Integrated Sensing and Communication Physical Layer Model

Steve Blandino, Tanguy Ropitault, Neeraj Varshney, Jian Wang, Anirud Sahoo, Camillo Gentile, Nada Golmie

National Institute of Standards and Technology
U.S. Department of Commerce

WIRELESS NETWORKS DIVISION, COMMUNICATION TECHNOLOGY LAB

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY,

GAITHERSBURG, MD, USA

SEPTEMBER, 2025

Licensing Statement

Contents

Contents v
Abbreviations vi
1 Introduction 2
2 Digital Baseband Transceiver and Sensing Processor 5
2.1 Data Processing 5
2.2 Pilot Sequence Processing 6
2.3 Sensing Processing 7
3 Software Structure 9
3.1 Block Diagram of ISAC-PLM 9
3.2 Folder and Code Structure 11
4 Usage description 16
4.1 Input 16
4.2 Output 26
5 Examples 34
5.1 Point Target in Empty Room: Passive Sensing-PPDU 34
5.2 Point Target in Empty Room: Bistatic-TRN-R 39
5.3 Human Target in Living Room: Bistatic-TRN-T 46
5.4 Human Target in Living Room: Passive-Beacon 47
5.5 Threshold-based Sensing Measurement and Reporting 49
5.6 Channel Measurements: vital sign sensing 52
5.7 Lossy and Jittery Conditions 55

A List of Input Parameters **57**
References 62

Abbreviations

AP Access Point.

BPSK Binary Phase Shift Key.
BRP Beam Refinement Protocol.

CFO carrier frequency offset.
CP Cyclic Prefix.

DCM Dual Carriere Modulation.
DFT Discrete Fourier Transform.

EDMG Enhanced Directional Multi-Gigabit.

IDFT Inverse Discrete Fourier Transform.
ISAC Integrated Sensing and Communication.

LDPC Low Density Parity Check.

MCS Modulation and Coding Scheme.
mm-wave millimeter-wave.
MMSE Minimum Mean Square Error.
MSE Mean Squared Error.
MU-MIMO Multi-User Multiple-Input Multiple-Output.

NMSE Normalized Mean Squared Error.

OFDM orthogonal frequency-division multiplexing.

PHY Physical Layer.
PPDU Physical layer Protocol Data Unit.

QAM Quadrature Amplitude Modulation.
QPSK Quadrature Phase Shift Key.

RDA Range-Doppler-Angle.

SC single carrier.

STA Station.
STFT Short-Time-Fourier-Transform.
SU-MIMO Single-User Multiple-Input Multiple-Output.
SU-SISO Single-User Single-Input Single-Output.
SVD Singular Value Decomposition.

TRN Training Field.

ZF Zero Forcing.

Integrated Sensing and Communication Physical Layer Model Documentation

Integrated Sensing and Communication Physical Layer Model (ISAC-PLM) is an open-source implementation for link level simulation of millimeter-wave (mm-wave) wireless communication and sensing systems.

ISAC-PLM includes two different modes (Figure 1):

- A communication mode, modeling the Physical Layer (PHY) of IEEE 802.11ay, including a growing set of features, such as Multi-User Multiple-Input Multiple-Output (MU-MIMO) link level simulation, IEEE 802.11ay [1]single carrier (SC)/ orthogonal frequency-division multiplexing (OFDM) waveform generation, synchronization, channel estimation, carrier frequency offset (CFO) estimation and correction.

- An ISAC mode, extending the IEEE 802.11ay model to support future wireless systems integrating communication and sensing, such as IEEE 802.11bf [2]. ISAC-PLM introduces Doppler processing to obtain an estimation of the range, velocity, and angles of moving targets using two different techniques:

  – Reusing an IEEE 802.11ay SC/OFDM communication link, extracting sensing information from existing pilot sequences, referred as passive sensing in this document.

  – Transmit dedicated sensing packets, containing pilot sequences, which are used exclusively to perform sensing tasks, referred to as active sensing.

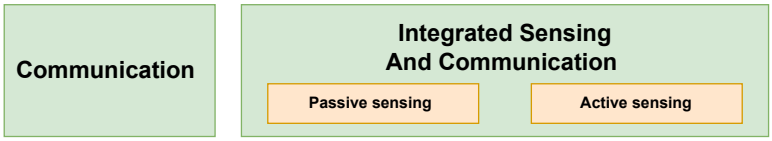


Figure 1: ISAC-PLM simulates wireless communication or Integrated Sensing and Communication (ISAC) systems.

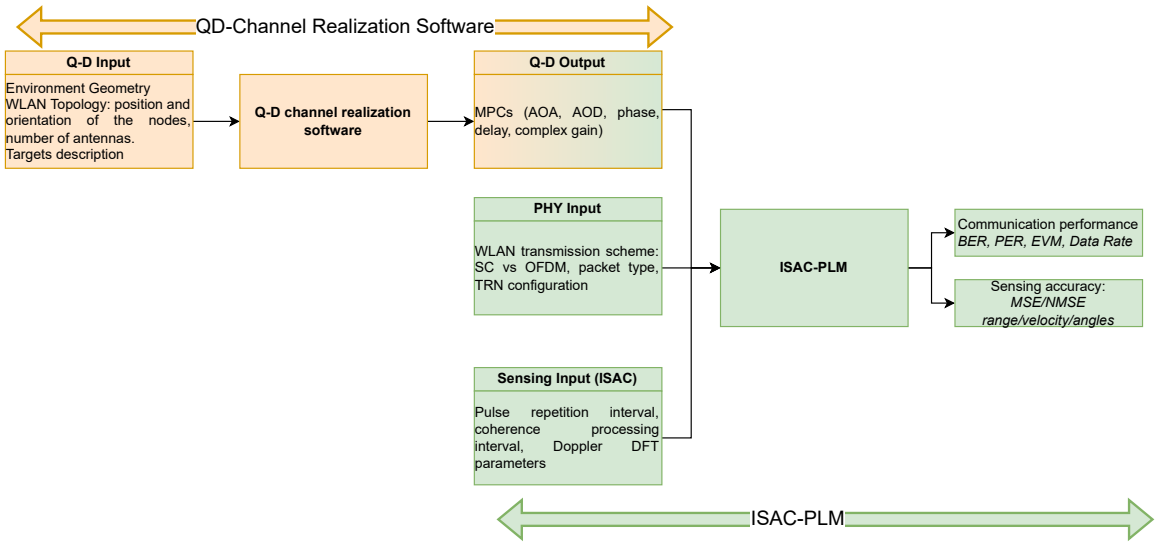


Figure 2: ISAC-PLM workflow.

1 Introduction

The ISAC-PLM is part of a collection of tools developed to support both communication systems and ISAC systems design and evaluation [3]. The ISAC-PLM models the end-to-end IEEE 802.11ay PHY processing based on a Matlab implementation[1]. It also extends the communication functionalities with sensing features. Sensing through wireless communication networks aims at utilizing communication signals to detect and sense targets enabling applications such as human presence detection, gesture recognition, or object tracking. The ISAC-PLM supports native integration with the open-source NIST Q-D channel realization software [4] to enable link-level simulations with realistic mm-wave channel models. The ray tracing method used in the NIST Q-D channel realization software, unlike conventional stochastic models, is 3-D environment and transceiver position specific, hence enabling accurate end-to-end performance evaluation. The NIST Q-D channel realization supports sensing applications evaluation by allowing to simulate channels including moving target. Reader interested about generating Q-D channels with targets support are invited to refer to Section 2.5 of the Q-D software documentation [4].

Figure 2 presents an overview of the ISAC-PLM workflow and its integration with the NIST Q-D channel realization software. The geometry of the environment and the topology of the scenario, e.g., position of the nodes and targets are defined in the NIST Q-D channel realization software.

The Q-D software receives the target coordinates evolution over time as input, and in turn, provides the multipath components (MPCs) as an output. These in-

[1]Requirements: R2021b+, WLAN toolbox

clude both the MPCs resulting from the signal's interaction with the target (target-related MPCs) and those generated from the signal's interaction with the environment (target-unrelated MPCs). The Q-D output is used as the input of ISAC-PLM . However, ISAC-PLM usage is not limited to channels generated by the NIST Q-D channel realization software. If the multipath components input provided matches the one expected by the ISAC-PLM (described in Section 4.1.4), the output of any channel model implementation or measurement campaign can be used as input for the ISAC-PLM .

ISAC-PLM , as shown in Figure 2, defines input parameters to design the digital baseband transmitter and receiver algorithms (*PHY input*), and it can be configured with a wide range of parameters that define the properties of the communication system and the transceiver settings. For ISAC mode, the sensing processing design is enabled by the *Sensing Input*, which allows to configure the micro-doppler processing parameters.

At the output of the ISAC-PLM , if the communication mode is enabled, the following communication metrics are returned:

- Bit Error Rate (BER).
- Packet Error Rate (PER).
- Constellation Error Vector Magnitude (EVM).
- Data Rate.

When using ISAC-PLM in ISAC mode the following sensing metrics are returned:

- Histogram of the accuracy of detected velocity.
- Histogram of the accuracy of detected range.
- Histogram of the accuracy of detected azimuth and elevation.
- Sensing accuracy in terms of MSE/NMSE.
- Channel state information.
- Beam SNR.

The key communication features of the software can be summarized as follows:

- IEEE 802.11ay SC/OFDM waveform generation.
- IEEE 802.11ay spatial multiplexing schemes up to 8 streams: Single-User Single-Input Single-Output (SU-SISO), Single-User Multiple-Input Multiple-Output (SU-MIMO), MU-MIMO.
- Receiver signal processing algorithms: synchronization, channel estimation, carrier frequency offset (CFO) correction.

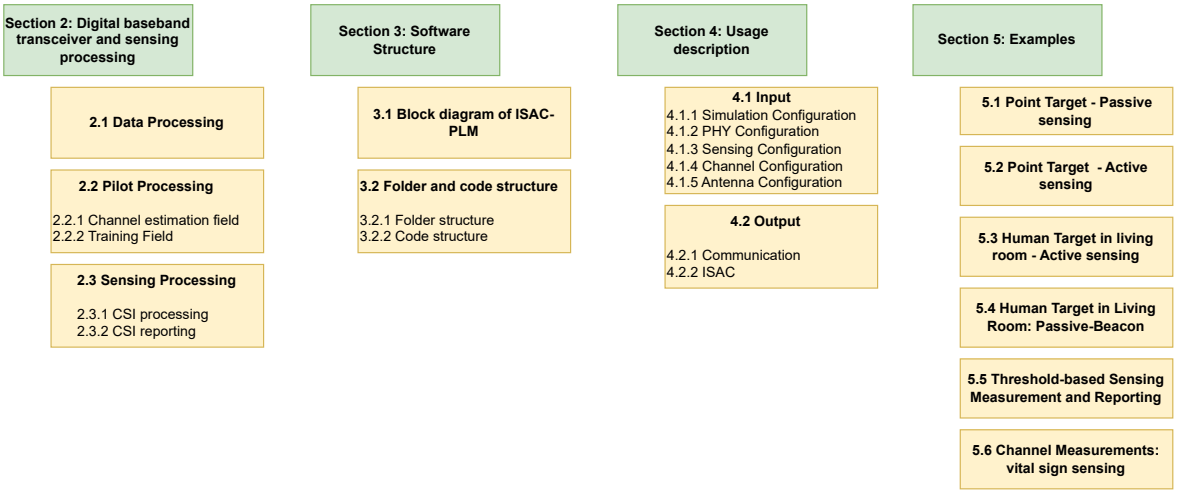


Figure 3: Organization of the ISAC-PLM documentation

- Perform link-level bit error rate (BER), packet error rate (PER), data rate tests and analytical link-level spectral efficiency (SE).

The key sensing features of the software can be summarized as follows:

- Sensing signal processing algorithms: clutter removal, Doppler processing, target detection, range, velocity and angle estimation.
- Sensing accuracy analysis in terms of Mean Squared Error (MSE) and Normalized Mean Squared Error (NMSE).

The rest of the documentation, depicted in Fig. 3, is organized as follows:

- Section 2 describes the PHY model and gives an overview of the algorithms used both for communication and sensing.
- Section 3 describes the architecture of the software, describing folder, code structure, and major functions.
- Section 4 explains how to use the software and the input/output design.
- Section 5 presents some examples.

2 Digital Baseband Transceiver and Sensing Processor

The digital transceiver extends the IEEE 802.11ad implemented in the MATLAB WLAN toolbox[2] to be IEEE 802.11ay compliant [5]. It supports SU-MIMO and-MU-MIMO for both OFDM and SC modes. It also supports several precoder and equalizer options, including Singular Value Decomposition (SVD), Zero Forcing (ZF) precoder, block diagonalized-ZF precoder, and Minimum Mean Square Error (MMSE) equalizer.

The Enhanced Directional Multi-Gigabit (EDMG) PPDU is made of three main parts: preamble, data and training fields (TRN). The data (Section 2.1) is decoded using the aid of a known pilot sequence contained in the preamble (Section 2.2). The proposed digital transceiver can perform synchronization of the received signal, as well as estimate the channel from the Golay pilot sequences provided in the IEEE 802.11ay packet preamble. The channel can be estimated also using TRN fields (2.2.2). Using the preamble of the PPDU or the TRN fields in sensing mode, a sensing processor determines the presence of moving targets using the evolution of the estimated channel. Finally, a DMG beacon frame has been implemented. The DMG beacon frame consists of the DMG packet preamble, which is steered in different directions. Hence, the angular-dependent channel response can be estimated using the same processing presented in Section 2.2.

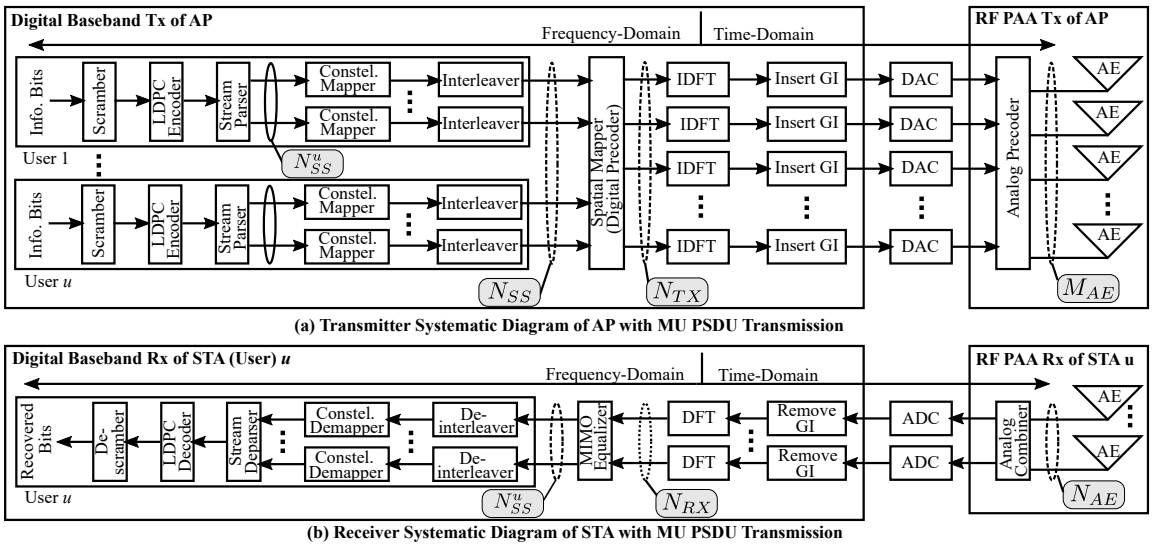
2.1 Data Processing

Fig. 4 shows the baseband OFDM MU-MIMO signal processing of the EDMG digital transceiver.

As shown in Fig. 4(a), the blocks at the Access Point (AP) transmitter, including scrambler, Low Density Parity Check (LDPC) encoder, stream parser, and constellation mapper, are operated on a per-user basis; while the remaining processing in the transmit chains, including the spatial mapper, Inverse Discrete Fourier Transform (IDFT), and Cyclic Prefix (CP) insertion, are carried out per digital antenna (or digital chain). The digitally precoded MU-MIMO symbols of each transmit chain are OFDM modulated with a 512 point IDFT, followed by the CP insertion.

The Station (STA) receiver procedure is illustrated in Fig. 4(b). The OFDM symbols are demodulated by Discrete Fourier Transform (DFT) operation followed by the CP removal and a MMSE MIMO equalizer. Subsequently, the constellation demapping, stream deparsing, and LDPC decoding are carried out to complete the PHY signal processing procedure. The LDPC encoder and decoder as well as constellation mapper and

[2]Certain commercial equipment, instruments, or materials are identified in this paper in order to specify the experimental procedure adequately. Such identification is not intended to imply recommendation or endorsement by the National Institute of Standards and Technology, nor is it intended to imply that the materials or equipment identified are necessarily the best available for the purpose.



