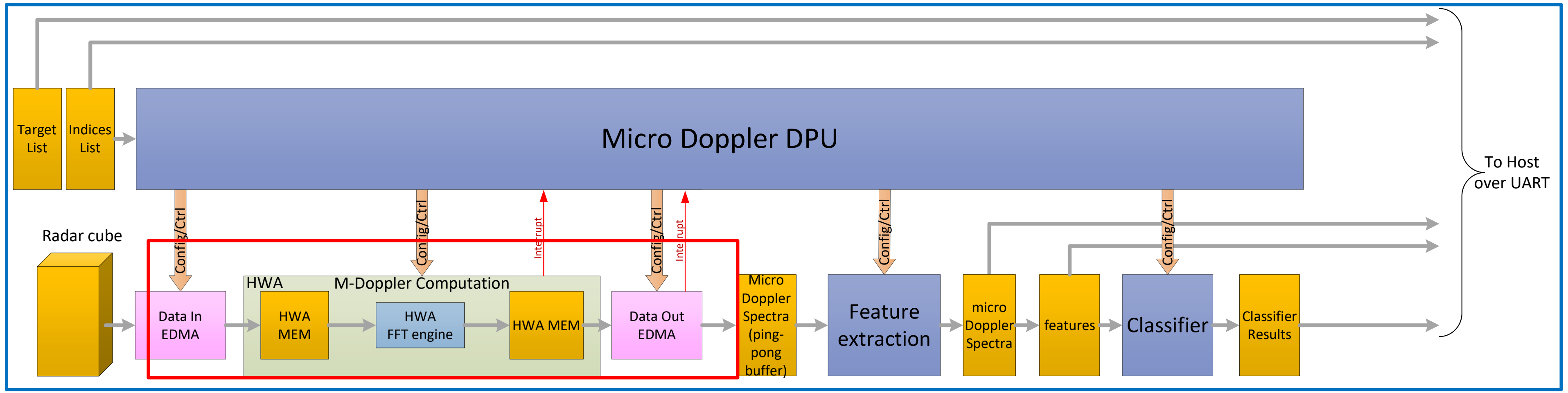
Timing diagram example for m-Doppler computation and feature extraction for three tracking objects



- The entire end-to-end chain takes only **~35ms** for 3 tracks.
- Each additional track requires an additional ~3ms (tracker + classifier).

