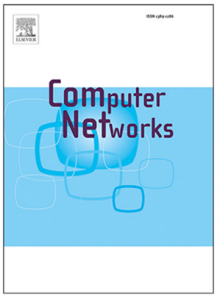
Journal Pre-proof



O-JRC: An open source software platform for mmWave joint radar-communication development and experimentation

Xin Liu, Haocheng Zhu, Eylem Ekici

PII: S1389-1286(25)00304-4
DOI: https://doi.org/10.1016/j.comnet.2025.111337
Reference: COMPNW 111337

To appear in: *Computer Networks*

Received date : 29 January 2025
Revised date : 14 April 2025
Accepted date : 22 April 2025

Please cite this article as: X. Liu, H. Zhu and E. Ekici, O-JRC: An open source software platform for mmWave joint radar-communication development and experimentation, *Computer Networks* (2025), doi: https://doi.org/10.1016/j.comnet.2025.111337.

O-JRC: An Open Source Software Platform for mmWave Joint Radar-Communication Development and Experimentation*

Xin Liu[a], Haocheng Zhu[b], Eylem Ekici[b,*]

[a]*Department of Computer Science, Florida State University, Tallahassee, FL, USA*
[b]*Department of Electrical and Computer Engineering, The Ohio State University, Columbus, OH, USA*

Abstract

Integrated Sensing and Communication (ISAC) systems unify sensing and communication functionalities on a single platform, opening avenues for innovative solutions in the mmWave spectrum. Joint Radar-Communication (JRC) represents one promising approach to realizing ISAC by integrating radar and communication functionalities on a single platform. The development of ISAC systems demands flexibility in both hardware and software to accommodate diverse experimental needs. However, existing Software-Defined Radio (SDR)-based platforms for ISAC often face limitations stemming from rigid hardware configurations and algorithmic constraints tied to SDR architectures. In this work, we present an open-source ISAC software platform, O-JRC, specifically designed to enable efficient development of experimental ISAC systems and validation of advanced algorithms under

*This research is supported by National Science Foundation (NSF) under Grant(1955535, 2030141, 2112471)

*Corresponding author

Email addresses: xinliu@cs.fsu.edu (Xin Liu), zhu.2991@osu.edu (Haocheng Zhu), ekici.2@osu.edu (Eylem Ekici)

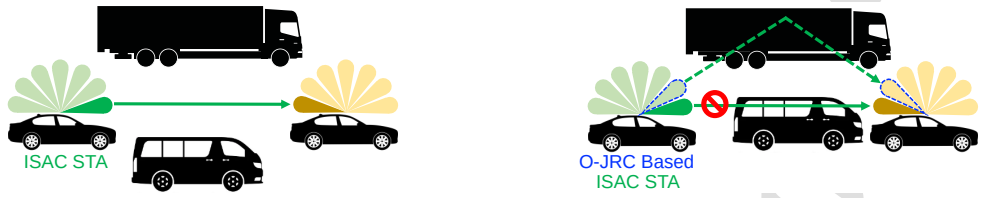
complex scenarios. A core feature of O-JRC is its layered and modular architecture, which disaggregates control logic from signal processing, facilitating seamless integration of advanced control algorithms developed in efficient programming languages. This modularity enhances development flexibility, enabling independent testing of various configurations without requiring code modifications, while also simplifying the evaluation of diverse algorithms. To demonstrate O-JRC's versatility, we implemented and tested two fundamentally different machine learning algorithms on a fully-digital 4x2 MIMO ISAC experimental platform operating at 24 GHz with a 200 MHz bandwidth: a Convolutional Neural Network (CNN)-based control algorithm and a Multi-Armed Bandit (MAB)-based reinforcement learning algorithm. These implementations highlight O-JRC's capability to support the development and experimentation of a wide range of control strategies. Comprehensive testing validated O-JRC's performance, underscoring its potential to drive innovation in the ISAC field [1].

Keywords: mmWave Integrated Sensing and Communication, Joint Radar-Communication, Open-source Software Architecture, Customized Beamforming Control

1. Introduction

mmWave technology is an integral part of next generation wireless networks, offering higher spectral efficiency and greater data transmission capacity. In these systems, the demand for comprehensive real-time contextual awareness is significantly amplified, especially in high-mobility scenarios. In

[1] The source code is available at https://github.com/mmWave-MIMO-Testbed/O-JRC



(a) Baseline LOS (b) LOS Disruption

Figure 1: Example high mobility scenario for beam agility.

this context, mmWave-based Joint Radar-Communication (JRC) systems emerge as a promising approach to Integrated Sensing and Communication (ISAC), offering platforms that deliver high-rate communication as well as radar sensing capabilities. Through the joint use of sensing and communication functions, ISAC systems can achieve high spectrum efficiency, better latency, and lower power requirements. To this end, experimental ISAC systems have been developed [1, 2, 3] based on SDRs.

Implementing communication and radar functions on the same platform, some existing experimental ISAC systems utilize radar sensing to support communication functions for beamforming [2, 1, 3]. The *situational awareness* required to adapt to rapidly changing environments is readily available through radar sensing and can be exploited through appropriate control algorithms. For instance, Fig.1a illustrates a baseline Line-of-Sight (LOS) scenario where a ISAC station effectively pinpoints the location of a target vehicle using its radar capabilities. In Fig.1b, another vehicle blocks the LOS link. The ISAC station can detect the presence of the blocker and identify the large reflective surface via radar sensing, and establish a Non-LOS link to the receiver.

Implementing advanced functionalities such as blockage prediction and

proactive beam steering require more advanced control algorithms. However, our analysis of the development complexity unveils multiple underlying challenges: 1) Existing experimental platforms predominantly rely on SDRs [1, 2, 3], which emphasize signal processing routines coded in C++ for real-time operation. However, implementing control algorithms such as Reinforcement Learning-based decision-making in C++ can be particularly complex, leading to convoluted code structures and increased development time. 2) Many SDR platforms offer Python scripting to facilitate more straightforward development. However, these scripts typically require the inclusion of complex libraries and interfaces derived from the SDR modules. Hence, even when using a language like Python, the development of control algorithms still requires strong dependencies on module structures and configurations derived from them. 3) Due to strong module dependencies, even minor changes to the hardware configuration have significant impact on communication routines, leading to the generation of a cascade of new scripts and restructuring of dependencies. These challenges are especially pronounced for experimental JRC systems on which algorithmic solutions are developed and tested.