(a) Transmitter Systematic Diagram of AP with MU PSDU Transmission

(b) Receiver Systematic Diagram of STA with MU PSDU Transmission

Figure 4: Transceiver diagram of IEEE 802.11ay OFDM mode, in MU-MIMO downlink transmission.

demapper support a variety of Modulation and Coding Schemes (MCSs) for OFDM mode transmission with different coding rates to comply with the IEEE 802.11ay standard, including Dual Carriere Modulation (DCM)-Binary Phase Shift Key (BPSK), DCM-Quadrature Phase Shift Key (QPSK), 16-Quadrature Amplitude Modulation (QAM), and 64-QAM.

2.2 Pilot Sequence Processing

2.2.1 Preamble Processing

The preamble contains known pilot sequences, namely Golay sequences, in the legacy short time field (L-STF), in the EDMG-STF, in the legacy channel estimation field (L-CEF), and in the EDMG-CEF. STF and CEF are used in the communication receiver signal processing, for time and frequency synchronizations and channel state information (CSI). These operations are achieved in multiple steps:

- The frame is detected and synchronized by finding the peak of the correlation between the received signal and the known STF pilots.
- Coarse frequency offset is computed by comparing the phase changes to the peaks of the STF correlation.
- SNR estimation is obtained from STF.
- Joint frequency offset and channel estimation are performed by correlating the received CEF with the known CEF.

2.2.2 TRN Processing

At the end of an IEEE 802.11ay Physical layer Protocol Data Unit (PPDU) or in an IEEE 802.11ay Beam Refinement Protocol (BRP) packet, Golay sequences can be sent in the Training Field (TRN) to train the receiver and transmitter antenna. TRN can be used to estimate the CSI. IEEE 802.11bf allows for the re-use of BRP frames for sensing purposes, and as of the writing of this documentation, it also introduces a dedicated multi-static PPDU that consists of a sequence of TRN fields.

ISAC-PLM uses multi-static PPDUs to analyze the sensing performance when using directional communications. TRN-R/TRN-T can be precoded using different antenna weight vectors (AWVs), such that the sensing information can be extracted from the channel components that fall into the analog beam created by the selected AWVs. TRN-R/TRN-T are used also to provide an SNR estimate.

2.3 Sensing Processing

ISAC-PLM enables sensing for SISO communication links by introducing a sensing processing that extracts sensing information from the estimated CSI at the receiver. The CSI can be obtained from the CEF of the PPDU or from the CEF of the DMG beacon frame. Both are already available in conventional IEEE 802.11ay systems, and they do not require a dedicated transmission, hence we refer to them as passive sensing. The CSI can be obtained also from TRN sequences transmitted with the purpose of performing sensing. We refer to this mode as active sensing. Active sensing can be bistatic if involves two stations, multi-static if involves more than two stations.

2.3.1 CSI processing

CSI is used to sense the environment by tracking the channel variations over time. Assuming a static transmitter and receiver, a perturbation of the CSI is likely to be caused by a change in the environment, such as a moving target. The sensing processing is executed at the receiver: the receiver, process the PPDUs sent by the transmitter and extract sensing information, beyond being a conventional communication receiver. From a remote sensing perspective, the channel estimated using the pilot sequences can be seen as echoes from the targets and the environment. The delays of the echoes from the targets MPCs are proportional to the bi-static distance, i.e. the distance the signal travels from the transmitter to the receiver, scattering out of the target.

ISAC-PLM can perform sensing using a IEEE 802.11ay PPDU (passive sensing) or using an IEEE 802.11bf multi-static PPDU (active sensing).

- Passive sensing: assuming IEEE 802.11ay PPDUs sent continuously over time, with omni antennas, the sensing processor can build the radar data matrix, collecting the estimated CSI (from the preamble) at each packet reception in a 2-dimensional matrix, i.e., the delay (referred also as fast time/range) and the evolution over the time (referred also as also slow time) [6].

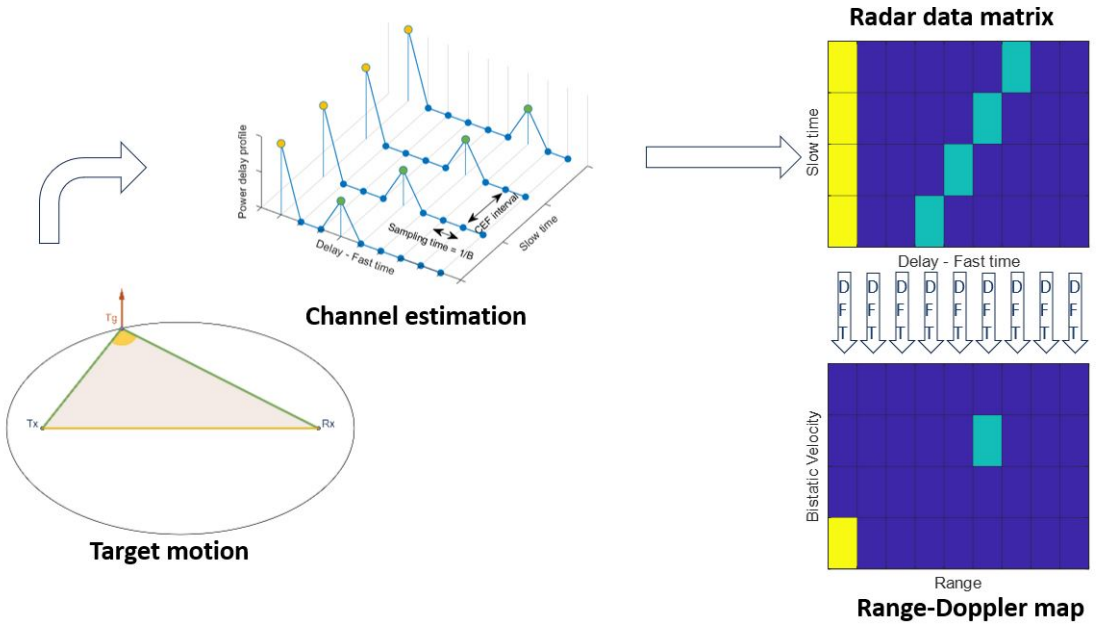


Figure 5: Doppler processing using channel estimation.

- Active sensing: Assuming directional communication, ISAC-PLM precodes the TRN fields, when using a multi-static PPDU. The estimated CSI (from the TRN) is collected in a 3-dimensional matrix including information about the angular domain beyond fast time and slow time [7].

To obtain the velocity of the targets, a discrete Fourier transform (DFT) is applied on the slow time dimension of the radar data matrix to allow the estimation of the target speed. A visual representation of the Doppler processing is shown in Figure 5. In case of directional communication with multi-static PPDU, the process is repeated for each transmit-receive AWVs configuration tested.

Null Doppler values in the range-Doppler map, i.e. null velocity, are considered static clutter, i.e., echoes coming from the static environments, thus they are not relevant to the remote monitoring. They can be filtered out, by removing the continuous component along the slow time dimension of the radar data matrix. At the output of the filter only echoes with a non-zero Doppler component are retained as they can be associated with a change in the environment.

2.3.2 Beam-SNR Fingerprinting

ISAC-PLM returns beam signal-to-noise ratios (SNRs). These measurements are more informative (e.g., in the spatial domain) than RSSI measurements and easier to access than CSI measurements as the beam SNRs are required to be estimated, using a DMG beacon frame, and reported by users during beam training [8]. From the variations of the SNR over time, sensing can be inferred. As the beacon is sent as conventional

signalling in communication and it is not dedicated for sensing, we refer to as passive sensing.

ISAC-PLM returns beam signal-to-noise ratios using either a DMG beacon or exploiting the IEEE 802.11bf multi-static PPDU.

- Passive sensing: the DMG beacon is normally used for beamforming training, by sounding the channel in different directions. A STA listening to the beacon estimates the SNR for each beam and store the values for Beam-SNR Fingerprinting. The estimated SNRs, obtained from the L-STF, are collected in a matrix associating the SNR to the beam index for each measurement instance. Using the DMG beacon, no additional sensing overhead is needed.

- Active sensing: TRN fields in the multistatic PPDU are used to estimate the SNR. The estimated SNRs (from the TRN) are collected in a matrix associating the SNR to the angular domain information for each measurement instance. Using the multi-static PPDU, additional sensing overhead is needed.

2.3.3 CSI Reporting

ISAC-PLM implements a reporting method defined to minimize the effect of the CSI reporting on the communication.

To minimize the signalling overhead, in the threshold-based sensing measurement and reporting procedure, the sensing receiver feedbacks the CSI only when the estimated CSI variation between successive packets becomes significant. More specifically, the receiver first quantify the CSI variations, comparing the current and previous estimated CSI measurements. If the CSI variation is higher than a threshold, the receiver feedbacks the estimated CSI to the transmitter, otherwise the transmitter reconstruct the missing CSI measurement using interpolation techniques.

This software also supports an adaptive threshold method where the threshold can be adapted to the motion considered. With the adaptive-threshold method the objective is to adjust the CSI-variation threshold dynamically to avoid degrading the sensing performance.

3 Software Structure

3.1 Block Diagram of ISAC-PLM

The ISAC-PLM is organized in three main building blocks: EDMG transmitter, EDMG receiver and in ISAC mode a sensing processor. In ISAC mode ISAC-PLM implements different sensing type, as shown in Figure 6:

- Passive sensing: this mode simulate frames available in conventional DMG/EDMG systems and sensing is derived without explicit signalling, hence no sensing overhead is added.

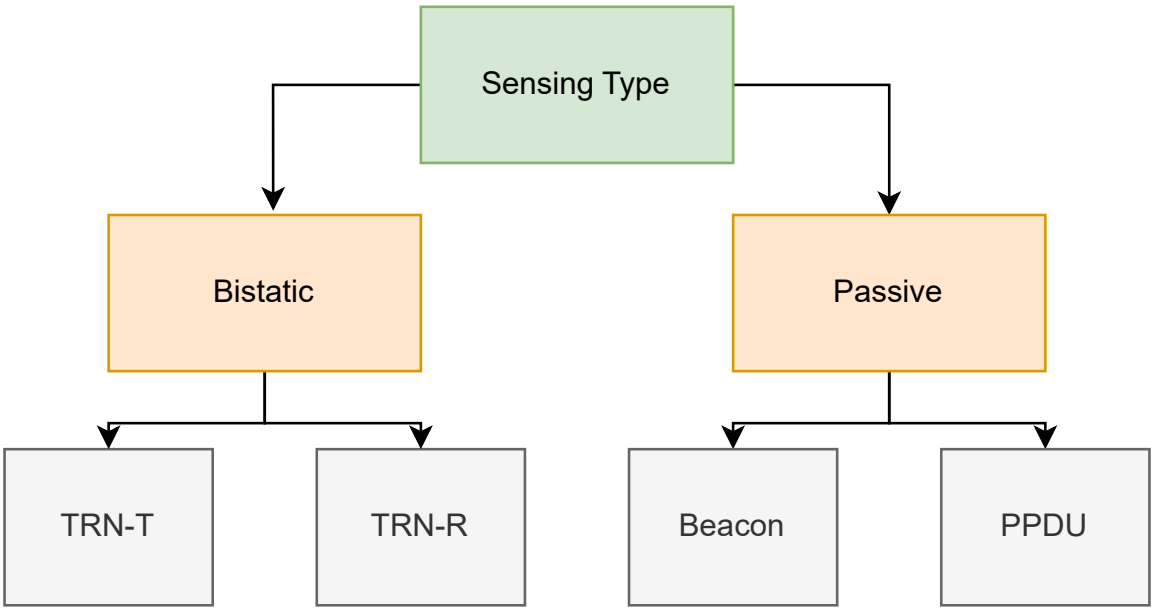


Figure 6: Sensing Types.

  - Beacon: it uses the beacon, which is steered towards multiple directions, generally for beamforming training.
  - PPDU: it uses the EDMG-PPDU, assuming an omnidirectional transmission

- Bistatic sensing (Active): this mode simulate 2 STAs (one transmitter and one receiver) exchanging dedicated sensing pilots (TRN) adding overhead on the communication.

3.1.1 ISAC-PLM transmitting PPDUs

Figure 7 shows the block diagram of ISAC-PLM with data transmission, when a PPDU is used for communication or the CEF of the PPDU is used to perform passive sensing:

- The EDMG transmitter is in charge to encode randomly generated bits by LDPC, modulate encoded bits to a constellation and apply SC or OFDM modulation to obtain the waveform.

- The EDMG receiver gets in input the waveform passed through the channel model with added white noise. The receiver performs packet by packet synchronization, SC/OFDM demodulation and channel estimation. The channel estimation is used to obtain the equalizer to retrieve the constellation. Demodulation and LDPC decoding allows to retrieve the information bits.

- When using the software in passive ISAC mode, an additional block is considered. The sensing processor uses the channel estimation computed using CEF

(IEEE 11ay packet). The channel estimation is first filtered through a clutter removal filter, that eliminates the static background. The Doppler processing returns range-Doppler maps enabling target velocity, range estimation. Target information are obtained finding peaks in the range-Doppler map.

3.1.2 ISAC-PLM transmitting Multi-Static PPDUs

Figure 8 shows the block diagram of ISAC-PLM in active sensing mode when sending dedicated sensing pilots, without data transmission:

- The EDMG transmitter is in charge of generating a dedicated sensing sequence.
- The EDMG receiver processing is simplified performing only synchronization and channel estimation.
- The sensing processor uses the channel estimation computed using TRN. The channel estimation is first filtered through a clutter removal filter, that eliminates the static background. The Doppler processing returns range-Doppler-angle maps enabling target velocity, range and angle estimation. Target information are obtained finding peaks in the range-Doppler-angle map.

3.1.3 ISAC-PLM transmitting Beacon

Figure 9 shows the block diagram of ISAC-PLM in passive sensing mode and without data transmission when the legacy preamble of the beacon is used for SNR and channel estimation :

- The DMG/EDMG transmitter generates the beacon and sounds the channel on different sectors (different angular directions).
- The DMG/EDMG receiver processing performs synchronization, channel estimation, and SNR estimation on the legacy preamble.

3.2 Folder and Code Structure

3.2.1 Directory Structure

The directory structure is shown in Figure 10. The MATLAB code is located in the folder `isac-plm` containing the main script `main.m` to run the software as a communication link or as an ISAC link. The folder `example` collects predefined configurations. Each of the examples has an `Input` folder defining the configuration files and an `Output` folder that is generated by the ISAC-PLM with the result of the simulation.

The folder `isac-plm\src` contains the Matlab code and includes several sub-folders:

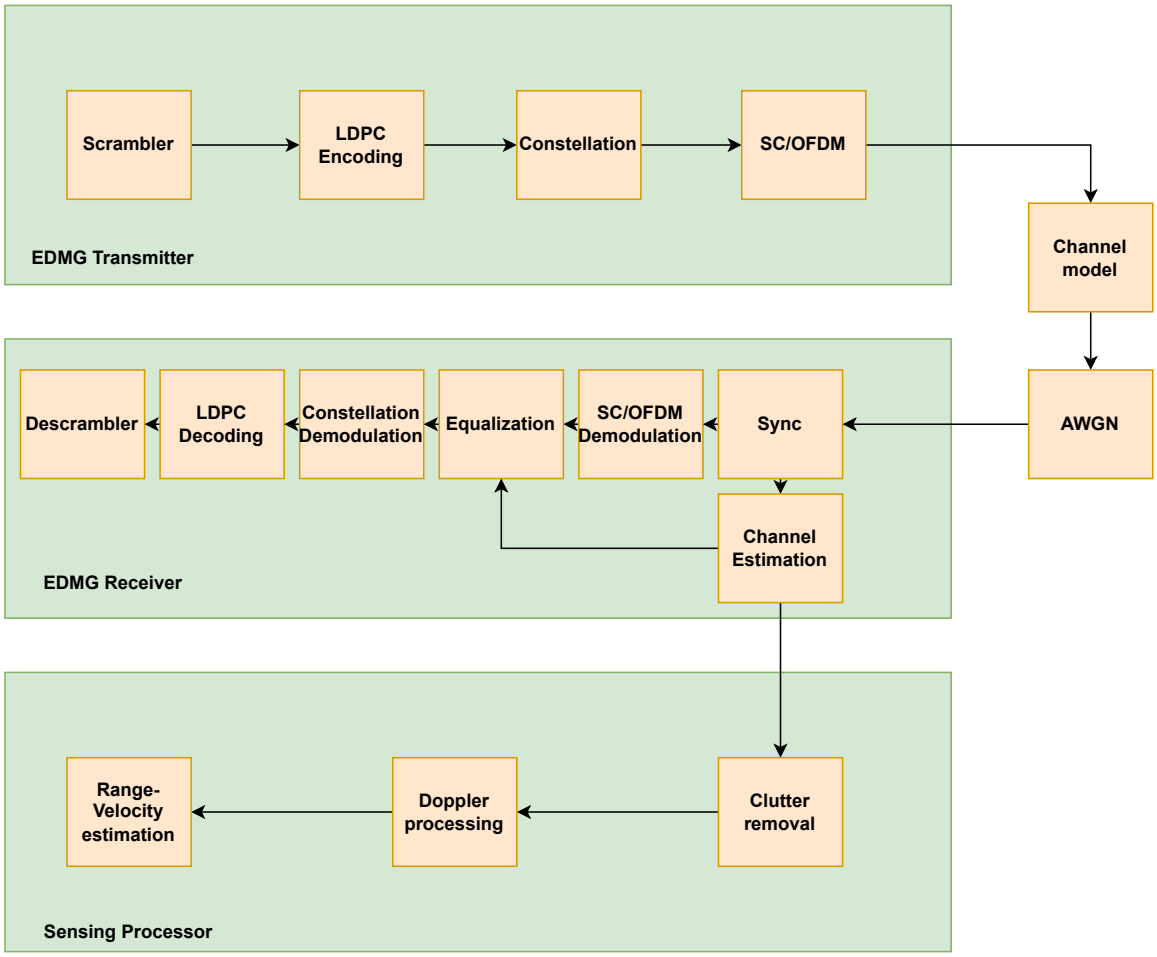


Figure 7: ISAC-PLM block diagram: communication and passive sensing.