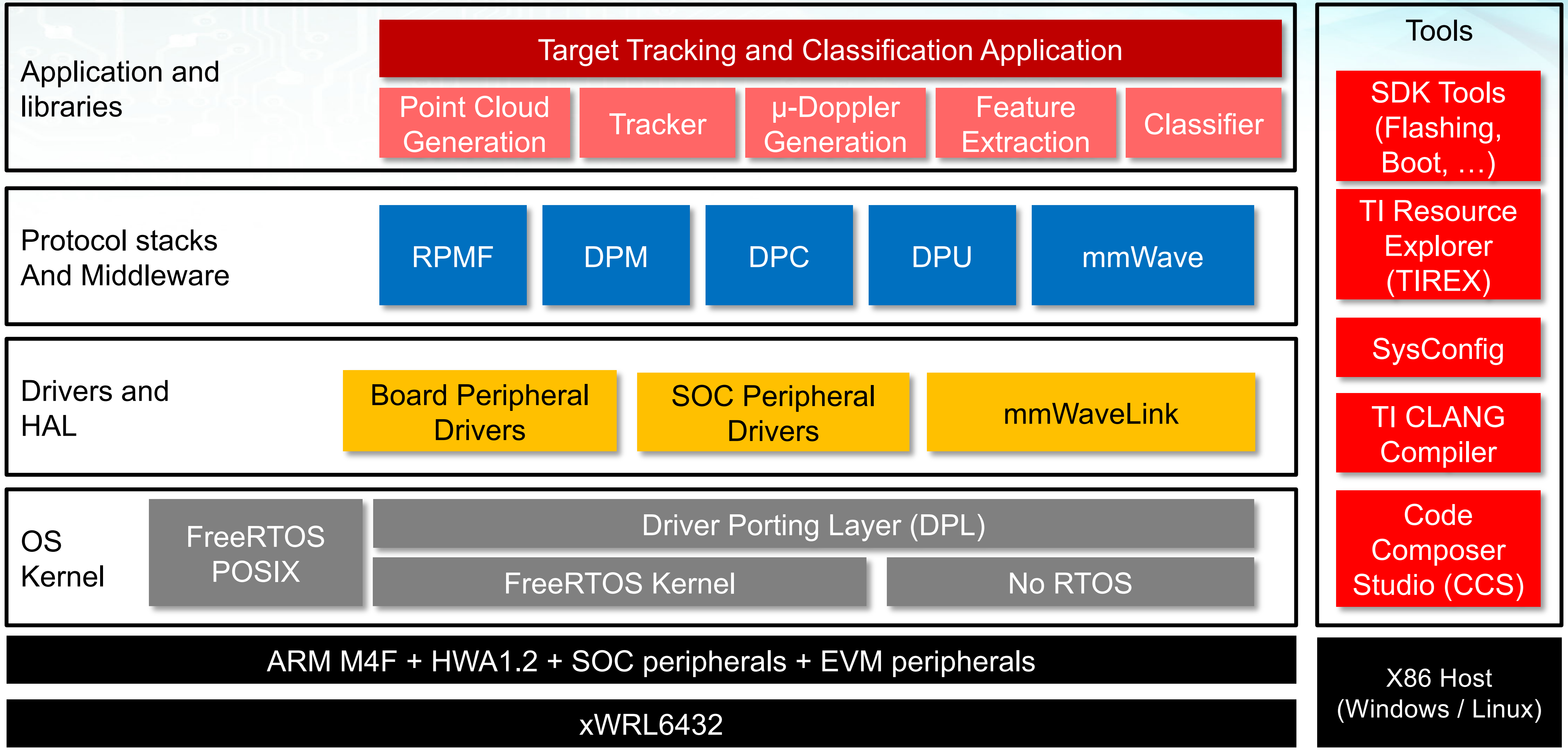
Highly parallelized architecture across processing cores to improve processing efficiency.

Efficient utilization of the HWA engine and EDMA channels is critical to the implementation, making the end-to-end chain work!



Note: See the next page for glossary.

Enable Edge AI – Software Architecture



Application and libraries

Target Tracking and Classification Application

Point Cloud Generation | Tracker | µ-Doppler Generation | Feature Extraction | Classifier

Protocol stacks And Middleware

RPMF | DPM | DPC | DPU | mmWave

Drivers and HAL

Board Peripheral Drivers | SOC Peripheral Drivers | mmWaveLink

OS Kernel

FreeRTOS POSIX | Driver Porting Layer (DPL) | FreeRTOS Kernel | No RTOS

ARM M4F + HWA1.2 + SOC peripherals + EVM peripherals

xWRL6432

Tools

SDK Tools (Flashing, Boot, …)

TI Resource Explorer (TIREX)

SysConfig

TI CLANG Compiler

Code Composer Studio (CCS)

X86 Host (Windows / Linux)

Glossary

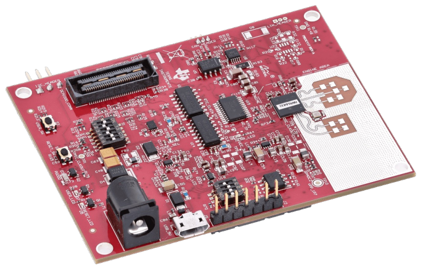
- RPMF: Radar Power Management Framework
- DPM: Data Processing Management
- DPC: Data Processing Chain
- DPU: Data Processing Unit
- SOC: System-on-Chip

Highlights:

- Open Source OS – FreeRTOS
- Simplified, low memory, low latency optimized drivers
- Pre-integrated mmWave components.
- Plug n Play middleware

Available on TI.com!