In this paper, we present O-JRC, an Open-source Joint Radar Communication software platform designed for efficient algorithm development and validation for diverse scenarios. Fig. 2 provides an architectural overview of O-JRC, reflecting the key design principle of disaggregating control logic from signal processing via a layered and modular architecture. The core design of O-JRC is centered around a complete isolation between communication and control layers, with data exchange through an intermediary data

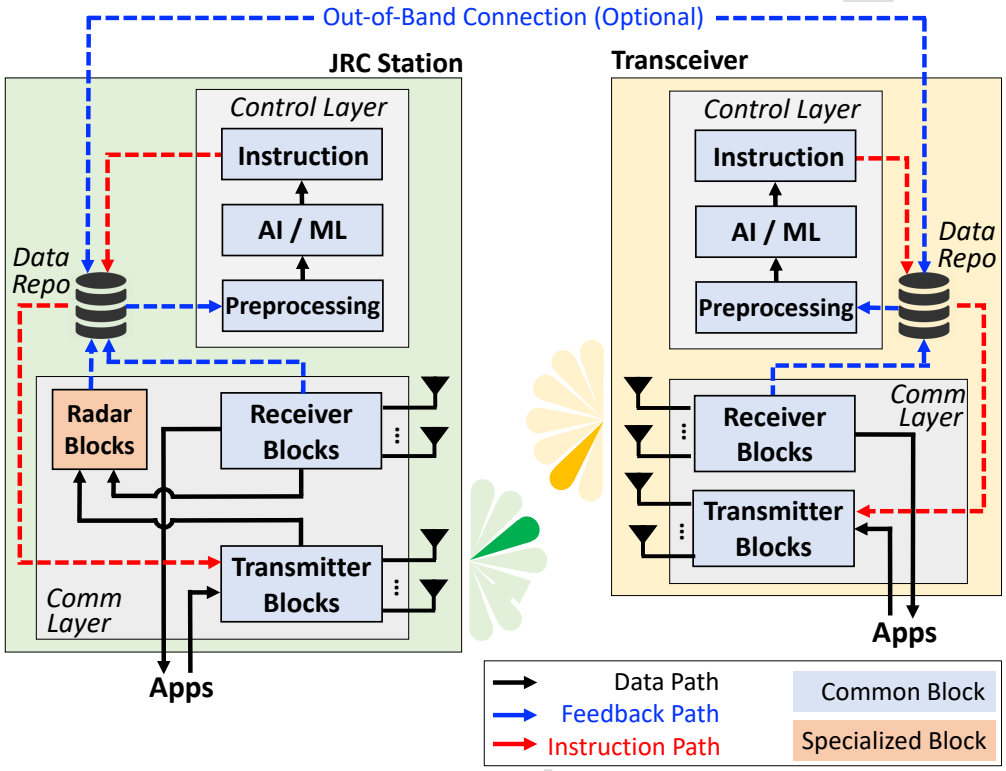


Figure 2: Architectural Overview of O-JRC Platform

repository. The communication layer is responsible for all communication functionalities, while the control layer handles all control logic. This repository serves as the sole conduit for data exchange between communication and control layers. The communication layer populates the repository with real-time feedback such as the target's angle, range, speed, and communication performance metrics. Then, the control layer processes this feedback using AI/ML (Machine Learning) techniques or other advanced algorithms for real-time decision-making, and deposits the derived instructions back into the repository, enabling flexibility to incorporate a variety of control strategies beyond ML-based approaches. The communication layer then retrieves

and executes these instructions in real-time. This process ensures seamless interactions between the layers while maintaining isolation, reducing the development complexity by eliminating interwoven dependencies between control and communication functions.

Each layer in O-JRC is composed of distinct blocks: the communication layer encompasses transmitter, receiver, and radar blocks, while the control layer comprises preprocessing, AI/ML, and instruction blocks. Blocks within the same layer are interoperable across both the JRC station and the transceiver, which correspond to different hardware architectures. The modular paradigm of O-JRC implies that alterations in communication modalities or algorithm updates necessitate only the relevant block replacement, thus preserving the systemic integrity and functionality. Our key contributions are summarized as follows:

• We introduce an open-source mmWave JRC software platform tailored for efficient development and validation of advanced control algorithms for complex scenarios.

• The modular architecture of JRC can accommodate various MIMO and beamforming configurations, and supports both online/offline model training and real-time control. To the best of our knowledge, O-JRC is the first mmWave MIMO OFDM ISAC platform capable of implementing multiple machine learning-based control algorithms and evaluating them in real-time.

• The functionality of O-JRC is validated with varying levels of algorithmic complexity for beam training and tracking, tested under diverse scenarios including static and mobile conditions, as well as environments with multipath and obstructions.

We implemented two distinct machine learning-based control algorithms: a CNN-based control algorithm and an MAB-based reinforcement learning algorithm, demonstrating O-JRC's capability to support and evaluate a wide range of advanced control strategies. These results highlight O-JRC's potential to accelerate the development and validation of innovative ISAC solutions.

The remainder of this paper is organized as follows: Section 2 reviews the related work on existing ISAC platforms. Section 3 presents an architectural overview of the O-JRC platform. Section 4 provides detailed information about the communication layer. Section 5 describes the control layer in detail. Section 6 outlines the software and hardware implementation. Section 7 evaluates the performance of the implemented control algorithms under various experimental settings. Finally, Section 8 summarizes the contributions and findings of this work.