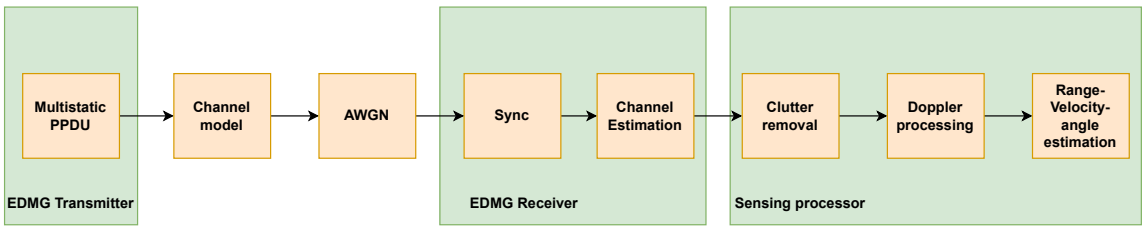


Figure 8: ISAC-PLM block diagram: multi-static PPDU for sensing (no data transmission).

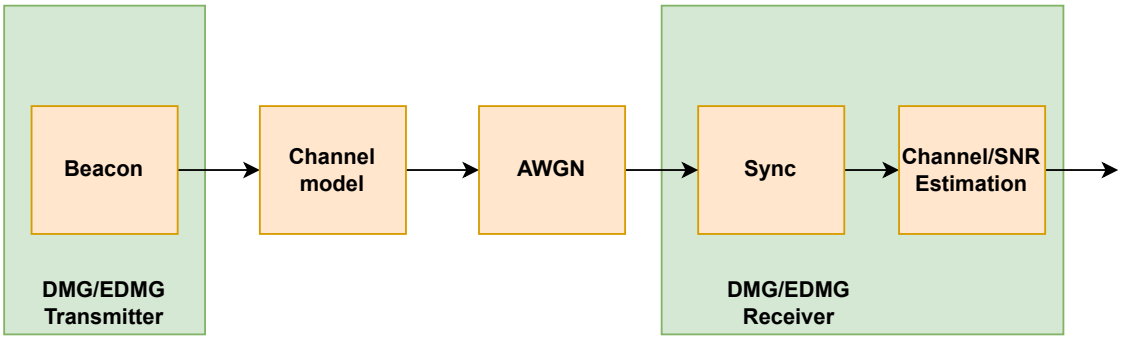


Figure 9: ISAC-PLM block diagram: a beacon for sensing (no data transmission).

- `channel`: includes the function to generate different channel models, as well as the function to convert the MPCs described in the NIST Q-D channel realization software format to a tapped delay line (TDL) channel model.
- `config`: configuration functions. It includes the files to load the input configuration and set default values.
- `data`: stores data needed to run the software, e.g., angle codebook.
- `ext`: external code files.
- `phy`: link-level simulations functions.
  - +nist: MathWorks authored codes revised by NIST.
- `sens`: includes the sensing functions.
- `tools`: includes helper functions.
- `src-antenna-model`: collects the functions to read the angle codebook.

3.2.2 Code Structure

The script `main.m` is used to launch the simulation. When the software is used to simulate a communication link `main.m` invokes `runPHYErrorRateDataRate.m` and `runPHYSpectralEfficiency.m` to run the link-level simulation obtaining BER, PER, data rate and spectral efficiency. When the software is used to simulate an ISAC link, `main.m` invokes `runIsac` to obtain, the sensing accuracy of moving targets. When using ISAC with a conventional IEEE 802.11ay PPDU BER, PER, data rate are also provided.

The ISAC-PLM uses four major structures throughout the implementation. These structures are:

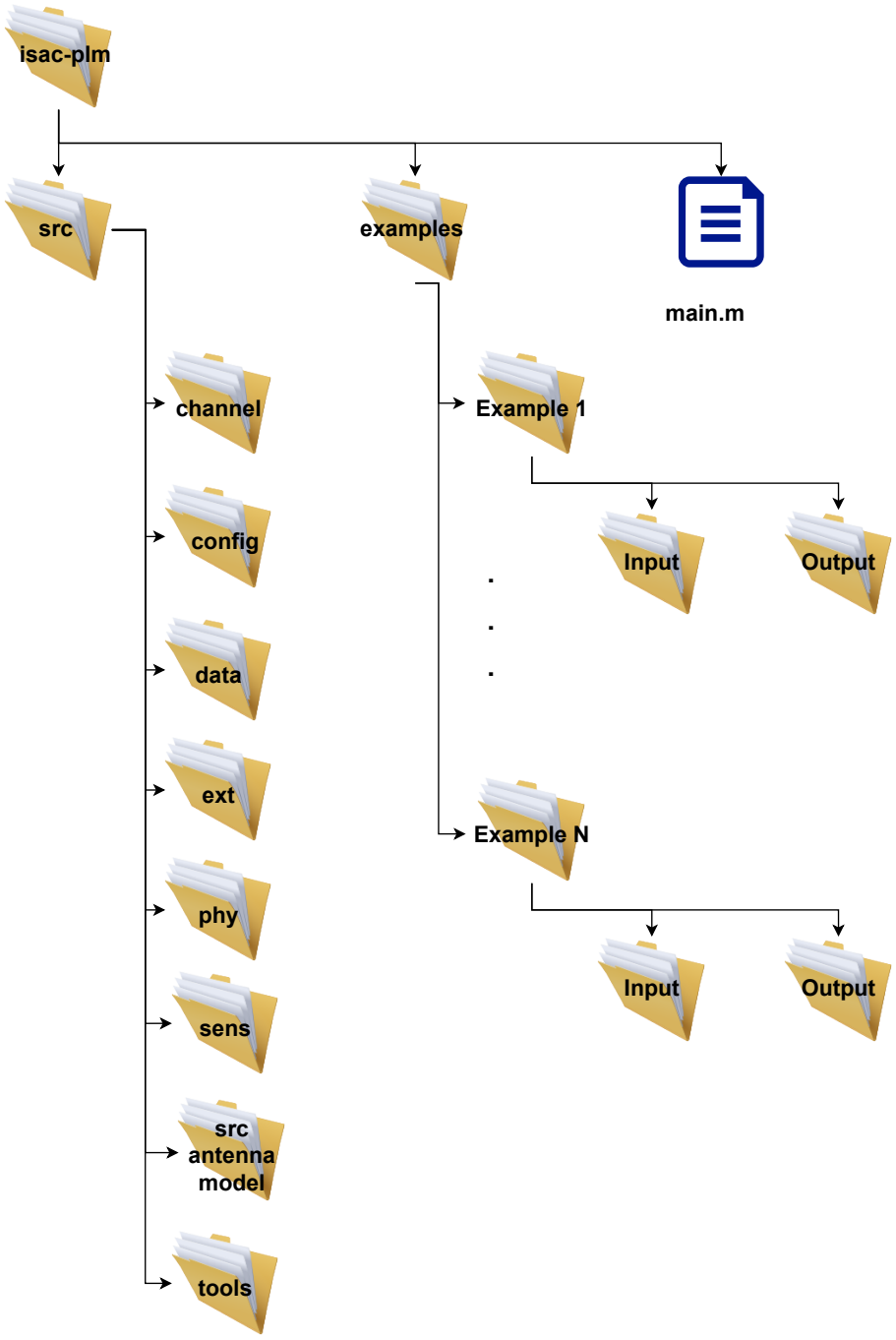


Figure 10: ISAC-PLM folder structure.

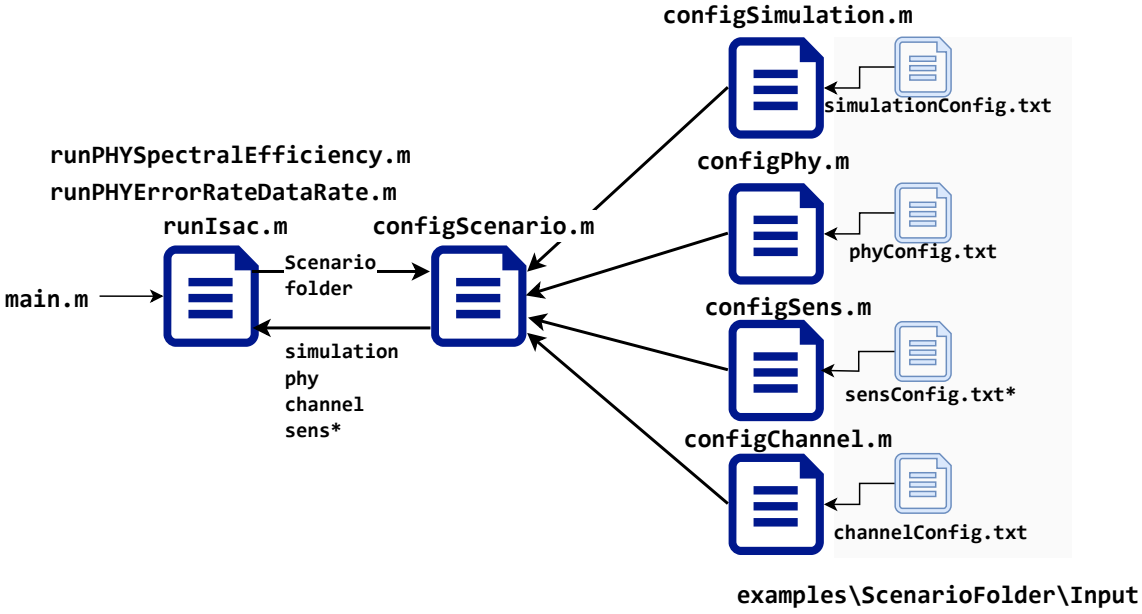


Figure 11: Code structure.

- `simulation`: it contains information about the simulation and the simulator setting. It includes parameters to set noise value, non-idealities, or to enable debug mode or plots.
- `phy`: the structure contains all the information relative to the transceiver operation and the algorithms used in the PHY.
- `sens`: this structure is used in ISAC mode and it contains the setting of the sensing processing.
- `channel`: it includes parameters related to the channel model.

These structures are constructed with dedicated configuration files. The configuration files provided as input and described in Section 4.1 are loaded using the functions `configSimulation.m`, `configPhy.m`, `configSens.m` and `configChannel.m`, defines inside `src\config`. In case some input parameter is missing in the configuration, these function initializes the parameter to a default value. The code structure is illustrated in Figure 11: the function `runPHYErrorRateDataRate` (for communication) or the function `runIsac` (for ISAC) invokes the `configScenario.m` function passing as input the scenario folder path, containing the configuration files. `configScenario.m` returns in output the Matlab structures used throughout the software.

After loading the configuration the ISAC-PLM uses three main functions implementing the three building blocks shown in Figure 7:

- The function `edmgTx.m` generates the IEEE 802.11ay transmit waveform.

- The function `edmgRx.m` implements the receiver processing.
- In ISAC mode, the function `sensingProcessing.m` implements the sensing processing on the estimated channel state information at the receiver.

4 Usage description

This section describes how to setup the ISAC-PLM and how to analyze the results obtained.

4.1 Input

The input folder must be properly defined in the scenario folder (for instance, `isac-plm-\examples\scenarioFolder\Input`). The configuration is broken down into four configuration files relative to the four main structures used in ISAC-PLM:

- `simulationConfig.txt` (Section 4.1.1).
- `phyConfig.txt` (Section 4.1.2).
- `sensConfig.txt` (Section 4.1.3) - ISAC mode.
- `channelConfig.txt` (Section 4.1.4)

Each file contains in each row a pair parameter-value separated by a tab space. In case of directional transmission using phased antenna arrays (PAAs), an extra file describing the antenna configuration of each node is used (Section 4.1.5).

- `nodePaa<x>.txt`, being `<x>` the index of the node starting from 0.

The input parameters of each file are described in the following and an overview of the accepted parameters is provided in appendix A.

4.1.1 Simulation Configuration

The simulation configuration aims at selecting the properties of the simulation that the users want to analyze and choose the operation mode of the simulator.

The file `simulationConfig.txt` may contain the following fields:

- `psduMode`: select the packet format.

  0 for PPDU format (header + data), 1 for PSDU format (data-field only).

  In communication mode: using the PPDU format, the receiver estimates the channel from the preamble to obtain the channel equalizer. If the field is not

defined, `psduMode` is set to 1 and the channel is assumed perfectly known at both transmitter and receiver.

In ISAC mode: `psduMode` is always set to 0, i.e., the channel is estimated (either from CEF or TRN).

- `dopplerFlag`: doppler flag.

  0: Doppler off (block fading). 1: Doppler ON. (Default = 0)

- `snrMode`: SNR definition.

  `EsNo`: Ratio of symbol energy to noise power spectral density

  `EbNo`: Ratio of bit energy to noise power spectral density

  `snr` Signal-to-noise ratio (SNR) per sample.

  (Default value: `EsNo`)

- `snrAntNormFlag`: SNR Rx antenna normalization flag.

  0: The receive signal is normalized such that the receive power at each antenna is 1 in average. 1: The receive signal is normalized such that the receive power at each antenna is 1. (Default 0)

- `snrRange`: Simulation SNR range in dB.

  Define the SNR range in dB specified as a 1-by-2 vector of scalar. (Default [0 20])

- `snrStep`: Define the SNR step in dB specified as a positive scalar.

  The simulation SNR points are generated in the range snrRange with a step of snrStep, i.e. snrRange(1):snrStep:snrRange(2). (Default 1)

- `maxNumErrors`: Maximum number of bit errors specified as a positive integer before sending a new packet. (Default = 100)

- `maxNumPackets`: Maximum number of packets specified as positive integer before quitting the simulation. (Default = 1000)

- `saveWs`: Save workspace. 0: Do not save workspace. 1: Set workspace. (Default = 0)

Additional ISAC-specific parameters:

- `nTimeSamp`: Number of channel time samples.

  Specify the length in channel samples of the simulation. Each channel sample corresponds to a packet transmission. This value must be lower or equal to the number of channel samples in time provided as input (see Channel Configuration).

- `sensPlot`: Save sensing plot.

  Specified as 1 to save the plots otherwise 0 to not save the plots. The plots saved are described in Section 4.2.2.

- `saveRdaMap`: Save Range-Doppler-Angle (RDA) maps

  When setting the saveRdaMap = 1, RDA maps are saved in jpeg files. The plots saved are described in Section 4.2.2. Setting the option `disableRdaAxis` = 1, the jpeg files are saved without axes.

- `saveCsi`: Return in output `csi.json`

  When setting the `saveCsi` = 1, CSI estimates are stored in `csi.json`. (Default `saveCsi` = 0).

4.1.2 PHY Configuration

The PHY configuration defines the properties of the PHY, the frame used, and the algorithms the PHY uses in simulation.

The file `phyConfig.txt` may contain the following fields:

- `phyMode`: PHY mode.

  `phyMode` is specified as OFDM or SC. (Default OFDM in communication mode, SC in ISAC mode)

- `lenPsduByt`: Length of PSDU.

  `lenPsduByt` is the length of PSDU expressed in byte. (Default = 4096)

- `giType`: Guard interval length.

  `giType` is specified as Short, Normal or Long.

- `numSTSVec`: Number of downlink spatial-time streams.

  `numSTSVec` specified as 1-by-STAs vector of positive integers in the range 1-8 such that sum(`numSTSVec`)<=8. (Default = 1) For ISAC, `numSTSVec` must be set to 1.

- `smTypeNDP`: Spatial mapping type for non-data packet (preamble only, non data-field of PPSU)

  Specified as Hadamard, Fourier, Custom or Direct. (Default is Direct)

- `smTypeDP`: Spatial mapping type for data packet (PSDU, data-field of PPSU).

  smTypeDP specified as Hadamard, Fourier, Custom or Direct. (Default is Direct)

- `mcs`: Modulation and coding scheme (MCS) Index.

  mcs is specified as index in the range 1-20 (21 for SC). (Default = 6).