- Software: MMWAVE-L-SDK
- Evaluation module: IWRL6432BOOST



SDK

Radar Academy

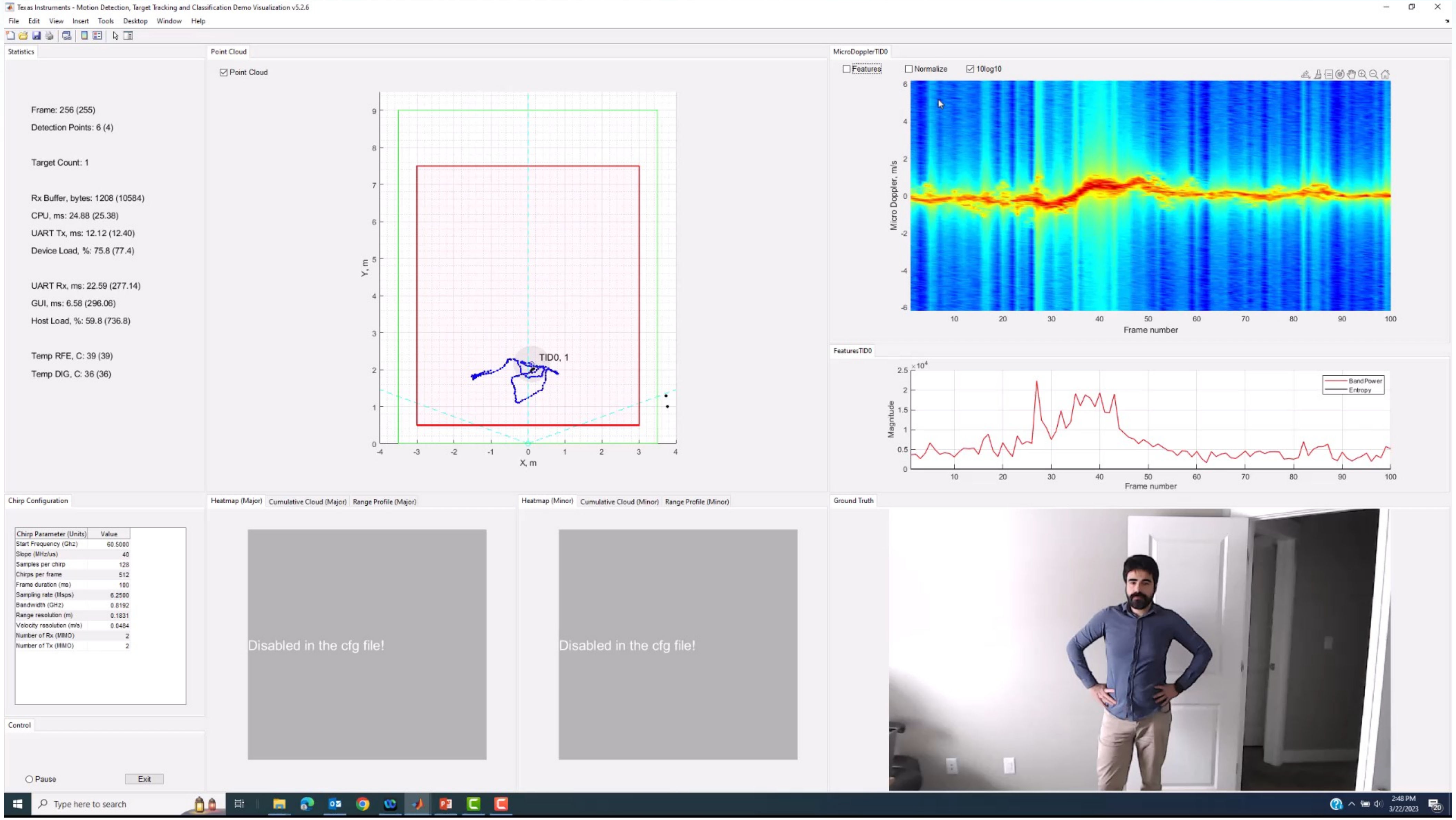
TIREX

CCS

freeRTOS

Real-Time Demo – µ-Doppler Generation

- In the real-time demo on IWRL6432, µ-Doppler spectrograms from multiple tracks can be **generated and streamed**.
- Different ML techniques can be explored using this information without putting effort into lower-level processing.