2. Related Work

The unique challenges presented by mmWave technology have led to the development of various specialized hardware experimentation platforms. These platforms vary widely in capabilities, limitations, and cost. Understanding their evolution and distinctions is crucial, not only for hardware-based research but also for the development of complementary software platforms like O-JRC.

2.1. Early Motorized mmWave Platforms

In the early stages of mmWave communication research, platforms with motorized horn antennas played an important role in studying channel char-

acteristics, as exemplified by the work in [4, 5]. These early systems, including the setup described in [4], primarily aimed to develop spatial channel profiles by measuring the Angle of Arrival (AoA) and Angle of Departure (AoD). They typically utilized dual-motor configurations—one linear to traverse the antenna along a set path and another rotational for azimuthal adjustments—to meticulously capture multipath components in 60 GHz channels. Although effective for initial explorations, these platforms were limited by their static nature and inability to rapidly adapt to dynamic scenarios, highlighting the need for electronically steerable phased array technologies.

2.2. Platforms Featuring Phased Arrays

The transition to platforms featuring programmable phased arrays marked a significant evolution in mmWave research, addressing flexibility limitations inherent in earlier motorized horn antenna systems. OpenMili [6], a pioneering 60 GHz software radio platform, introduced a programmable phased array using discrete RF components. However, it primarily catered to SISO channels, missing out on the mmWave MIMO capabilities crucial for future technologies. The X60 system [7] integrated horn antennas or small phased arrays into a more comprehensive framework, covering both physical and MAC layers, but remained confined to basic, low-order MIMO experiments [8]. Meanwhile, mm-FLEX [9] excelled with its rapid antenna reconfiguration and compliance with IEEE 802.11ad, yet faced scalability challenges, particularly when extending to multiple spatial streams due to complex multi-FPGA synchronization.

2.3. MIMO-Enabled mmWave Platforms

The rapid progress in mmWave technology has elevated the significance of MIMO capabilities, spurring the development of specialized mmWave MIMO platforms. MiRa [10] offers a 24 GHz RF front-end built from discrete components such as power amplifiers, mixers, and local oscillators, serving as a daughterboard for USRPs. Multiple MiRa radios can be used to support mmWave MU-MIMO. M-Cube [11] took a reverse-engineered approach, using an external processor to reconfigure multiple phased array antennas from commercial devices, thereby bypassing the devices' original processors. This method provided a cost-effective solution. MillimeTera [12] employed a fully-digital transceiver design based on an RFSoC FPGA system with four independent streams, enabling more complex and precise MIMO operations. MIMORPH [13] creatively assembled multiple phased arrays in close proximity, constructing mmWave MIMO systems that offer high configurability and flexibility.

As AI/ML-based control systems become increasingly popular and important in integrated RF networks, platforms that support their real-time development and evaluation are crucial. The NeXT framework [14] provides a software-defined testing platform that achieves modular architecture and flexible control algorithm development by separating the control plane and data plane. Both the NeXT framework and the O-JRC platform share a similar architectural approach that separates the control plane and data plane, enhancing modularity and flexibility. This structural separation is particularly beneficial for the iterative development and validation of AI/ML-based control algorithms, enabling independent design, testing, and optimization

without disrupting underlying data transmission processes.

2.4. mmWave JRC Platforms

The concept of integrating radar and communication systems into a single system dates back to the 1960s [15], initially focusing on multifunctional military radars. Recently, this concept has re-emerged as a key interest in both academia and industry, especially relevant in the context of mmWave technology for next-generation edge networks. Such integration is now seen as a promising approach in applications like cellular/mobile networks [16, 17, 18, 19, 20, 21, 22, 23], Vehicle-to-Everything (V2X) [24, 25, 26, 27, 28, 29, 30, 31, 32, 33], and the Internet of Things (IoT) [34], offering advantages in device efficiency, spectrum usage, energy consumption, and overall performance.

As mmWave JRC continued to evolve, various experimental platforms were developed [1, 2, 3], with preliminary radar and communication functionalities implemented on SDRs. JCR70 [1] and RadMAC [3] rely on physically moving the transmitting antenna for direction control. The JRC system [2] can modify beam direction using signal phase manipulation. While these platforms have shown promise, they struggle in intelligently adapting to complex scenarios, primarily due to the lack of a versatile software platform that can significantly reduce the complexity of developing advanced algorithms. To overcome these limitations, we designed O-JRC, an open-source JRC software platform designed for efficient algorithm development and validation in diverse scenarios.

3. O-JRC Architecture Overview

Fig. 2 illustrates the O-JRC software platform, supporting flexible deployment on two types of hardware entity: the JRC station and the standard communication transceiver. Each of these platforms operates within a layered structure comprising communication and control layers. These layers are further modularized into various functional blocks.

3.1. Controllable Platforms

• **JRC Station**: Serving as a multifunctional platform, the JRC station integrates a specialized radar block within its communication layer, in addition to transmitter and receiver blocks. O-JRC is designed to streamline the coordination of these blocks, enhancing the efficacy of combined radar and communication operations within the JRC system.

• **Transceiver**: Exclusively designed for communication operations, this platform does not include the radar block. O-JRC effectively manages its transmission and reception processes, ensuring efficient communication.

3.2. Layered Architecture

• **Communication Layer**. The communication layer, implemented in GNU Radio C++, handles both communication and radar blocks in real time. This includes tasks such as MIMO precoding, OFDM modulation and demodulation, channel estimation, and radar signal processing. It collects and stores relevant feedback data in a data repository. This data encompasses features such as target angle, range, speed, and various communication performance metrics. The layer also retrieves and executes instructions from the data repository.

• **Control Layer**. This layer is completely isolated from the communication layer, allowing for implementation in any programming language, particularly those favorable for AI/ML development like Python. It reads the data from the repository, processes it using AI/ML algorithms, and generates instructions accordingly. These instructions are then deposited back into the repository for the communication layer to retrieve and execute in real-time.

• **Data Repository**. The data exchange between layers occurs through an intermediary data repository, which can be implemented as either a database or a simple log. It serves as the sole conduit between layers, ensuring coherent interactions while preserving their isolation.