- `processFlag`: transceiver digital processing flag.

  `processFlag` selects specific MIMO signal processing for OFDM/SC mode transmitter and receiver. processFlag is specified as a scalar between 0-5 (Default 0).

  - For the transmitter processing,
    - processFlag = 0, OFDM supports both SC SISO and SU-MIMO without transmitter precoding.
    - processFlag = 1, OFDM supports SU- and MU-MIMO with frequency-domain precoding based on regularized zero-forcing (RZF) criteria; SC supports SU-MIMO with time-domain one-tap precoding based on RZF criteria.
    - processFlag = 2, OFDM supports SU-MIMO with frequency-domain precoding based on SVD and RZF filtering; SC supports SU-MIMO with time-domain one-tap precoding based on SVD and RZF filtering.
    - processFlag = 3, OFDM supports MU-MIMO with frequency-domain precoding based on SVD and RZF filtering; SC supports SU-MIMO with time-domain one-tap precoding based on SVD and RZF filtering.
    - processFlag = 4, OFDM supports SU- and MU-MIMO with frequency-domain precoding based on block diagonalization and ZF filtering (BD=ZF); SC supports SU-MIMO with time-domain one-tap precoding based on BD-ZF.
    - processFlag = 5, SC supports MU-MIMO with time-domain multi-tap precoding with ZF.
  - For the receiver processing, all processFlag = 0-5 support OFDM/SC joint frequency-domain equalization and MIMO detection based on linear minimum mean squared error (MMSE) criteria.

- `symbOffset`: Symbol sampling offset.

  `symbOffset` is specified as values from 0 to 1. When `symbOffset` is 0, no offset is applied. When `symbOffset` is 1 an offset equal to the GI length is applied. (Default 0.75)

- `softCsiFlag`: Demodulation with soft channel state information flag.

  `softCsiFlag` is specified as 0 for disabled or 1 for enabled. (Default 1 for OFDM and 0 for SC)

- `ldpcDecMethod`: LDPC decoding method.

`ldpcDecMethod` is specified as `norm-min-sum` or `bp`, respectively. (Default norm-min-sum)

- `sensingType`: DMG Sensing Type.

  `sensingType` is specified as `bistatic-trn` (multi-static PPDU), `passive-ppdu` (PPDU), `passive-beacon` (Beacon), or `none` (sensing disabled). (Default `none`)

To configure the TRN design as specified in [2] when `sensingType` is specified as `bistatic-trn`:

- `packetType`: Define TRN type

  Define TRN type, specified as 'TRN-R' for receiver training, 'TRN-T' for transmitter training.

- `trainingLength`: Number of TRN Units

  Indicates the number of TRN Units present in the TRN field of the PPDU.

- `subfieldSeqLength`: Length of subfield sequence (Golay).

  Define the number of samples of the subfield as 128, 256 or 64.

- `unitP`: TRN P sequence

  Value indicated by the EDMG-TRN-Unit P defined as 0,1,2,3.

- `unitM`: TRN M sequence

  Value indicated by the EDMG-TRN-Unit M field.

- `unitN`: TRN N sequence

  Value indicated by the EDMG-TRN-Unit N field.

- `UnitRxPerUnitTx`

  RX TRN-Units per Each TX TRN-Unit.

4.1.3 Sensing Configuration

The sensing configuration aims at defining the parameters used in the sensing processing. The file `sensConfig.txt` may contain the following field:

- `pri`: pulse repetition interval

  Specifies the interval in seconds between each packet as a positive real number. Default 0.0005.

- `dopplerFftLen`: Doppler FFT length.

  Specifies the Doppler FFT length as a positive integer. Default 64.

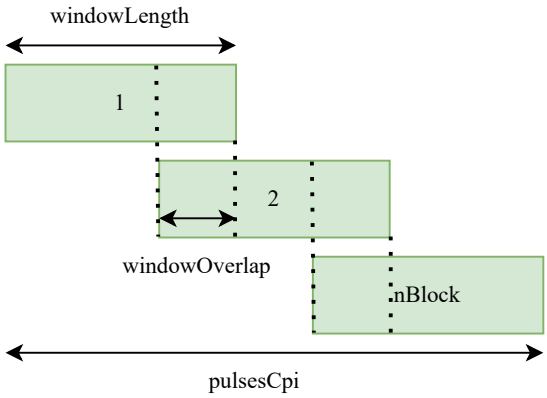


Figure 12: Short-Time Fourier transform input arguments (`windowLength`, `windowOverlap`, `pulsesCpi`)

- `pulsesCpi`: number of pulses in a coherent processing interval.

  Specifies the number of packets to collect before to compute the doppler FFT. Default 64.

- `window`: FFT window.

  FFT window specified as `rect`, `hamming`, `blackmanharris`, `gaussian`. Default `rect`.

- `windowLen`: Short-Time-Fourier-Transform (STFT) window length

  Split data before doppler FFT in overlapping windows of length `windowLen` (Figure 12). Default 64.

- `windowOverlap`: STFT window overlap

  Specifies the percentage of the window overlapping between 0 and 1 (Figure 12). Default 0 (no overlap).

- `cfarGrdCellRange`: Number of guard cells in the range dimension.

- `cfarGrdCellVelocity`: Number of guard cells in the velocity or Doppler dimension.

- `cfarTrnCellRange`: Number of training cells in the range dimension.

- `cfarTrnCellVelocity`: Number of training cells in the velocity or Doppler dimension.

- `cfarThreshold`:Threshold level for detection in dB.

- `clutterRemovalMethod`: high pass clutter removal method defined as:

  `'remove_static_component'`,`'recursive_first_order_hp'`, `'none'`. (Default `'remove_static_component'`)

Optional for threshold-based sensing:

- `thresholdSensing`: enables threshold-based sensing measurement and reporting procedure to analyze the sensing performance. Default = 0.
- `adaptiveThreshold`: activates adaptive thresholding for threshold-based sensing. Default = 0.
- `numTimeDivisions`: specifies the number of time-divisions for which the threshold has to be adaptively calculated. This is valid only for adaptiveThreshold = 1. Default = 5.
- `threshold`: specifies the threshold value used for threshold-based sensing. In case adaptive threshold is activated i.e., adaptiveThreshold = 1, this specifies the initial threshold. Default = 0.
- `stepThreshold`: specifies the step size for threshold. This is valid only for adaptiveThreshold = 1. Default = 0.1.
- `percentMeasurement`: specifies the percent of the measurements expected by the node for sensing processing. This is valid only for adaptiveThreshold = 1. Default = 90.
- `csiVariationScheme`: defines the CSI variation scheme used for threshold based sensing. The software currently supports `EucDistance`, `TRRS`, and `FRRS`. This is valid only for `thresholdSensing` = 1. Default = `TRRS`.
- `interpolationScheme`: defines the interpolation scheme used for missing measurement in threshold-based sensing. The software currently supports `previousMeasuremment`, `linearInterpolation`, and `zeroPadding`. This is valid only for `thresholdSensing` = 1. Default = `previousMeasuremment`.
- `sparseCsi`: when set to 1, the simulator consumes a timing file (`sensingTiming.csv`) that encodes real/irregular measurement times and losses; when 0, the simulator uses an ideal, uniform schedule. In sparse mode the timing file is auto-loaded during configSens.

  Sparse CSI Timing File (`sensingTiming.csv`)

  When `sparseCsi`=1, this CSV defines the actual (irregular) arrival times of CSI measurements. Each row corresponds to one measurement (one DMG Sensing Measurement Exchange). The canonical schema used by our generator is:

  ```
  TIME,SRC_ID,DST_ID,SETUP_ID,BURST_ID,INSTANCE_ID,SNR
  ```

Fields

`TIME` Timestamp in nanoseconds (ns) since scenario start.

`SRC_ID / DST_ID` Logical node IDs for TX/RX (integers).

`SETUP_ID` Scenario/run identifier (integer) - Not used in the current implementation.

`BURST_ID` CPI/burst index $(1, 2, \ldots)$.

`INSTANCE_ID` Index of the measurement within a burst (1..`pulsesCpi`). Gaps (missing numbers) indicate lost exchanges.

`SNR` Optional (dB) metadata.

- `interBI` (inter-burst interval): time between consecutive DMG sensing bursts. This is valid only when `sparseCsi`=1.

4.1.4 Channel Configuration

The channel configuration specifies the properties of the channel over which the packets are transmitted.

- In communication mode, several channel models can be used, such as AWGN and Rayleigh.
- To evaluate the performance of an ISAC system, any channel model or measurements provided in the format of the NIST Q-D channel realization model can be used.

The channel model is configured with the file `channelConfig.txt`. The file `channelConfig.txt` must contain the field `chanModel`. When using ISAC-PLM in communication mode, the field `chanModel` must be specified as:

- AWGN
- Rayleigh

When using ISAC-PLM in sensing mode, the field `chanModel` must be specified as:

- Sensing

AWGN

To use an AWGN channel it is enough to specify the property `chanModel` as `AWGN` in the configuration file `channelConfig.txt`.

Rayleigh Channel Parameter Configuration

The file `channelConfig.txt` may contain the following fields:

- `numTaps`: Number of channel taps.

  Number of channel taps specified as a positive integer. If the field is not defined `numTaps` is set to 10.

- `pdpMethodStr`: Power delay profile for tapped-delay line (TDL) channel model.

  `pdpMethodStr` is specified as `PS`, `Equ` or `Exp`, using phase shift, equal power or exponential power, respectively.

- `tdlTypeChannel` Interpolation Method of CIR.

  `tdlTypeChannel` is specified as `Impulse` or `Sinc`.(Default `Impulse`)

Sensing

ISAC-PLM provides a native integration with the Q-D channel realization output format specifying the property `chanModel` as `sensing` in the configuration file `channelConfig.txt`.

The folder `Input\sensingChannel` must include the file `qdOutput.json` and may include the file `qdTargetOutput.json`, both described in Figure 13. `qdOutput.json` includes all the MPCs, including target and environment. The file `qdTargetOutput.json` is used when a reference MPC describing the target is available, and it will be used to compute accuracy metrics. Each row of the file describes a combination transmitter/transmitter-paa receiver/receiver-paa. Each row includes the following fields:

- delay (s)
- gain (dB)
- phase (rad)
- AOD EL (deg), AOD AZ (deg)
- AOA EL (deg), AOA AZ (deg)

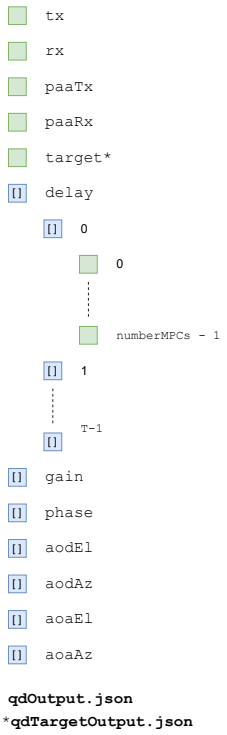


Figure 13: Channel Input: qdOutput.json and qdTargetOutput.json

Each of these fields is a vector of length T, i.e., the number of channel samples. Moreover, each time sample entry is a vector describing the properties at each MPC. The file `qdOutput.json` contains the description of all the MPCs of the channel. The file `qdTargetOutput.json`, contains only the information of a single MPC per target (specified by the target field), which can be assumed as the reference to compute ground truth velocity, range and angles. The ground truth information it is used to provide accuracy metrics of detection. In case the file `qdTargetOutput.json` error analysis is not provided. An example on how to use the ISAC-PLM together with the Q-D channel realization software is provided in Section 5.1.

The file `channelConfig.txt` may contain the following fields:

- `normalization`: normalization of the channel. Normalization of the channel specified as `packet` for a per packet normalization of the channel or `none` to avoid any normalization.

4.1.5 Antenna configuration

When simulating directional propagation in active sensing mode, the antenna pattern and the direction of the antenna steering defined by the antenna wave vector, need to be taken into account. ISAC-PLM provides a set of predefined angular codebooks, specifying the array steering vectors and the antenna wave vectors.

The antenna pattern has one main beam which is steered to a particular direction. The entries of the angle-based codebook define the possible steering directions of the PAA. The provided codeboks are relative to rectangular PAA of different sizes:

- 2×8: 2 vertical antennas, 8 horizontal antennas.

  Codebook Azimuth: [0 11 23 34 45 56 68 79 90 270 281 292 304 315 326 337 349]

  Codebook Elevation: [0 45 90 135 180]

- 4×8: 4 vertical antennas, 8 horizontal antennas.

  Codebook Azimuth: [0 11 23 34 45 56 68 79 90 270 281 292 304 315 326 337 349]

  Codebook Elevation: [0 22 45 67 90 113 135 158 180]

- 8×8: 8 vertical antennas, 8 horizontal antennas.

  Codebook Azimuth: [0 11 23 34 45 56 68 79 90 270 281 292 304 315 326 337 349]

  Codebook Elevation: [0 11 22 33 45 56 67 78 90 101 112 123 135 146 157 168 180]

The file `nodePaa<x>.txt` loading one of the predefined channel, may contain the following fields that allows to load the correct codebook:

- Codebook: specify the angle-based codebook.

  Select the PAA dimension and load the angle-based codebook. Specified as one of the codebooks provided: 'cb2x8.txt', 'cb4x8', 'cb8x8'. cbRxC referrers to a rectangular PAA with R rows and C columns. The files `nodePaa<x>.txt` are optional for isotropical transmission. The files `nodePaa<x>.txt` may be defined for directional transmission. `nodePaa0.txt` refers to the AP. `nodePaa1.txt` refers to the STA.

4.2 Output

ISAC-PLM provides as output the results of the link-layer simulation and the results of the sensing accuracy.

4.2.1 Communication output

The evaluation of the communication system is provided in terms of BER, PER, EVM and data rate, while an ISAC system is evaluated also in terms of the sensing accuracy, hence, velocity and range MSE and NMSE are provided.

The results of the simulation are stored in `isac-plm\examples\ScenarioFolder\Output`. The file `isac-plm-ws.mat` stores the following information:

- `berIndiUser`: BER for each individual user.
- `perIndiUser`: PER for each individual user.
- `evmIndiUser`: EVM expressed in dB for each individual user.
- `berAvgUser`: Average BER among users.
- `perAvgUser`: Average PER among users.
- `evmAvgUser`: Average EVM among users.
- `gbitRateIndiUser`: Data rate for each individual user.
- `gbitRateAvgUser`: Average data rate for each individual user.
- `gbitRateSumUser`: Sum data rate.

4.2.2 Sensing output

In case of an ISAC simulation ISAC-PLM provides in `isac-plm\examples\scenarioFolder-\Output\Sensing` the results in JSON format. Three files are provided:

sensingInfo.json

`sensingInfo.json`: stores variables relative to time and velocity scale as well as the ground truth of the target information of range and velocity.

- `axVelocity`: vector of velocities bins expressed in m/s. The length is `dopplerFftLen`.
- `axFastTime`: vector of delay bins relative to sync point expressed in s. The length of the vector is `maxDelayBin`.
- `timeOffset`: time offset expressed in s to map the relative fast time `axFastTime` into an absolute temporal scale `axAbsFastTime` as `axAbsFastTime` = `axFastTime` - `timeOffset`.
- `axDopFftTime`: vector of slow time bins at which the Doppler FFT is computed, expressed in s.

- `axPri`: vector of slow time bins expressed in s.
- `gtRange`: ground truth range expressed in m.
- `gtVelocity`: ground truth velocity expressed in m/s.
- `gtAz`: ground truth azimuth expressed in deg.
- `gtEl`: ground truth elevation expressed in deg.

Optional output for threshold-based sensing:

- `normCSIVarValue`: normalized CSI variation values for threshold-based sensing.
- `threshold`: fixed or adaptive threshold values.
- `adaptiveThreshold`: report if the adaptive threshold was selected in the configuration.

targetEstimation.json

`targetEstimation.json`: stores variables relative to the estimated target information such as range, velocity, and angles:

- `snr`: SNR
- `range`: range estimation expressed in m.
- `velocity`: velocity estimation expressed in m/s.
- `angleAz`: azimuth estimation expressed in deg.
- `angleEl`: elevation estimation expressed in deg.

rda.json

`rda.json`: stores the denoised range-Doppler-angle map. The denoised range-Doppler-angle map is a sparse matrix, hence only the N non-zero elements of the matrix are saved. The file includes the subscripts corresponding to the velocity, range, and angle vectors defined in `sensingInfo.json`:

- `snr`: SNR
- `sensInstanceId`: index identifying the FFT block processed at the time corresponding to the `dopplerFftId`th entry of the vector `axDopFftTime`.
- `axisRange`: range indices of the N non-zero entries of the RDA matrix. Range value can be retrieved from the vector axFastTime.