Texas Instruments - Motion Detection, Target Tracking and Classification Demo Visualization v5.2.6

Frame: 256 (255)
Detection Points: 6 (4)

Target Count: 1

Rx Buffer, bytes: 1208 (10584)
CPU, ms: 24.88 (25.38)
UART Tx, ms: 12.12 (12.40)
Device Load, %: 75.8 (77.4)

UART Rx, ms: 22.59 (277.14)
GUI, ms: 6.58 (296.06)
Host Load, %: 59.8 (736.8)

Temp RFE, C: 39 (39)
Temp DIG, C: 36 (36)

Chirp Parameter (Units)	Value
Start Frequency (Ghz)	60.5000
Slope (MHz/us)	40
Samples per chirp	128
Chirps per frame	512
Frame duration (ms)	100
Sampling rate (Msps)	6.2500
Bandwidth (GHz)	0.8192
Range resolution (m)	0.1831
Velocity resolution (m/s)	0.0484
Number of Rx (MIMO)	2
Number of Tx (MIMO)	2

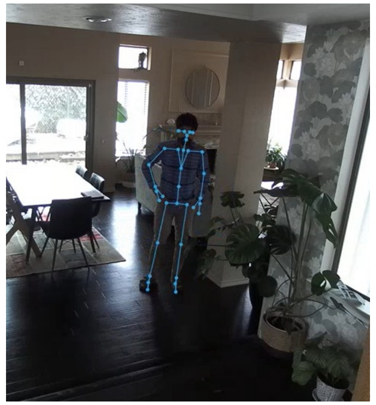
Disabled in the cfg file!

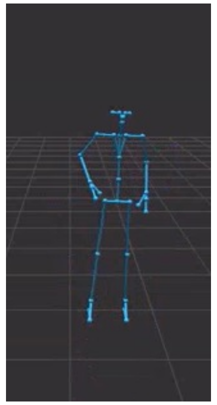
Disabled in the cfg file!

- The micro-Doppler maps of a human vs. non-human object:

RC helicopter

- This feature can also help in more use cases (pose estimation, fall detection, etc.) when privacy is important.





Summary – Available on TI.com!