3.3. Modular Design

The design organizes each layer into a series of blocks, categorized as common and specialized based on their functionality.

• **Common Blocks**. These are standard blocks utilized across both platforms for generic operations. They are interchangeable between the platforms, promoting a high degree of modularity and flexibility.

• **Specialized Blocks**. Tailored for radar operations, the specialized blocks are exclusively designed to implement the radar functionalities within the JRC station. These blocks rely on the data generated by common blocks to effectively carry out radar-specific operations. Therefore, transitioning between two types of platforms is streamlined: the common blocks form the backbone of the transceiver, while the addition of specialized blocks upgrades it to a JRC station.

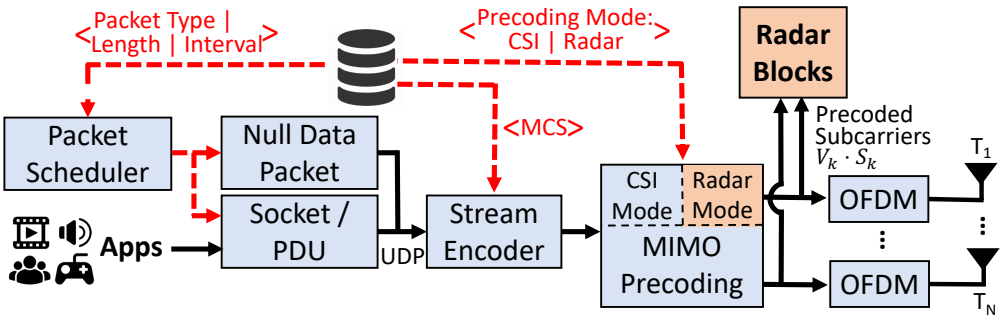


Figure 3: Transmitter Blocks

4. Communication Layer

In the communication layer, we have implemented a comprehensive set of fundamental signal processing modules, which include transmitter blocks, radar blocks, and receiver blocks. The entire layer is engineered using GNU Radio C++, offering both high-speed and precise performance for both communication and radar functionalities.

4.1. Transmitter Blocks

The transmitter blocks, which are configurable for both JRC station and transceiver, enable MIMO-OFDM transmission with support for various modulation schemes. The O-JRC platform currently supports BPSK, QPSK, and 16-QAM modulation. These schemes can be selected and switched at runtime via a graphical user interface (GUI), offering flexibility for protocol experimentation and rapid evaluation of different configurations.

These transmitter blocks not only ensure multiplexing and high data rates through OFDM but also enable precise spatial beamforming via MIMO.

• **Packet Generation (Common).** As illustrated in Fig. 3, the O-JRC platform accommodates data from a wide variety of applications. The data

flow enters the system through a Socket / PDU (Protocol Data Unit), which converts it into UDP (User Datagram Protocol) packets, referred to as data packets. In parallel, the NDP (Null Data Packet) generator serves a similar function but generates a UDP packet without a payload.

The Packet Scheduler is coordinates the scheduling of NDPs and data packets. Following instructions retrieved from the data repository, the scheduler significantly improves resource utilization and performance metrics by harmonizing three key parameters:

1. Packet Type: This parameter enables the selection of data packet or NDP for transmission.
2. Packet Length: This parameter allows the control layer to dictate data packet sizes, influencing several key performance metrics such as throughput, latency, network congestion, reliability, and power.
3. Packet Interval: This parameter allows the control layer to regulate the time interval between packet generations, impacting several performance metrics including throughput, latency, network congestion, channel utilization, and quality of service.

These UDP packets serve as the input for the subsequent stream encoder block. Operating under the instruction of MCS (Modulation and Coding Scheme), the stream encoder converts the incoming data bytes into bits through a sequence of operations including scrambling, convolutional encoding, and puncturing. The encoded bits are then mapped into complex-valued symbols by the modulation scheme.

• **MIMO Precoding**. This block is responsible for generating the transmission frame and its precoding to multiple transmitting antennas [2]. As

depicted in Fig. 3, the control layer manages the selection between two distinct precoding modes:

1. CSI Mode **(Common)**:

   In this mode, NDPs function as channel sounding packets. Upon the reception of an NDP, the communication transceiver estimates the channel. The CSI is subsequently relayed back to the JRC station. There, the control layer refines the CSI or predicts new CSI values, denoted by H^{com}, which are then supplied to this block. Specifically, for the k^{th} subcarrier, the precoding matrix V_k is obtained as the complex conjugate transpose of the right singular vectors from the SVD (Singular Value Decomposition) of H_k^{com}:

   $$H_k^{com} = U_k \Sigma_k V_k^* \tag{1}$$

   where U_k is a complex unitary matrix and Σ_k is rectangular diagonal matrix with non-negative real numbers on the diagonal.

2. Radar Mode **(Specialized for JRC Station)**: In this mode, this block computes the steering vector for precoding based on the angle of target user identified in radar images. Once the target user's angle θ is identified, the transmit steering vector $\mathbf{a}_{tx}$ for each TX l can be formulated as:

   $$\mathbf{a}_{tx}(\theta)[l] = \exp\left(j2\pi f_n \frac{(l-1)d_{tx}\sin(\theta)}{c}\right) \tag{2}$$

   where f_n denotes the frequency of each subcarrier, d_{tx} denotes the antenna spacing, and c is the speed of light.