- `axisVelocity`: velocities indices of the N non-zero entries of the RDA matrix. Velocities value can be retrieved from the vector axVelocity.
- `axisAngle`: angle indices of the N non-zero entries of the RDA matrix. Angles value can be retrieved from the vector axAngle.
- `reflectionPower`: Value in linear scale of the N non-zero entries of the RDA matrix.

sensingResults.json

`sensingResults.json`: stores the sensing accuracy, estimated range, and Doppler and the raw range-Doppler map:

- `snr`: SNR
- `rangeNMSEdB`: range NMSE expressed in dB scale.
- `velocityNMSEdB`: velocity NMSE expressed in dB scale.
- `rangeMSEdB`: range MSE expressed in dB scale.
- `velocityMSEdB`: velocity MSE expressed in dB scale.
- `beamSnr`: Beam SNR in dB scale
- `elErr`: elevation error in degree.
- `azErr`: azimuth error in degree.

csi.json

`csi.json`: stores the CSI estimates as:

- `snr`: SNR
- `measurementInstanceId`: Time index.
- `sectorId`: Beam index.
- `csiI`: I component of CSI estimate.
- `csiQ`: Q component of CSI estimate.

Figure 14: Sensing Output.

Sensing Plot

In case of an ISAC simulation, ISAC-PLM can provide output plots in `\Output-\Sensing\Figures` by setting the field sensPlot = 1 in `configSimulation.txt`. The following plots are provided in .fig format:

- Range-Doppler map (Fig. 15a). The Range-Doppler map is plotted for each `dopplerFftId`.
- Micro-Doppler (Fig. 15b): evolution of the velocity over time.
- Range vs time (Fig. 15c): evolution of the range over time.

If ground truth information is provided (see Section 4.1.4):

- Histogram of velocity estimation accuracy (Fig. 16a).
- Histogram of range estimation accuracy (Fig. 16b).

When using active sensing mode (`msSensing`= 1 in `phyConfig.txt`):

- Azimuth-elevation maps. The azimuth-elevation map is plotted for each `dopplerFftId`(Fig 17a).
- Azimuth vs time (Fig 17b).
- Elevation vs time (Fig 17c).

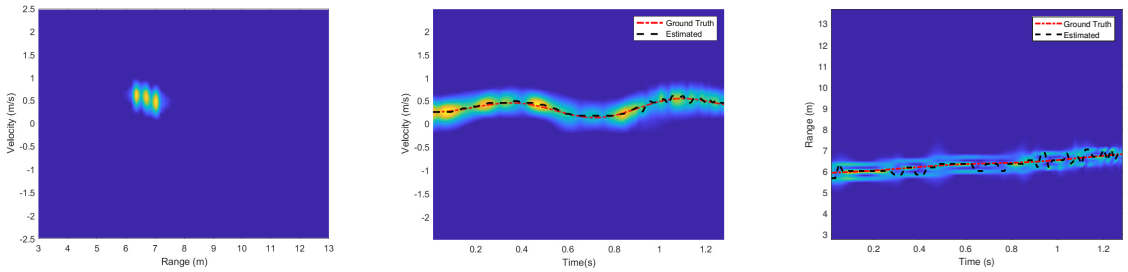
When using active sensing mode, if ground truth is provided:

- Histogram of azimuth estimation accuracy.
- Histogram of elevation estimation accuracy.

Optional plots for threshold-based sensing:

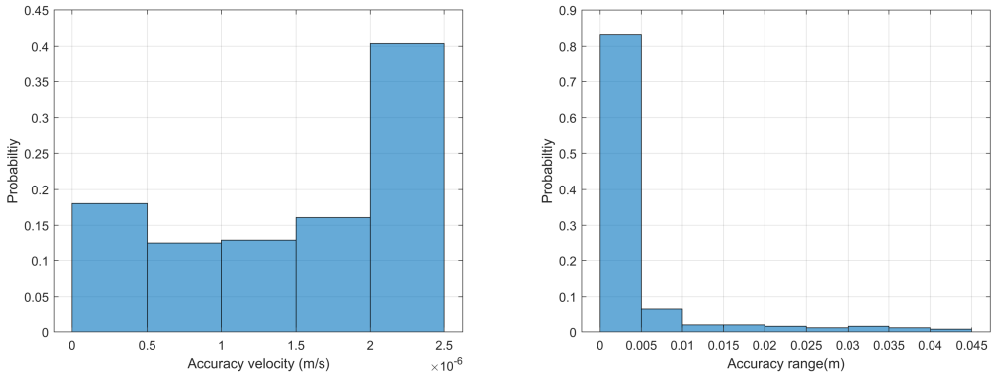
- Normalized CSI variation vs time: variation of normalized CSI variation with respect to time.
- Number of measurements vs threshold: number of measurements required to feedback for varying threshold. Also, provides the number of measurements required to feedback for a threshold considered in the simulation.

When setting `saveRda = 1` jpeg files of the 2D plots extracted from the 3D RDA are saved as shown in Figure 18.



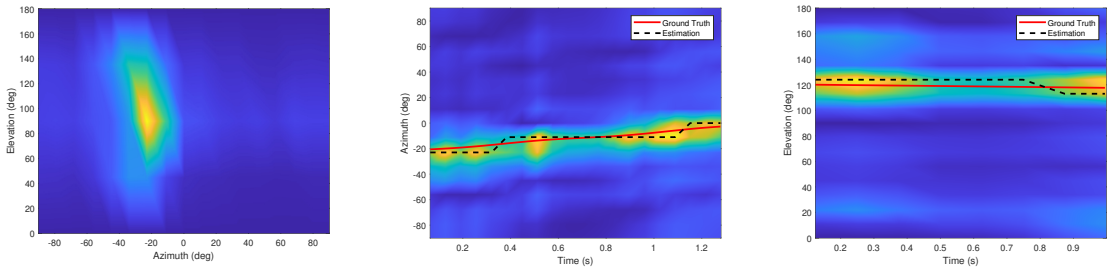
(a) Range-Doppler Map **(b)** Micro-Doppler **(c)** Range vs Time

Figure 15: Sensing plot output.



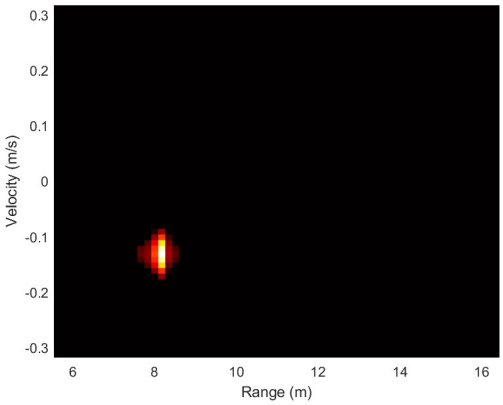
(a) Error probability of velocity **(b)** Error probability of range

Figure 16: Sensing plot output if ground truth is provided.

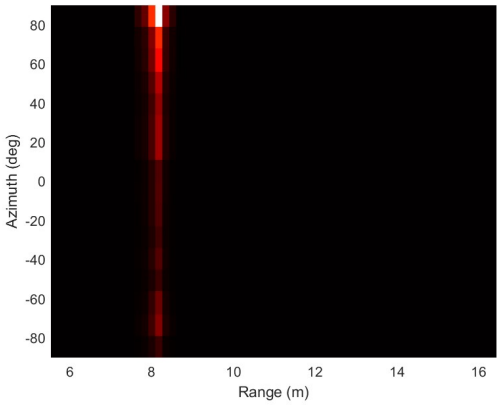


(a) Azimuth-Elevation Map **(b)** Azimuth Vs Time **(c)** Elevation vs Time

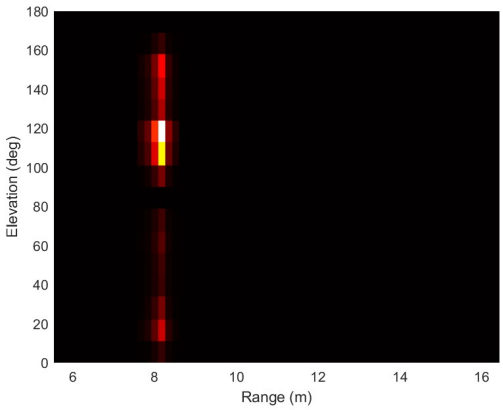
Figure 17: Sensing plot output when using active sensing.



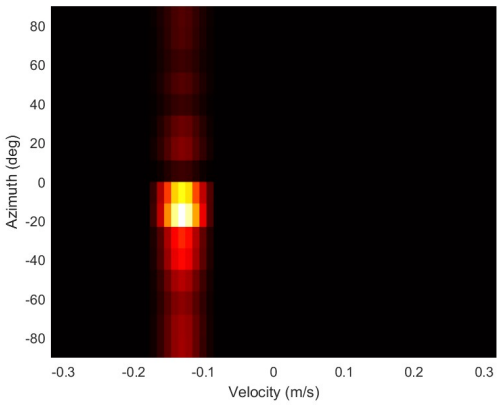
(a) Range-Doppler Map



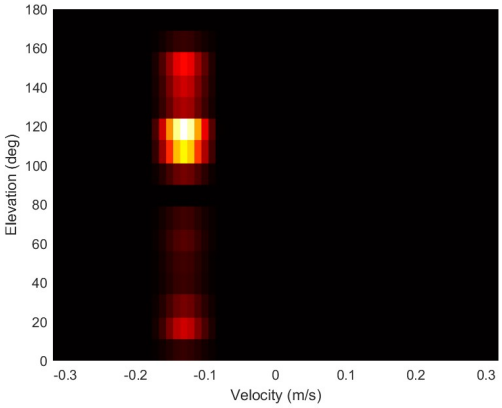
(b) Azimuth-Range Map



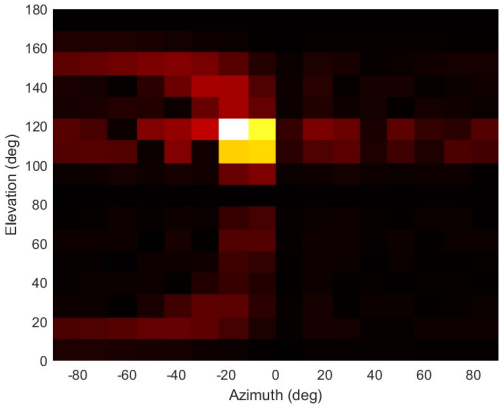
(c) Elevation-Range



(d) Azimuth-Velocity



(e) Elevation-Velocity



(f) Elevation-Azimuth

Figure 18: Sensing plot setting `saveRda =1`.

5 Examples

The following example demonstrates how to use the tools provided in [3] to simulate an ISAC system.

Specifically, Section 5.1, shows how to use ISAC-PLM alongside the QD Channel Realization software. The steps for initially generating the channel data before running an ISAC simulation. Subsequent examples will concentrate on various configurations of the ISAC-PLM, providing you with the knowledge to tailor the simulation to your specific needs.

5.1 Point Target in Empty Room: Passive Sensing-PPDU

This example (`isac-plm\examples\pointTargetPassiveSensing`) starts with the configuration of the NIST QD channel realization software. A single dimensionless point target moving is a ray-traced between one transmitter and one receiver in a 3-D empty room. The ray-traced multipath components provide a deterministic channel model, which is specific for the transmitter and receiver location, the environment, and the motion of the target analyzed. The channel model is used as input configuration of ISAC-PLM to simulate a SISO-SC ISAC link.

The input configuration and the output results are already provided; however, in the following, a step-by-step tutorial is provided to re-design the example by using Matlab. The configuration of the software can also be obtained with other tools since the input parameters are stored in text files.

To follow these examples simply checkout both Q-D channel realization software (https://github.com/wigig-tools/qd-realization) and ISAC-PLM in `yourfolder\`.

The folder structure will be as follows:

- `yourfolder\qd-realization`
- `yourfolder\isac-plm`

5.1.1 Deterministic Channel Model

The Q-D channel configuration folder of the Point Target example is already provided in `qd-realization\src\examples\PointTarget`.

To re-create the Point Target example, generate a new input folder, `PointTargetTutorial`, in which the configuration files will be added:

```
   currentPath = pwd;
   exampleFolder = fullfile(currentPath,...
'\qd-realization\src\examples\PointTargetTutorial');
   inputFolder = ([exampleFolder , '\Input']);
   mkdir(inputFolder)
```

3-D Scenario Configuration

- Specify the indoor 3-D map in XML format.

  In the example folder `qd-realization\src\examples\PointTarget\Input` an empty living room of dimension $7 \times 7 \times 3\,\mathrm{m}$ is provided in `LivingRoom.xml`, however a custom map can be created. For a tutorial on how to create a custom map, please refer to [4], otherwise just paste the provided file.

- Specify the position of the nodes in `NodePositionX.dat`

  The SU-SISO link is composed by two nodes, one transmitter and one receiver. The transmitter (node 0) is placed at (-3.9,0,2.8) m while the receiver (node 1) is placed at (3,0,2.8) m.

  The configuration files can be written as:

```
%Generate transmitter
writematrix([-3.9,0,2.8],[inputFolder, filesep,...
'NodePosition0.dat'])
%Generate receiver
writematrix([3,0,2.8],[inputFolder, filesep,...
'NodePosition1.dat'])
```

- Specify the point target trajectory in `TargetBase0.dat`

  The target is moving over time, hence a list of 3-D coordinates need to be provided. In this example, a linear motion is used and it can be generated as:

```
x = linspace(-1, 0, 512)';
y = linspace(-1, 0, 512)';
z = linspace(1, 1, 512)';
writematrix([x,y,z], [inputFolder,'\TargetBase0.dat'])
```

- Specify the parameter configuration of the Q-D channel realization software in `paraCfgCurrent.txt`.

  The Q-D is configured to generate 256 channel realization in the indoor living room map provided. The channel is configured to have a single reflection order from the environment and the target rays are not interacting with the environment (no target ghost on the reflecting surfaces of the environment). The total motion is assumed to be 2 seconds. The integration with the ISAC-PLM is enabled by writing the QD channel realization software in JSON format and by returning the target rays.

```
%Environment file
cfg.ParameterName{1,1} = 'environmentFileName';
cfg.ParameterValue{1,1} = 'LivingRoom.xml';
%Number of channel realization to generate
cfg.ParameterName{2,1} = 'numberOfTimeDivisions';
cfg.ParameterValue{2,1} = 256;
%Environment reflection order
cfg.ParameterName{3,1} = 'totalNumberOfReflections';
cfg.ParameterValue{3,1} = 1;
%Target reflection order
cfg.ParameterName{4,1} = 'totalNumberOfReflectionsSens';
cfg.ParameterValue{4,1} = 0;
%Simulation time in seconds
cfg.ParameterName{5,1} = 'totalTimeDuration';
cfg.ParameterValue{5,1} = 2;
%Output file format
cfg.ParameterName{6,1} = 'outputFormat';
cfg.ParameterValue{6,1} = 'json';
%Store a dedicated target MPC file
cfg.ParameterName{7,1} = 'writeTRayOutput';
cfg.ParameterValue{7,1} = 1;
%Write phase of MPCs
cfg.ParameterName{8,1} = 'enablePhaseOutput';
cfg.ParameterValue{8,1} = 1;
%Write Configuration File
writetable(struct2table(cfg), ...
[inputFolder, '\paraCfgCurrent'], "FileType", 'text',...
'delimiter', '\t')
```

Ray tracing

To perform the ray tracing:

- Open `main.m` in `yourfolder\qd-realization\src`
- Set `scenarioNameStr = 'examples\PointTargetTutorial'`
- Run `main.m`

5.1.2 ISAC system and signal processing

The ISAC-PLM configuration example is provided in \isac-plm\examples\PointTarget.

To re-create the Point Target example, generate a new input folder in which the configuration files will be added:

```
   plmExampleFolder = fullfile(currentPath, ...
'\isac-plm\examples\PointTargetTutorial');
   inputFolder = ([plmExampleFolder , '\Input']);
   mkdir(inputFolder)
```

ISAC configuration

- Specify the PHY configuration in `phyConfig.txt`

  Configure a SC SU-SISO comminication link that uses a 16-QAM.

```
% Set SC mode
cfgPhy.ParameterName{1,1} = 'phyMode';
cfgPhy.ParameterValue{1,1} = 'SC';
% Set SISO
cfgPhy.ParameterName{2,1} = 'numSTSVec';
cfgPhy.ParameterValue{2,1} = 1;
% Set MCS
cfgPhy.ParameterName{3,1} = 'mcs';
cfgPhy.ParameterValue{3,1} = 12;
% Set Sensing Type
cfgPhy.ParameterName{4,1} = 'lenPsduByt';
cfgPhy.ParameterValue{4,1} = 4096;
% Set Sensing Type
cfgPhy.ParameterName{5,1} = 'sensingType';
cfgPhy.ParameterValue{5,1} = 'passive-ppdu';
% Write Configuration File
writetable(struct2table(cfgPhy), ...
[inputFolder, '\phyConfig'], "FileType", 'text',...
'delimiter', '\t')
```

- Specify the sensing configuration in `sensConfig.txt`.