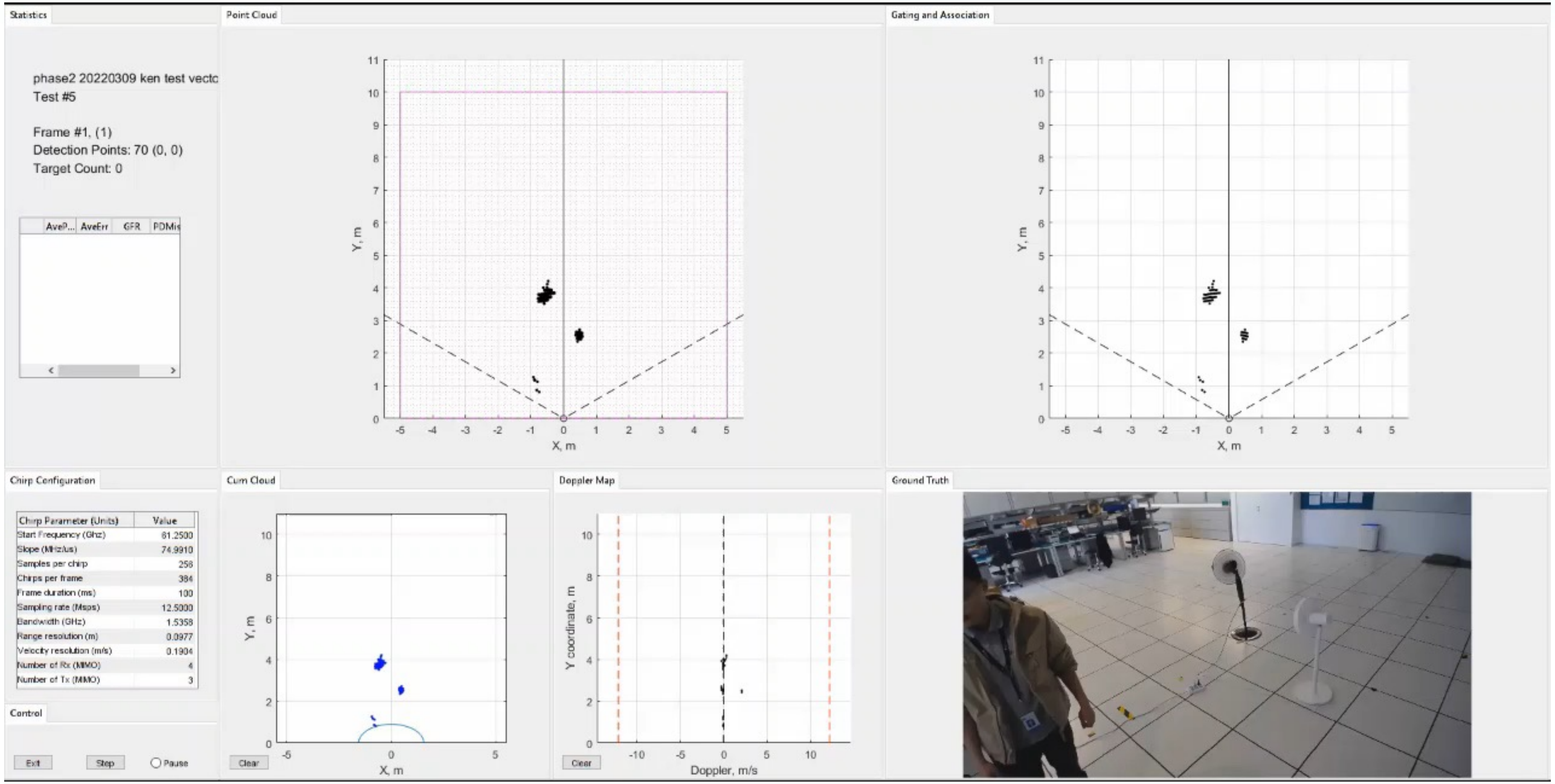
Summary: Using the multi-target tracker (MTT) integration, a robust μ-Doppler map generation method is built to support classifying **multiple objects concurrently** (to support more realistic scenarios) with reduced complexity.

- Reduced the complexity of the classifier with hand-crafted feature extraction.
- Majority of the existing works use μ-Doppler maps directly.

- Enabled **simultaneous** tracking and classification of **multiple targets** in real-world environments.
- Majority of the existing works do not address such multi-object realistic scenarios.

Important facts:

- The machine learning model has never seen this specific scenario in training.



Statistics

phase2 20220309 ken test vectc
Test #5

Frame #1, (1)
Detection Points: 70 (0, 0)
Target Count: 0

	AveP...	AveErr	GFR	PDMis

Chirp Configuration

Chirp Parameter (Units)	Value
Start Frequency (Ghz)	61.2500
Slope (MHz/us)	74.9910
Samples per chirp	256
Chirps per frame	384
Frame duration (ms)	100
Sampling rate (Msps)	12.5000
Bandwidth (GHz)	1.5358
Range resolution (m)	0.0977
Velocity resolution (m/s)	0.1904
Number of Rx (MIMO)	4
Number of Tx (MIMO)	3

TEXAS INSTRUMENTS

TINY ML Asia 2023

Thank You!
Questions?

TINY ML ASIA
Seoul, South Korea
November 16, 2023

Copyright Notice

www.tinyml.org