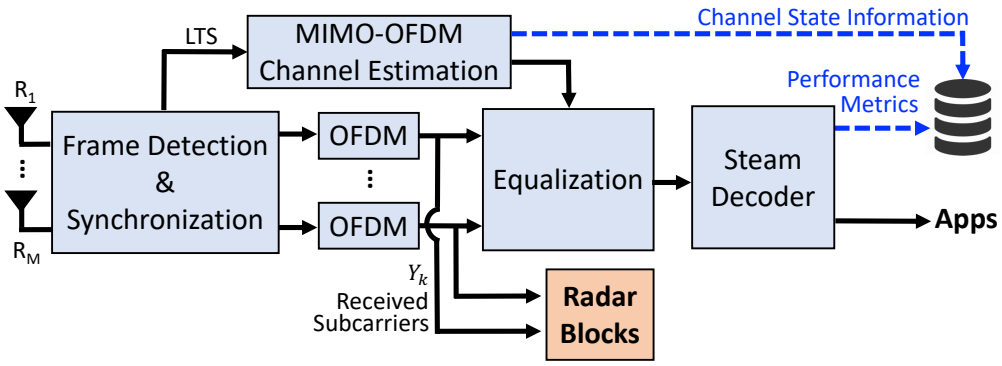


Figure 4: Receiver Blocks

• **OFDM Modulation (Common)**. The precoded subcarriers undergo several transformations within the OFDM blocks to create a time-domain MIMO-OFDM signal for transmission [2].

4.2. Receiver Blocks

The receiver blocks support standard MIMO OFDM receiver functions and can be configured in both transceiver and JRC station [2]. They manage data reception and provide feedback on channel and performance, which is then stored in the data repository for further analysis by the control layer. When installed in a JRC station, these blocks also provide the received subcarriers Y_k to the radar blocks for radar processing. As illustrated in Fig. 4, key blocks include:

• **Frame Detection & Synchronization (Common)**. Integrates signals from all antennas for efficient frame detection and timing synchronization.

• **OFDM Demodulation (Common)**. Converts received time-domain symbols into frequency-domain subcarriers, preparing them for further processing.

• **Channel Estimation & Equalization (Common)**. Uses the LS (Least Squares) algorithm with long training sequences for channel estimation and equalization.

• **Decoding & Performance Metrics (Common)**. Decodes the received signals and evaluates system performance through metrics like SNR (Signal-to-Noise Ratio), PER (Packet Error Rate), and throughput.

4.3. Radar Blocks

The radar blocks are specialized for the JRC station. They are designed to acquire radar channel response and positional information of target users, such as range and angle [2].

• **Radar Channel Estimation (Specialized for JRC Station)**. This block acquires both the precoded transmitted subcarrier of radar symbols $V_k \cdot S_k$ where V_k is the precoding matrix and the received subcarrier Y_k (Fig. 5). It employs the LS algorithm to estimate the radar channel as follows:

$$\hat{H}_k^{rad} = Y_k \cdot (V_k \cdot S_k)^H \cdot (V_k \cdot S_k \cdot (V_k \cdot S_k)^H)^{-1} \tag{3}$$

Notably, the radar blocks can share functionalities with the receiver blocks, such as frame detection & synchronization and OFDM demodulation. However, due to the integration of the radar and transmitter blocks into a single, unified system, which has full control over the timing and method of packet transmission, the radar blocks can obtain Y_k without the need for separate frame detection and synchronization processes for the reflected signals.

• **Range-Angle Estimation (Specialized for JRC Station)**. This block combines the spectral richness of OFDM with the spatial richness of MIMO

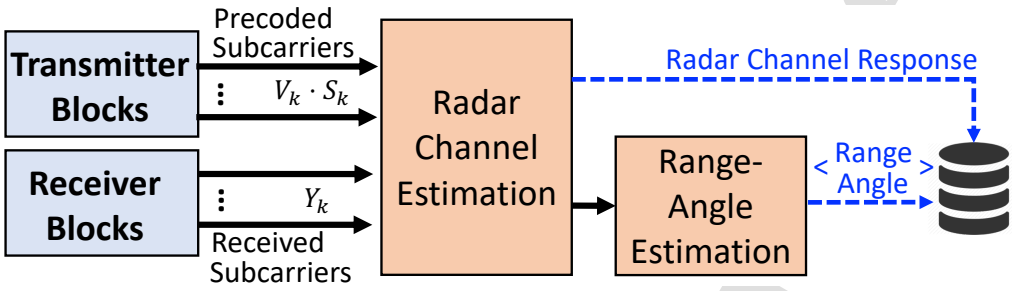


Figure 5: Radar Blocks

to perform range and angle estimation. It primarily consists of two main computational steps: Inverse Discrete Fourier Transform (IDFT) followed by Discrete Fourier Transform (DFT). First, IDFT is performed across different subcarrier frequencies k. This transforms the frequency-domain profiles into a set of time-domain range profiles. Each peak r in a time-domain profile corresponds to a particular distance to a reflecting object. The IDFT effectively correlates the received signal across all subcarriers to identify unique time delays introduced by targets at specific distances:

$$H[r,n] = \sum_{k=0}^{K-1} \hat{H}_k^{rad} exp(j2\pi rk/K) \tag{4}$$

$H[r,n]$ contains the range profiles observed by each transmit-receive antenna pair with a peak where the exponent terms are canceled. With the index of the peak denoted by $\hat{r}$, the range estimate $\hat{R}$ is derived as:

$$\hat{R} = \frac{\hat{r}c}{2K\Delta f} \tag{5}$$

where Δf and c represent the frequency spacing and speed of light, respectively.

Next, DFT is executed on the time-domain range profiles, but now across the virtual antenna indices (which are the product of the number of transmit

antennas, N_{tx}, and the number of receive antennas, N_{rx}, thus $N_{\text{tx}} \times N_{\text{rx}}$) of the MIMO array. This operation unveils the angular spectrum of the target scene. Since antenna elements have known spacing, the resulting phase differences manifest as angular offsets, thereby revealing the AoD:

$$H[r,\theta] = \sum_{n=0}^{N_{\text{tx}} \times N_{\text{rx}} - 1} H[r,n] \exp\left(j2\pi \frac{\theta n}{N_{\text{tx}} \times N_{\text{rx}}}\right) \tag{6}$$

$H[r,\theta]$ contains peaks where the exponents are canceled. With the index of the peak at $\hat{\theta}$, the angle estimate $\hat{\Theta}$ is given as:

$$\hat{\Theta} = \sin^{-1}\left(\frac{2\hat{\theta}}{N_{\text{tx}} \times N_{\text{rx}}}\right) \tag{7}$$

The Range-Angle Estimation approach leverages the frequency diversity of OFDM subcarriers for precise distance determination and the spatial diversity of the MIMO antenna array for accurate angular resolution. The resulting data, encompassing both range and angular information, is subsequently relayed to the data repository for further processing by the control layer.