  Configure the sensing processor to use range-doppler processing using an FFT length of 64 and for a CPI of 64. No STF overlap is used in this example and no data windowing.

```
% Set pulse repetition interval
cfgSens.ParameterName{1,1} = 'pri';
cfgSens.ParameterValue{1,1} = 0.0005;
% Set FFT-lenght for doppler processing
cfgSens.ParameterName{2,1} = 'dopplerFftLen';
cfgSens.ParameterValue{2,1} = 256;
% Set FFT-window for doppler processing
cfgSens.ParameterName{3,1} = 'window';
cfgSens.ParameterValue{3,1} = 'rect';
% Set FFT-window lenght
cfgSens.ParameterName{4,1} = 'windowLen';
cfgSens.ParameterValue{4,1} = 64;
% Set FFT-window overlap
cfgSens.ParameterName{5,1} = 'windowOverlap';
cfgSens.ParameterValue{5,1} = 0;
% Set CPI
cfgSens.ParameterName{6,1} = 'pulsesCpi';
cfgSens.ParameterValue{6,1} = 64;
% Write Configuration File
writetable(struct2table(cfgSens), ...
[inputFolder, '\sensConfig'], "FileType", 'text',...
'delimiter', '\t')
```

- Specify the simulation setting in `simulationConfig.txt`

  Set the simulator to transmit 256 complete packets (data+preamble) with an SNR of 20 dB.

```
% Use preamble in packet transmission
cfgSim.ParameterName{1,1} = 'psduMode';
cfgSim.ParameterValue{1,1} = 0;
% Define length of the simulation
cfgSim.ParameterName{2,1} = 'nTimeSamp';
cfgSim.ParameterValue{2,1} = 256;
% Define SNR
cfgSim.ParameterName{3,1} = 'snrRange';
cfgSim.ParameterValue{3,1} = 20;
% Write Configuration File
writetable(struct2table(cfgSim), ...
```

```
[inputFolder, '\simulationConfig'], "FileType", 'text',...
'delimiter', '\t')
```

- Channel configuration in `channelConfig.txt`

  Configure the ISAC-PLM to read the output of the Q-D channel realization software:

```
cfgChan.ParameterName{1,1} = 'chanModel';
cfgChan.ParameterValue{1,1} = 'sensing';
% Write Configuration File
writetable(struct2table(cfgChan), ...
[inputFolder, '\channelConfig'], "FileType", 'text',...
'delimiter', '\t')
```

- Copy Q-D output to ISAC-PLM input

  Copy the Q-D output (`qdOutput.json` and `qdTargetOutout.json`) to the ISAC-PLM input folder `isac-plm\examples\PointTargetTutorial\Input\qdChannel`.

ISAC simulation

To perform the isac simulation:

- Open `mainIsac.m` in `yourfolder\isac-plm\src`
- Set `scenarioNameStr = 'examples\PointTargetTutorial'`
- Run `mainIsac.m`

ISAC results

Beyond the output files stored in `\Output`, several figures are available in `\Output\-Sensing\Figures` for a quick interpretation of the sensing results. Figure 19 shows the range-Doppler map and the micro-Doppler. The range-Doppler map on the left, shows a single peak that correspond to the velocity and range of the target. The micro-Doppler on the right shows the variation of the detected velocity over time.

5.2 Point Target in Empty Room: Bistatic-TRN-R

In this example (`isac-plm\examples\pointTargetActiveSensing`) a single dimensionless point target is moving between one transmitter and one receiver in a 3-D empty room, as in the previous example. To simulate a directional ISAC link uses the

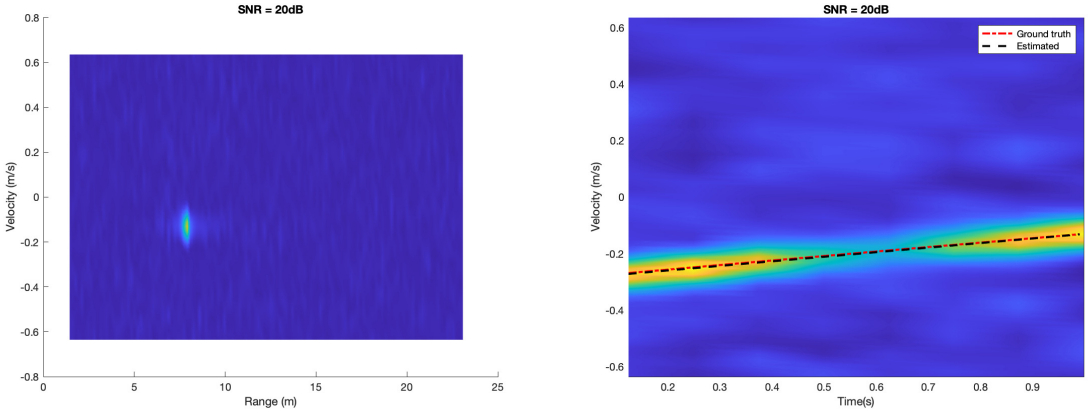


Figure 19: Point target sensing results.

TRN design of a multistatic PPDU proposed for IEEE 802.11bf. Assuming TRN-R, i.e., training at the receiver an 8×8 phased antenna array, while the transmitter uses an omni antenna, ISAC-PLM must be configured as follows.

Configuration of the channel input

Define the file `channelConfig.txt` specifying the channel model used.

- Specify the channel model: `chanModel sensing`

Define the folder `qdChannel` including the files:

- `qdOutput`. This file includes the channel MPCs
- `qdTargetOutput`. This file includes a single MPC per target, which is considered as the reference for computing target related information (such as, velocity, range and angle) and it is used to compute results and statistics of the designed system.

Configuration of the receiver antenna pattern

Define the file `nodePaa1.txt` specifying the antenna model among the models in `isac-plm\src\data`. In this example, an 8×8 phased antenna array is provided. This codebook includes 289 directions, a combination of 17 azimuth and 17 elevation directions. To configure the file `nodePaa1.txt`:

- Specify the codebook name: `Codebook cb8x8.txt`

TRN-R design to scan the codebook

Define the file `phyConfig.txt`, enabling the multistatic PPDU and specifying the design of the TRN-R. To configure the file `phyConfig.txt`:

- Enable the multistatic PPDU: `sensingType bistatic-trn`
- Specify the type of TRN: `packetType TRN-R`
- Specify the golay length: `subfieldSeqLength 128`.
- Define the training length to support 289 directions: `trainingLength 29`.

Doppler Processing Design

Define the file `sensConfig.txt` to use a coherent process interval of 32 packets. The Doppler FFT of length 64 is executed pre-filtering the data with a Blackman-Harris window and without the use of STFT. The pulse repetition interval in seconds, is dictated by the channel sampling rate (see previous section). To configure the file `sensConfig.txt`:

- Define PRI: `pri 0.0039`
- Set FFT window as the number of pulses per CPI: `windowLen 32`, `pulsesCpi 32`
- Set window overlap: `windowOverlap 0`.
- Specify Doppler FFT length: `dopplerFftLen 64`
- Specify the the pre-FFT window: `window blackmanharris`

Define Simulation parameters.

Define the length of the simulation in the number of channel samples. In this example, the maximum length allowed by the channel provided in the input is considered. The transmission is noise-free. The sensing plot are provided in output as well as the 2D images extracted from the 3D RDA map. To configure the file `simulationconfig.txt`:

- Specify the length of the simulation in the number of channel samples: `nTimeSamp 256`.
- Specify the SNR in dB: `snrRange 300`.
- Save sensing plot and 2D RDA images as: `sensPlot 1 saveRdaMap 1`

Output

Beyond the output files stored in `\Output`, several figures are available in `\Output\Sensing\Figures` for a quick interpretation of the sensing results.

- Figure 20 shows the Range-Doppler map evolution over the slow time. Since 256 packets have been transmitted and a CPI of 32 packets has been considered, $256/32 = 8$ RD maps are provided as subplots of the same figure, corresponding to each Doppler FFT.
- Figure 21 shows the velocity vs time and the range vs time maps. The estimated target information and the ground truth information are provided.
- Since ground truth information is provided, ISAC-PLM returns the performance of the designed system. Figure 22 shows the performance in terms of error probability of velocity and range.
- Since multi-static PPDU is used to sense multiple directions, ISAC-PLM returns angular information. Figure 23 shows the angle of arrival azimuth and elevation map over the slow time. 8 angular maps are provided as subplots of the same figure, corresponding to each Doppler FFT.
- Figure 24 shows the azimuth vs time and the elevation vs time maps. The estimated target information and the ground truth information are provided.
- Since ground truth information is provided, ISAC-PLM returns the performance of the designed system in the angular domain. Figure 25 shows the performance in terms of error probability of azimuth and elevation.

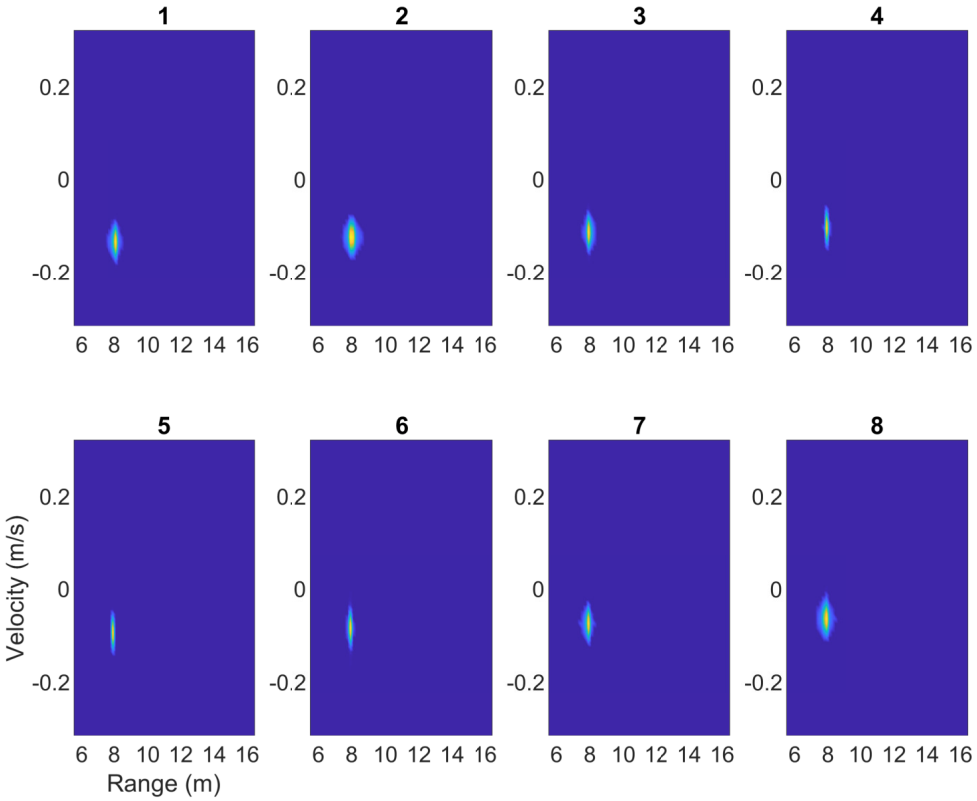


Figure 20: RD maps evolution over the slow time

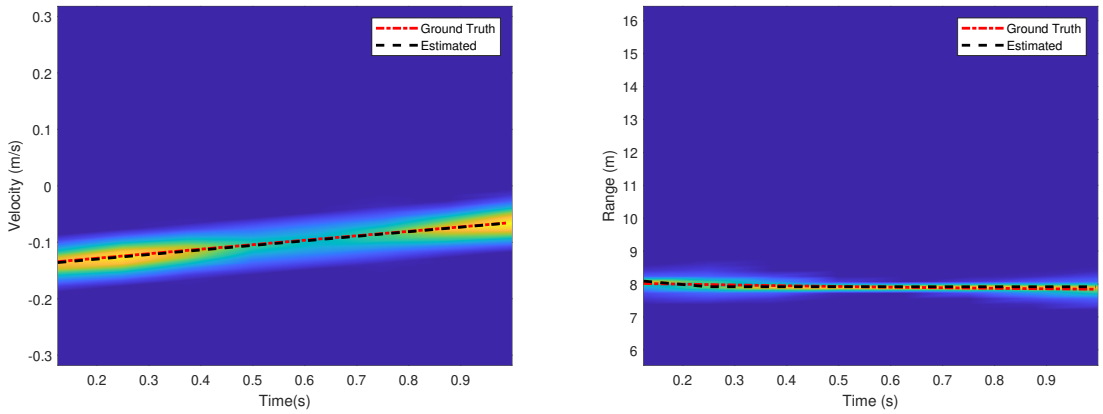


Figure 21: Velocity and range evolution over time.

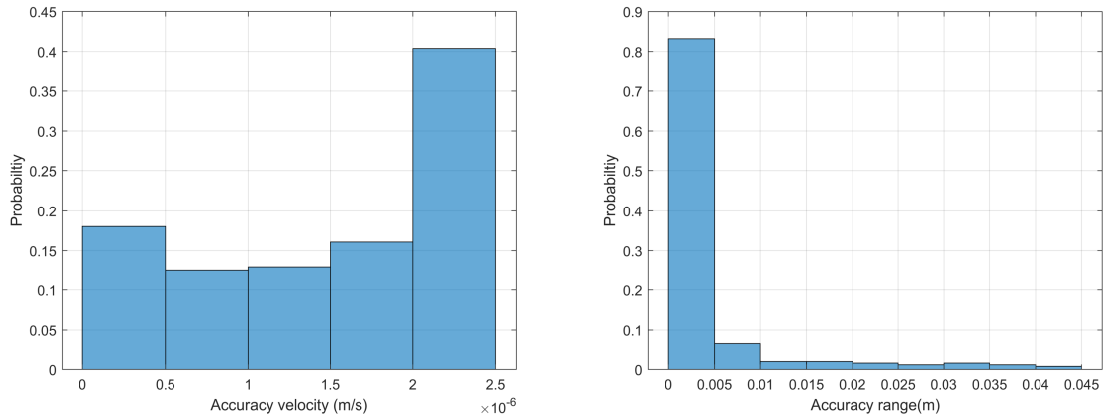


Figure 22: Sensing results in terms of error probability of velocity and range estimation.

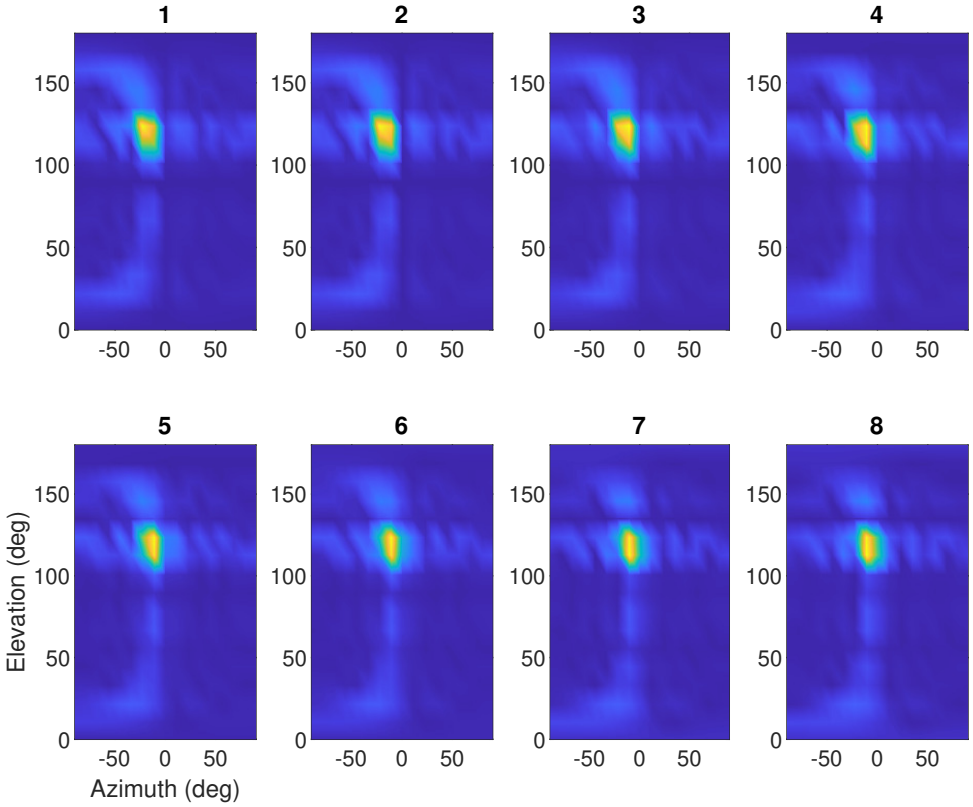


Figure 23: Azimuth-Elevation maps evolution over the slow time

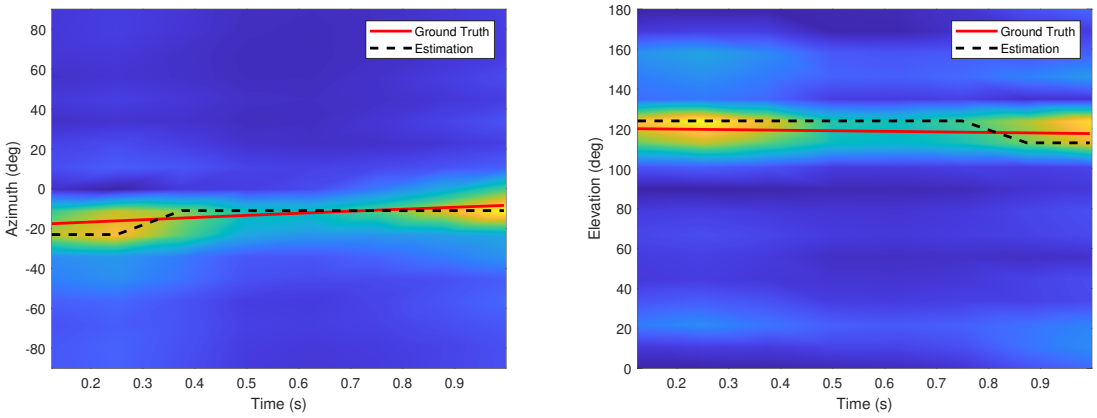


Figure 24: Azimuth/Elevation estimation over time.