4.4. Out-of-Band Connection

O-JRC also offers the option of Out-of-Band (OOB) connections, such as Ethernet or optical fiber, in addition to its primary beamforming-based communication. This flexibility is particularly beneficial for development, testing, and algorithm debugging, as it enables efficient data exchange between platforms without interfering with the primary communication or radar operations. In particular, OOB connections provide a convenient method for retrieving real-time communication channel state information from the communication receiver to inform or adjust beamforming strategies or scheduling

algorithms. Moreover, the O-JRC platform is designed as a flexible development and experimentation environment where OOB connections can be selectively activated or deactivated during different stages of the development cycle. This capability facilitates the debugging, validation, and optimization of control algorithms before deployment in real-world scenarios.

5. Control Layer

The control layer fundamentally acts as the controller of the O-JRC platform, utilizing advanced AI/ML algorithms to make intelligent decisions, enhancing the system's efficiency and adaptability across varying operational conditions. The data in the data repository includes vital parameters and metrics sourced from the communication layer. The control layer accesses this repository to retrieve the data, processes it through a series of blocks, and upon deriving actionable insights, populates the repository with instructions.

The separation between layers enables the control layer to be implemented in various programming languages, thereby offering substantial flexibility for algorithm development and implementation. This design is particularly advantageous for AI/ML applications, where computationally intensive tasks may require specialized hardware. For example, the O-JRC platform supports the use of external computational devices, such as graphics cards (GPUs), to accelerate AI/ML operations like CNN-based beamforming control. By offloading complex computations to GPUs, processing time can be significantly reduced compared to CPU-only implementations.

The flexibility of the O-JRC platform facilitates the use of various compu-

tational resources—such as CPUs, GPUs, or dedicated AI accelerators—for executing AI/ML models. This hardware-agnostic approach ensures that users can balance computational complexity and responsiveness according to their specific requirements, enhancing the platform's applicability to a wide range of experimental scenarios.

• **Preprocessing (Common)**. This block serves as the initial processing stage for the raw data stream sourced from the communication layer. It is responsible for conditioning data to a suitable form for further analysis. Typical operations conducted in this block include filtering, normalization, feature extraction, and possibly, error correction. The objective is to refine the data by extracting the salient features and discarding irrelevant noise, thereby creating a refined dataset for the subsequent AI/ML block.

• **AI/ML (Common)**. After preprocessing, the refined data flows into the AI/ML block, the core of the control layer. This block employs advanced learning algorithms to analyze the data, derive insights, and propose optimizations. Depending on the specifics of the O-JRC platform, it implements supervised, unsupervised, or reinforcement learning algorithms to detect complex patterns, predict future states, or optimize certain parameters. It supports both offline training for model development and online training for real-time control.

• **Instruction (Common)**. This block translates optimizations generated by the AI/ML block to actionable instructions for the transmitter, receiver, and radar blocks. These instructions adhere to the requirements of mmWave communication protocols, from beam training and beam tracking from the PHY layer to packet scheduling and MCS selection at the MAC layer.

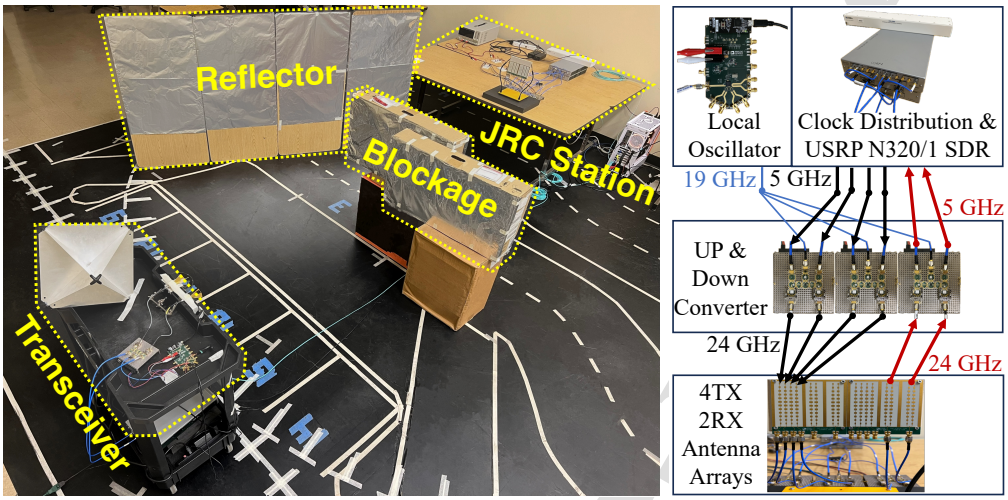
6. Implementation

The evaluation of O-JRC is conducted in an indoor environment (Fig. 6a). The JRC station is placed on a bench top and stationary. The transceiver is placed on a cart, allowing it to be moved along predefined paths on the floor, which are marked with specific distances and angles for precise positioning. Blockages and reflectors are strategically placed between the JRC station and transceiver according to the requirements of each experimental setup, facilitating an assessment of O-JRC's capability to manage diverse scenarios.

6.1. Hardware Configurations

O-JRC accommodates various hardware configurations, seamlessly transitioning from complex MIMO systems to simpler SISO (Single-Input Single-Output) setups. To test these capabilities, we set up two distinct hardware configurations at the JRC station and transceiver.

4X2 MIMO JRC Station. O-JRC supports arbitrary digital MIMO configurations, often requiring more channels than a single SDR can provide. This is achieced by using multiple synchronized SDRs (Fig. 6b). The JRC station has a 4x2 MIMO configuration. As a single N32X series USRP can accommodate up to 2 TX channels, two USRPs are used. The 4 TX channels are split across two USRPs and 2 RX channels are connected to one USRP. The remaining task involves simply adjusting the IP addresses of the USRPs to allow the O-JRC system to recognize them, and setting the 'Number of Channels' parameter of the MIMO Precoding Block to 4 and that of the Radar Channel Estimation Block to 2. The entire process requires minimal workload, ensuring quick and efficient deployment.



(a) Indoor Testbed Deployment (b) JRC STA Hardware

Figure 6: Implementation for O-JRC Testing

We use a custom mmWave front-end unit to perform the up and down conversions between the USRPs and mmWave bands [2]. A common LO (local oscillator) signal is generated at 19 GHz using a wideband frequency synthesizer, which is then distributed to each mmWave front-end unit through a driver amplifier and a 4-way splitter. Utilizing the 19 GHz LO signal, the mmWave front-end unit converts the 5 GHz IF (Intermediate Frequency) signal from the USRP up to 24 GHz for transmission and similarly down-converts received 24 GHz mmWave signals back to 5 GHz. A clock distribution module is deployed to share a common 10 MHz reference signal between the two USRPs, ensuring the internal clocks and timers on the JRC station side are synchronized.

SISO Transceiver. The SISO configuration at the transceiver side mirrors the simplicity of the JRC station setup. Due to the lower channel count of two, only a single N32X series USRP is required. The setup process

is streamlined: we configure the USRP's IP address and set the 'Number of Channels' parameter in both the transmitter and receiver blocks to one each.

A similar custom mmWave front-end unit used in the JRC station is also employed here to perform the up and down conversions between the USRP and mmWave bands. The system uses one horn antenna for transmission and one for reception. Given the single USRP configuration, clock synchronization is not required.

Host PC. O-JRC with its size of approximately 43.1 MB operates on Ubuntu 22.04, GNU-Radio 3.8.10, and Python 3. The Host PC supports real-time operations with a 48 GB memory and an Intel i9-12900K 16-core CPU offering up to 5.2 GHz and 24 threads. For high-rate sample exchange with SDRs, it includes two Intel X520-DA2 and one X710-DA2, totaling six 10 Gbps optical Ethernet ports.

Thanks to the separation of the communication layer and control layer, control algorithms can run on the same PC as the SDR, sharing GPU and CPU resources. Alternatively, the control algorithms can be deployed on a separate machine to fully utilize its computational power for handling complex control tasks, with instructions exchanged with the SDR via the data repository. In our implementation, we utilize shared GPU and CPU resources on a single PC to handle both SDR operations and control algorithms. The GPU is specifically employed to train and run the ML-based control algorithms, highlighting the flexibility and efficiency of the O-JRC platform in seamlessly managing communication and control functionalities, as well as adapting to diverse computational demands.

6.2. Software Configurations

• **Communication Layer**. Reflecting the hardware testbed's structure, we drag-and-drop the corresponding JRC communication blocks in GNU Radio to construct the communication layer. Specifically, at the JRC station, the transmitter is configured to support four data streams, incorporating four distinct OFDM modulation blocks along with other common blocks, as outlined in Section 4.1. On the radar side, two OFDM demodulation blocks and specialized blocks, as detailed in Section 4.3, are employed. The transceiver is streamlined, requiring only a single set of receiver blocks. This configuration establishes a 4x2 MIMO radar and a MISO link between JRC station and transceiver, matching the hardware arrangement.

• **Control Layer and Sample Scripts**. The control layer is implemented through a series of Python scripts, each designed to accommodate various mmWave communication strategies. Below, we present a selection of sample scripts that demonstrate the functionality and effectiveness of the control layer:

1. Preprocessing: Filter: Background Subtraction. This module implements a background subtraction (BG) filter to suppress static clutter and highlight dynamic objects. The filter computes a mean radar response over a temporal window and subtracts it from incoming radar images. This approach enhances the visibility of moving targets while eliminating consistent reflections from static structures, such as walls or furniture, as discussed in [2].
2. AI/ML: Machine Learning-based Beamforming Control. The objective of this module is to optimize beamforming decisions by analyzing

contextual data, such as radar images, and mapping them into precise beamforming angles. By utilizing advanced ML-based techniques, this module aims to enhance adaptability and efficiency in dynamic scenarios, supporting real-time decision-making for both communication and sensing tasks. We demonstrate the application of machine learning techniques by implementing two distinct control algorithms: the GP-UCB algorithm and a CNN-based algorithm. These algorithms serve as examples of how machine learning can be utilized for beamforming, with radar images providing contextual input and beamforming angles as the output. Detailed descriptions of the GP-UCB algorithm and the CNN-based algorithm are provided in Section 6.3 and Section 6.4, respectively.

To facilitate the integration of AI/ML algorithms into the O-JRC platform, we have designed a modular interface that supports seamless implementation and evaluation of different control algorithms. Specifically, the platform provides a dedicated interface `data_interface.py`, which defines two core class structures: `packet_data` and `Radar_data`. The `packet_data` class handles communication parameters such as packet type and length, while the `Radar_data` class provides functionalities for dynamic beamforming configuration.

The open-source implementation includes the `data_interface.py` and its associated modules, offering a structured framework for the development and evaluation of AI/ML algorithms. Further details and implementation guidelines are publicly available at [35].

3. Instruction: PHY: Beamforming: Beam Sweep. Within the instruc-

tion block, we establish a baseline implementation at the PHY layer using a sample script named 'Beam Sweep.' Beam Sweep is a widely adopted technique for beam training and tracking, determining the optimal beam direction by probing across all possible angles. It cyclically transmits NDP packets across a sweeping angular range with a resolution of 1°, allowing the receiver to identify the optimal beam index $k^\star$ that maximizes the received signal power:

$$k^\star = \arg\max_k |y_k|^2 = \arg\max_k |\mathbf{h}^H \mathbf{a}_k|^2 \tag{8}$$

where $\mathbf{h}$ is the channel vector, and $\mathbf{a_k}$ denotes the transmit beamforming vector selected from a predefined beambook $\mathbf{a_1}, \mathbf{a_2}, ..., \mathbf{a_k}$, with each $\mathbf{a_k}$ corresponding to angle θ_k as defined in equation (2).