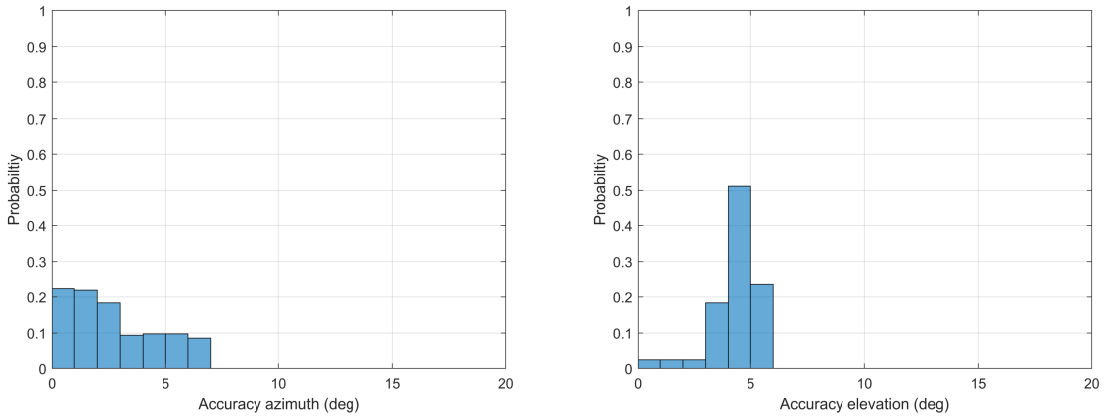


Figure 25: Sensing results in terms of error probability of azimuth and elevation estimation.

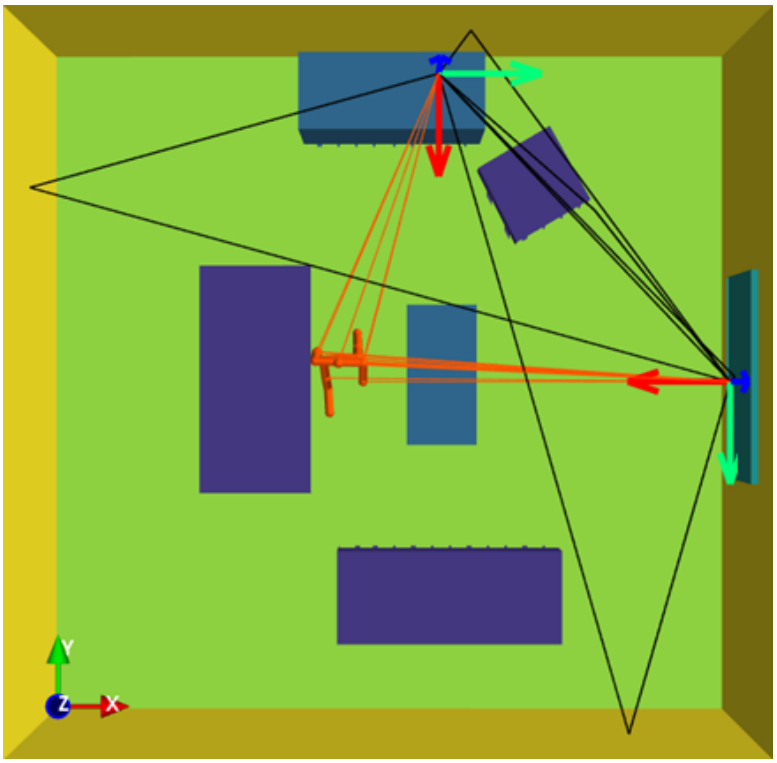


Figure 26: Human target in the living room.

5.3 Human Target in Living Room: Bistatic-TRN-T

In this example (`isac-plm\examples\bistaticLivingRoomTRN-T`) a human target composed by dimensionless points (boulic model [9]) is moving between one transmitter and one receiver in a living room, which includes furniture as in Figure 26. To simulate a directional ISAC link uses the TRN design of a multistatic PPDU proposed for IEEE 802.11bf. Assuming TRN-T, i.e., training at the transmitter a 2×8 phased antenna array, while the receiver uses an omni antenna, ISAC-PLM must be configured as follows.

Configuration of the channel input

As in Section 5.2.

Configuration of the transmitter antenna pattern

Define the file `nodePaa0.txt` specifying the antenna model among the models in `isac-plm\src\data`. In this example, a 2×8 phased antenna array is provided. This codebook includes 85 directions, a combination of 17 azimuth and 5 elevation directions. To configure the file `nodePaa0.txt`:

- Specify the codebook name: `Codebook cb2x8.txt`

TRN-T design to scan the codebook

Define the file `phyConfig.txt`, enabling the multistatic PPDU and specifying the design of the TRN-R. To configure the file `phyConfig.txt`:

- Enable the multistatic PPDU: `sensingType bistatic-trn`
- Specify the type of TRN: `packetType TRN-T`
- Specify the golay length: `subfieldSeqLength 128`.
- Define the training length to support 85 directions:
  - `unitP 2`.
  - `unitM 15`.
  - `unitN 1`.

Doppler Processing Design

As in Section 5.2.

Define Simulation parameters.

As in Section 5.2.

Output

Similar plots as in Section 5.2 are found in the output folder. As an example, Figure 27 shows the angle of departure azimuth estimation and the azimuth error probability.

5.4 Human Target in Living Room: Passive-Beacon

This example (`isac-plm\examples\bistaticLivingRoomBeacon`) reuses the same environment geometry and human motion presented in Section 5.3. The channel is obtained using the NIST Q-D channel realization software. The NIST Q-D channel realization software is configured to generate 13 channel realization for a simulation duration of 1.3 s, hence a beacon interval of 100 ms is considered.

Configuration of the channel input

As in Section 5.2.

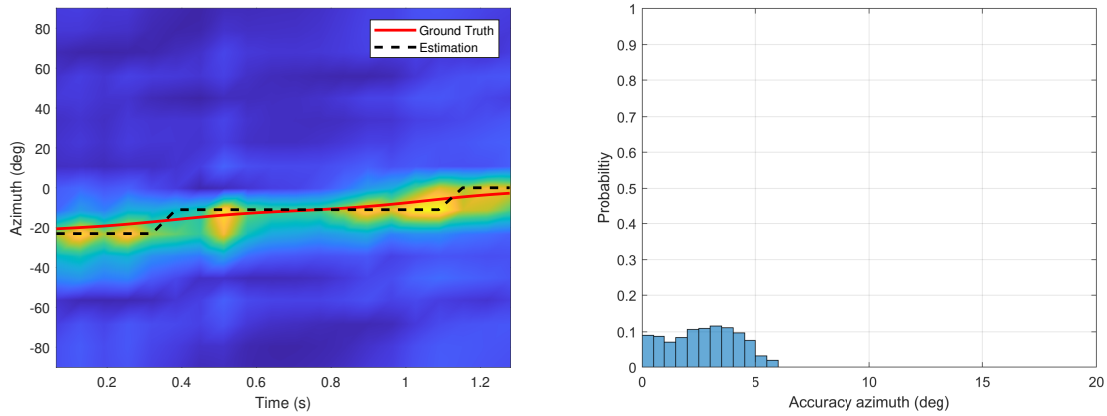


Figure 27: Human target in living room angle of departure azimuth estimation and azimuth error probability

Configuration of the transmitter antenna pattern

As in Section 5.3.

Doppler Processing Design

As Doppler processing is not used when transmitting a beacon, the file `sensConfig.txt` can be left empty.

Define Simulation parameters.

Define the length of the simulation in the number of channel samples. In this example, the maximum length allowed by the channel provided in the input is considered. The SNR is 25 dB, which is achieved when the beacon is transmitted towards the direction of the receiver. To configure the file `simulationconfig.txt`:

- Specify the length of the simulation in the number of channel samples: `nTimeSamp 13`.
- Specify the SNR in dB: `snrRange 25`.
- Store the CSI in output: `saveCSI 1`

Define PHY parameters

Define the file `phyConfig.txt`, enabling the transmission of the beacon.

- Enable the beacon frame: `sensingType passive-beacon`

Output

The following files are returned:

- `sensingInfo.json`: contains the values of azimuth and elevation of each sector.
- `sensingResults.json`: contains the SNR per sector and per measurement instance.
- `csi.json`: contains the channel estimates per sector and per measurement instance.

5.5 Threshold-based Sensing Measurement and Reporting

In this example (`isac-plm\examples\thresholdSensing`) a human target composed by dimensionless points is considered. To demonstrate the impact of threshold-based sensing measurement and reporting, we considered a transmitter and receiver pair with omni-directional antenna in a living room. ISAC-PLM is used in passive sensing mode. ISAC-PLM must be configured as follows.

Configuration of the channel input

As in Section 5.2.

Doppler Processing Design

Define the file `sensConfig.txt` to use a coherent process interval of 64 packets. The Doppler FFT of length 64 is executed by pre-filtering the data with a Hamming window and with the use of STFT. The STFT is designed with a window length of 16 pulses and a window overlap of 10%, (parameters shown in Fig. 12). The pulse repetition interval in seconds, is dictated by the channel sampling rate (see the previous section). To configure the file `sensConfig.txt`:

- Define PRI: `pri 0.0005`
- Set FFT window length: `windowLen 16`,
- Set the number of pulses per CPI: `pulsesCpi 64`
- Set window overlap: `windowOverlap 0.1`.
- Specify Doppler FFT length: `dopplerFftLen 64`
- Specify the pre-FFT window: `window hamming`

Threshold Based Sensing Design

In this example we quantify the CSI variation using the TRRS metric. The AP receives as feeback measurements with irregular sampling, thus the AP uses linear interpolation to reconstruct measurements with regular sampling. The file `sensConfig.txt` includes the following fields:

- Specify the thresholdSensing: `thresholdSensing 1`
- Specify the threshold: `threshold 0.001`
- Specify the csiVariationScheme: `csiVariationScheme TRRS`
- Specify the interpolationScheme: `interpolationScheme linearInterpolation`

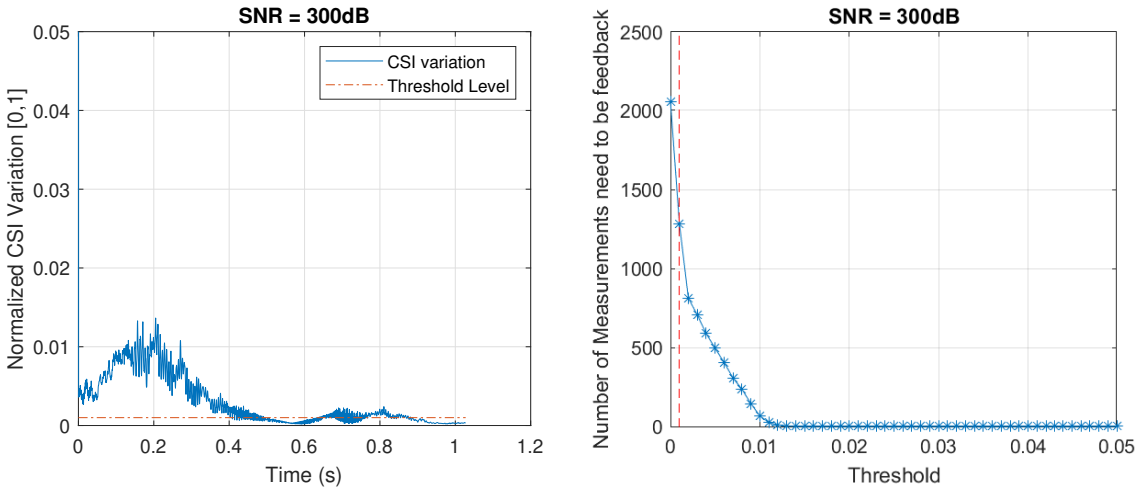
Define Simulation parameters.

As in Section 5.2.

Output

As output, besides the results stored in `sensInfo.json` the following figures are available in `Output\Sensing\Figures`:

- Figure 28a shows the Normalized CSI variation vs time. For reference also the set threshold is visualized. The sensing measurements with a normalized CSI variation above the threshold are reported.
- Figure 28b shows the number of measurements reported varying the threshold.



(a) Normalized CSI variation versus time

(b) Number of measurements need to be feedback from the receiver

Figure 28: Treshold-bases sensing measurement and reporting.

5.6 Channel Measurements: vital sign sensing

The example `isac-plm\examples\vitalsSensing` uses the channel obtained using NIST measurements. NIST carried out a measurement campaign with a mmWave phased-antenna array channel sounder [10, 11]. The channel sounder operates at a center frequency of 28.5 GHz. The system transmits at 2 Gb/s. The measurement campaign adopts a quasi-mono-static deployment, and the transmitter has a phased antenna array with 64 elements, while the receiver employs a phased antenna array with 256 elements. In the measurement campaign, the channel was sampled at a time interval of 26 ms. A human target is sitting at 2.5 m away from the assembly containing the transmitter and receiver. To extract the MPCs from the received signal, the CLEAN algorithm was combined with a least square power estimation technique [12]. The MPC are gated in delay to remove the background, hence only the MPCs coming from the target are used, and no further clutter removal is used. The MPCs are used as input of ISAC-PLM to extract the microdoppler generated by the motion produced by the movement of the chest and heart.

Configuration of the channel input

Define the file `channelConfig.txt` specifying the channel model used.

- Specify the channel model: `chanModel sensing`

Define the folder `qdChannel` including the files:

- `qdOutput`. This file includes the channel MPCs

Note that the file `qdTargetOutput` is not defined since using real measurements, a reference MPC cannot be described. For this reason, performance metrics related to accuracy are not generated

Configuration of the receiver antenna pattern

In this example, an omni antenna is used, so no antenna model is required.

PHY design

The file `phyConfig.txt` can be used to design the dedicated sensing pilots. In this example we use TRN-R. Since there is no antenna, beam scanning is unnecessary, and trainingLength can be set to 1. To configure the file `phyConfig.txt`:

- Enable the multistatic PPDU: `sensingType bistatic-trn`
- Specify the type of TRN: `packetType TRN-R`
- Specify the golay length: `subfieldSeqLength 128`.

Doppler Processing Design

In the configuration file named `sensConfig.txt`, adjust the Short-Time Fourier Transform (STFT) parameters to use a window length of 32 and a window overlap of 90%. Set the pulses in a CPI as the number of measurements, to allow the STFT window to traverse the entire signal seamlessly, ignoring the boundaries of a CPI.

For the Doppler processing, an FFT of length 256 is used. Preprocess the signal with a Hamming window. The PRI, which determines the temporal spacing between consecutive pulses, is set based on the channel's sampling rate of the measurement campaign, amounting to 0.026 milliseconds.

The range-doppler maps are filtered using a 2D Constant False Alarm Rate (CFAR) detection algorithm.

To configure the file `sensConfig.txt`:

- Define PRI: `pri 0.026`
- Set FFT window as: `windowLen 32`
- Set the CPI as the number of channel samples (see next section) as: `pulsesCpi 900`
- Set window overlap: `windowOverlap 0.9`.
- Specify Doppler FFT length: `dopplerFftLen 256`
- Specify the the pre-FFT window: `window hamming`
- To design the CFAR filter:
  - Number of guard cells in the range dimension `cfarGrdCellRange`: 0
  - Number of guard cells in the velocity or Doppler dimension `cfarGrdCellVelocity`: 30
  - Number of training cells in the range dimension `cfarTrnCellRange`: 0
  - Number of training cells in the velocity or Doppler dimension `cfarTrnCellVelocity`: 36
  - Threshold level for detection in dB `cfarThreshold`: 6
- Set `clutterRemovalMethod` as `'none'`.

Define Simulation parameters.

Configure the duration of the simulation by specifying the total number of channel samples. In this particular example, the transmission runs over 900 channel samples, which corresponds to a transmission period of 23.4 seconds. The SNR is configured to be 30 dB.

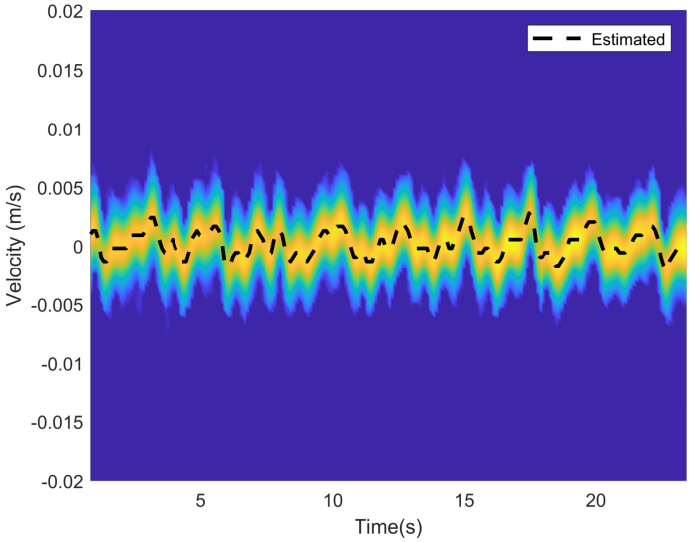


Figure 29: Microdoppler.

Upon completion, 2D images are generated from the 3D RDA map.
To configure the file `simulationconfig.txt`:

- Specify the length of the simulation in the number of channel samples: `nTimeSamp 900`.
- Specify the SNR in dB: `snrRange 30`.
- Save sensing plot and 2D RDA images as: `sensPlot 1 saveRdaMap 1`

Output

The following files are returned:

- `sensingInfo.json`.
- `sensingResults.json`.
- `rda.json`.
- `targetEstimation.json`.

Several plots including RD maps and microdoppler are inside `Output\Sensing\figures`. Figure 29 shows the microdoppler signatures obtained. The target is detected and the velocity is estimated.

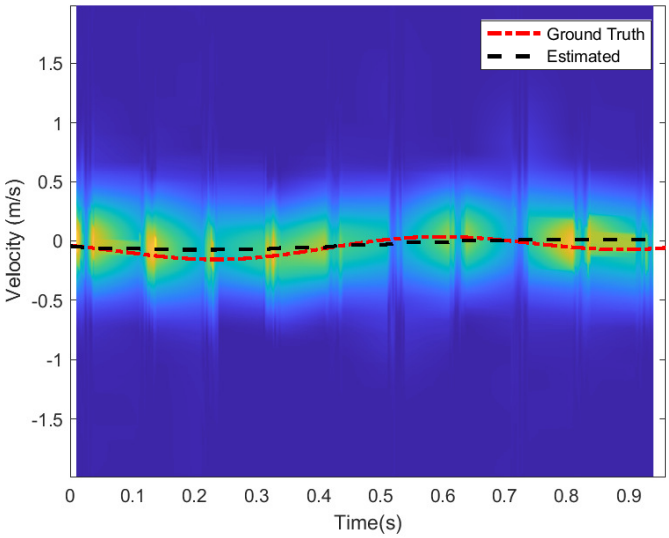


Figure 30: Microdoppler with Packet Loss.

5.7 Lossy and Jittery Conditions

This scenario extends ISAC-PLM to evaluate IEEE 802.11bf-style sensing when CSI arrives irregularly (jitter) and with packet loss. Two new sensing parameters must be configured in `sensConfig.txt`:

- `interBI` (inter-burst interval): time between consecutive DMG sensing bursts.
- `sparseCsi`: when set to 1, the simulator consumes a timing file (`sensingTiming.csv`) that encodes real/irregular measurement times and losses; when 0, the simulator uses an ideal, uniform schedule. In sparse mode the timing file is auto-loaded during configSens.

We evaluate sensing under a lossy acquisition schedule. The nominal timing parameters are:

`PRI` $= 0.000625\,\text{s}$ $(625\,\mu\text{s})$, `pulsesCpi` $= 64$, `interBI` $= 0.1\,\text{s}$, `sparseCsi` $= 1$.

With `sparseCsi`=1 the simulator uses the irregular arrivals from `Input/sensingTiming.csv` instead of the ideal uniform grid.

5.7.1 Packet Loss

The example `isac-plm\examples\loss_0.6_intraStd_0us_interStd_0ms` shows the effect of packet losses, without jitter.

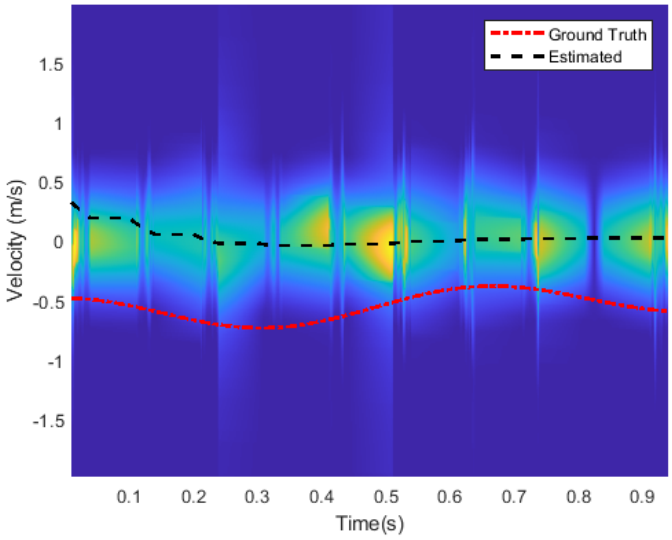


Figure 31: Microdoppler with Packet Loss and Intra-Burst Jitter.

Timing profile. The timing file was generated with the following parameters, (i) packet *loss probability* $p_{\text{loss}} = 0.6$ on intra-burst exchanges, (ii) *zero intra-burst jitter* $\sigma_{\text{intra}} = 0$ μs, (iii) *zero inter-burst jitter* $\sigma_{\text{inter}} = 0$ ms. Consequently, the surviving timestamps within each burst lie on the nominal PRI grid, while *loss* manifests as missing instances. With 64 opportunities per burst and 40% retention, each burst carries ≈ 26 measurements on average; the provided file contains 10 bursts and 258 total measurements (≈ 25.8 per burst). The microdoppler generated is shown in Figure 30.

5.7.2 Packet Loss with Intra-Burst Jitter

The example `isac-plm\examples\loss_0.8_intraStd_500us_interStd_0ms` stresses the pipeline with *heavy* loss and *heavy* intra-burst jitter while keeping burst anchors periodic.

Timing profile. The timing file was generated with the following parameter, (i) packet loss probability $p_{\text{loss}} = 0.8$ on intra-burst exchanges, (ii) intra-burst jitter with standard deviation $\sigma_{\text{intra}} = 500$ μs, (iii) *inter-burst jitter* $\sigma_{\text{inter}} = 0$ ms. With the nominal grid `PRI` $= 625$ μs, `pulsesCpi` $= 64$, `interBI` $= 0.1$ s, 500 μs corresponds to $\approx 0.8 \cdot \text{PRI}$, making the received samples within each burst highly non-uniform. With 64 opportunities per burst and 20% retention, each burst carries ≈ 12 measurements on average; the provided file contains 10 bursts and 121 total measurements (≈ 12.1 per burst). The microdoppler generated is shown in Figure 31.

Appendix A

List of Input Parameters

Table A.1: Tunable properties of `simulation` in `simulationConfig.txt`.

Parameter	Description	Data Type	Valid Range	Default
`debugFlag`	Enables debug mode	double	$\{0, 1\}$	0
`psduMode`	Packet type	double	$\{0, 1\}$	1, 0*
`dopplerFlag`	Enables Doppler effect	double	$\{0, 1\}$	0
`snrMode`	SNR definition	char — string	'EsNo', 'EbNo'	'EsNo'
`snrAntNormFlag`	SNR normalization	double	$\{0, 1\}$	0
`snrRange`	SNR range	1-by-2 double	Positive Values	[0 20]
`snrStep`	SNR step	double	Positive integer	1
`maxNumErrors`	Max number of errors	double	Positive integer	100
`maxNumPackets`	Max number of packets	double	Positive integer	1000
`nTimeSamp`*	Channel time samples	double	Positive integer	32
`sensPlot`*	Plot range doppler	double	$\{0, 1\}$	0
`saveRdaMap`*	Save RDA jpeg	double	$\{0, 1\}$	0
`saveCsi`*	Save csi	double	$\{0, 1\}$	0

*ISAC

Table A.2: Tunable properties of `phy` in `phyConfig.txt`.

Parameter	Description	Data Type	Valid Range	Default
`phyMode`	PHY mode	char — string	'OFDM', 'SC'	'OFDM'
`lenPsduByt`	Length of PSDU.	double	positive scalar	4096
`giType`	Guard Interval Type	char — string	'Short', 'Normal,'Long'	'Long'
`numSTSVec`	Num. spatial-time streams	double	Integer in [1,8]	1
`smTypeNDP`	Spatial Mapping Pilots	char — string	'Hadamard', 'Fourier, 'Custom ', 'Direct'	'Direct'
`smTypeDP`	Spatial Mapping Data	char — string	'Hadamard', 'Fourier, 'Custom ', 'Direct'	'Direct'
`mcs`	Modulation-Coding Index	double	Integer in [0,20-21]	6
`processFlag`	Transceiver mode	double	Integer in [0,5]	1
`symbOffset`	Symbol sync offset	double	Real in [0,1]	0.75
`softCsiFlag`	Soft CSI	double	$\{0, 1\}$	0 (SC), 1 (OFDM)
`ldpcDecMethod`	LDPC decoding method	char — string	'norm-min-sum', 'bp'	'norm-min-sum'
`sensingType`	DMG Sensing Type	char-string	'passive-beacon','passive-ppdu', 'bistatic-trn', 'none'	'none'
`packetType`	Type of TRN in MS-PPDU	char — string	'TRN-R', 'TRN-T'	'TRN-R'
`trainingLength`	EDMG TRN Length	Double	Integer in [0-255]	30
`subfieldSeqLength`	TRN Subfield Sequence Length	Double	128,256, 64	128
`unitP`	EDMG TRN-Unit P	Double	0,1,2,4	2
`unitM`	EDMG TRN-Unit M	Double	Integer in [0-15]	1

`unitN`	EDMG TRN-Unit N	Double	Integer in [1 2 3 4 8]	1
`UnitRxPerUnitTx`	RX TRN-Units per Each TX TRN-Unit	Double	Integer in [0-255]	54

Table A.3: Tunable properties of `sens` in `sensConfig.txt`.

Parameter	Description	Data Type	Valid Range	Default
`pri`	Pulse Repetition Interval	double	positive scalar	0.0005
`dopplerFftLen`	FFT length	double	positive integer	64
`pulsesCpi`	Coherent Pulses	double	positive integer	64
`window`	FFT window	double	positive integer	64
`windowLen`	FFT window length	double	positive integer	64
`windowOverlap`	FFT window overlap	double	Real in [0,1]	0
`cfarGrdCellRange`	Range Guard Cells	double	positive integer	64
`cfarGrdCellVelocity`	Velocity Guard Cells	double	positive integer	64
`cfarTrnCellRange`	Range Training Cells	double	positive integer	64
`cfarTrnCellVelocity`	Range Training Cells	double	positive integer	64
`cfarThreshold`	CFAR Detection Threshold	double	positive integer	64
`clutterRemovalMethod`	Clutter removal	char-string	'remove_ static _component', 'recursive_ first_ order_hp', 'none'	'remove_ static _component'
`thresholdSensing`*	Threshold-based sensing	double	Real in [0,1]	0
`adaptiveThreshold`*	Adaptive threshold	double	Real in [0,1]	0
`numTimeDivisions`*	Number of time-divisions	double	positive integer	5
`threshold`*	Threshold	double	Real in [0,1]	0
`stepThreshold`*	threshold step size	double	Real in [0,0.5]	0.1
`percentMeasurement`*	percent of measurements	double	Real in [0,100]	90

`csiVariationScheme`*	CSI variation scheme	char-string	EucDistance, TRRS, FRRS	TRRS
`interpolationScheme`*	Interpolation scheme	double	previous-Measuremment, linearInterpolation, zeroPadding	previous-Measuremment

*Optional for threshold based sensing.

Table A.4: Tunable properties of `chan` in `channelConfig.txt` if `chanModel = Rayleigh`.

Parameter	Description	Data Type	Valid Range	Default
`numTaps`	Number of taps	double	positive scalar	10
`pdpMethodStr`	Power delay profile	char — string	`PS`, `Equ`, `Exp`	`Exp`
`tdlTypeChannel`	Interpolation	char — string	`Impulse`, `Sinc`	`Impulse`

Table A.5: Tunable properties of `chan` in `channelConfig.txt` if `chanModel = NIST`.

Parameter	Description	Data Type	Valid Range	Default
`environmentFile-NameChannel`	Environment	char — string	`LR`, `SC`, `OAH`	`LR`
`totalNumber-OfReflection`	Reflection order	double	positive integer	2
`tdlTypeChannel`	Interpolation	char — string	`Impulse`, `Sinc`	`Impulse`

Table A.6: Tunable properties in `paaConfigNodeX.txt` if `chanModel = NIST`.

Parameter	Description	Data Type	Valid Range	Default
`numAntenna`	Number of antennas	double	positive integer	16
`Geometry`	Geometry	char — string	`UniformLinear` `Uniform-Rectangular`	`Uniform-Rectangular`
`numAntennaVert`	Number of vertical antennas	double	positive integer	4

Table A.7: Tunable properties of `chan` in `channelConfig.txt` if `chanModel = sensing`.

Parameter	Description	Data Type	Valid Range	Default
`Normalization`	Channel Normalization	char — string	'packet', 'none'	'packet'

References

[1] *Part 11: Wireless LAN (MAC) and (PHY) Specifications–Amendment 2: Enhanced Throughput for Operation in License-Exempt Bands Above 45 GHz*, IEEE Std. IEEE P802.11ay/D7.0, 2020.

[2] *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications–Amendment 2: Enhancements for Wireless LAN Sensing*, IEEE Std. IEEE P802.11bf/D0.1, 2022.

[3] H. Assasa *et al.* A Collection of Open-source Tools to Simulate IEEE 802.11ad/ay WLAN Networks in Network Simulator ns-3. [Online]. Available: https://github.com/wigig-tools

[4] A. Bodi, S. Blandino, N. Varshney, J. Zhang, T. Ropitault, M. Lecci, P. Testolina, J. Wang, C. Lai, and C. Gentile, "A quasi-deterministic (Q-D) channel implementation in MATLAB software," https://github.com/wigig-tools/qd-realization, 2021.

[5] J. Zhang, S. Blandino, N. Varshney, J. Wang, C. Gentile, and N. Golmie, "Multi-user MIMO enabled virtual reality in IEEE 802.11ay WLAN," in *2022 IEEE Wireless Communications and Networking Conference (WCNC)*, 2022, pp. 2595–2600.

[6] S. Blandino, T. Ropitault, A. Sahoo, and N. Golmie, "Tools, models and dataset for IEEE 802.11ay CSI-based sensing," in *2022 IEEE Wireless Communications and Networking Conference (WCNC)*, 2022, pp. 662–667.

[7] S. Blandino, T. Ropitault, C. R. C. M. da Silva, A. Sahoo, and N. Golmie, "IEEE 802.11bf DMG sensing: Enabling high-resolution mmWave Wi-Fi sensing," *IEEE Open Journal of Vehicular Technology*, vol. 4, pp. 342–355, 2023.

[8] T. Koike-Akino, P. Wang, M. Pajovic, H. Sun, and P. V. Orlik, "Fingerprinting-based indoor localization with commercial MMWave WiFi: A deep learning approach," *IEEE Access*, vol. 8, pp. 84 879–84 892, 2020.

[9] R. Boulic, N. M. Thalmann, and D. Thalmann, "A global human walking model with real-time kinematic personification," *The visual computer*, vol. 6, no. 6, pp. 344–358, 1990.

[10] C. Gentile, J. Senic, A. Bodi, S. Berwerger, and N. Golmie, "Context-aware channel sounder for AI-assisted radio-frequency channel modeling," in *2024 18th European Conference on Antennas and Propagation (EuCAP)*, 2024, pp. 1–5.

[11] D. Caudill, J. Chuang, S. Y. Jun, C. Gentile, and N. Golmie, "Real-time mmWave channel sounding through switched beamforming with 3-D dual-polarized phased-array antennas," *IEEE Transactions on Microwave Theory and Techniques*, vol. 69, no. 11, pp. 5021–5032, 2021.

[12] J. Högbom, "Aperture synthesis with a non-regular distribution of interferometer baselines," *Astronomy and Astrophysics Supplement, Vol. 15, p. 417*, vol. 15, p. 417, 1974.