This method reflects conventional IEEE 802.11ad beam training procedures.

4. Instruction: PHY: Beamforming: `RaB`. The `RaB` (Radar-Aided Beamforming) script represents a beamforming technique exclusive to the JRC platform. The `RaB` module executes the radar-assisted beamforming strategy proposed in [2]. It leverages radar-derived range-angle information, extracted from reflected NDPs or data packets, to steer the communication beam toward the estimated target direction. This approach eliminates the need for feedback and supports low-latency adaptation, offering a practical implementation of JRC beam alignment. The angle of the target is estimated by processing the radar channel CSI using a DFT-based range-angle processing method, as detailed in Section 4.3. The estimated direction θ_{target} is subsequently mapped to a transmit beamforming vector via the steering vector for-

mulation given in Equation (2). Therefore, in the absence of advanced algorithms within the AI/ML block, it operates under the assumption that the angle of the target directly detected by the radar image corresponds to the optimal beam direction.

5. Instruction: MAC: 802.11ad. We deploy a packet scheduling script to adhere to the MAC protocol outlined by the IEEE 802.11ad standard, demonstrating our platform's compatibility with protocol-compliant mmWave communication. Our experimentation specifically targets the Beacon Header Interval (BHI) phase, leveraging this period for conducting beam sweep or radar sensing activities. According to the standard, channel access is organized into 100 ms Beacon Intervals (BI), each consisting of a BHI for initial beam training/tracking, followed by a Data Transmission Interval (DTI) for data exchange [36].

6.3. GP-UCB

The `GP-UCB` (Contextual Gaussian Process Upper Confidence Bound Multi-Arm Bandit) algorithm integrates both offline learning and online decision-making and is particularly well-suited for wireless beam alignment in mmWave systems. In such systems, the channel response is highly dynamic due to mobility, blockage, and multipath effects, making traditional exhaustive beam training approaches inefficient and impractical.

To address this, `GP-UCB` formulates beam alignment as a sequential learning problem, where each beamforming vector corresponds to an action, and the received signal strength serves as the reward. Unlike conventional bandit algorithms that treat each beam as an independent arm, `GP-UCB` leverages a kernelized Gaussian Process (GP) model to capture the correlation between

nearby beam directions. This is grounded in the observation that physically adjacent beamforming vectors often produce similar radiation patterns and receive comparable channel gains, especially in antenna arrays with smooth angular transitions.

This learning framework aligns well with the properties of mmWave communication, where fast adaptation, sample efficiency, and spatial correlation exploitation are critical. As demonstrated in [37], kernelized bandit models offer theoretical guarantees in terms of regret and constraint satisfaction, making `GP-UCB` a powerful tool for dynamic beam alignment in mmWave wireless networks.

• **Offline Learning**. Utilizing the principle that similar beamforming angles produce comparable beam patterns [37], we implement the `GP-UCB` method [38] to model the relationship between communication SNR and radar imagery. This correlation is crucial, as the SNR directly relates to beam patterns, and the beamforming angles are derivable from radar images. Accordingly, our model processes data from communication sessions at the JRC station, utilizing `beam sweep` and `RaB` scripts. The collected dataset includes historical contextual information such as time, SNR, PER, range, angles, and radar images.

During the learning phase, the model systematically selects the beamforming angle to maximize signal strength at the user end as follows:

$$\max_{a_t} |\mathbf{H}_t w(a_t)| \cdot P_{signal} \tag{9}$$

where $w(a_t)$ is the beamforming gain of beamforming angle a_t at time t, $\mathbf{H}_t$ denotes the channel gain, which varies depending on location and time, and

P_{signal} denotes the transmitted signal power.

Since $\mathbf{H}_t$ is unknown, the model is positioned as a sequential decision-making problem of beamforming angle selection. To address this, we implemented a GP bandit. In each round t, the bandit selects the beamforming angle a_t using the following optimization:

$$a_t = \underset{a \in \mathcal{A}}{\operatorname{argmax}} \left(\mu_{t-1}(a, r_t) + \beta_t^{1/2} \sigma_{t-1}(a, r_t) \right) \tag{10}$$

where $\mathcal{A}$ represents the range of beamforming angles, r_t denotes the most recent radar image, and β_t consists of a set of constants. The functions $\mu(\cdot)$ and $\sigma(\cdot)$ represent the posterior mean and variance, respectively, computed from all historical contextual information that correlates with the desired SNR, where the posterior mean and variance are defined as follows.

$$\mu_t(x) = k_t(x)^T (K_t + \lambda I)^{-1} y_{1:t} \tag{11}$$

$$\sigma_t^2(x) = k(x, x) - K_t(x)^T (K_t + \lambda I)^{-1} k_t(x) \tag{12}$$

where $y_{1:t}$ is the reward vector, $x \in X$ denotes the set of all beamforming angle and radar image pairs, λ denotes the noise variance, K_t denotes the kernel matrix, and $k_t(x) = [k(x_1, x), ..., k(x_t, x)]^T$ is the kernel vector for kernel function k.

• **Online Decision-Making**. The system first retrieves radar images to serve as contextual data, which are then input into the `GP-UCB` model to determine the optimal beamforming angle.

• **Development Complexity**. The entire GP-UCB algorithm was developed by a novice in SDR programming within just two days, demonstrating

the efficiency of development on the O-JRC platform. This efficiency primarily stems from the disaggregation of control logic from signal processing via a layered architecture, thereby eliminating the need for extensive SDR programming. Additionally, the programmer effectively utilized the regression functions and the Radial Basis Function (RBF) kernel from the external library GPy [39] for regression and kernel operations in `GP-UCB`, significantly accelerating the implementation of the algorithm.

6.4. CORAL-CNN

Correlation Alignment (CORAL) is an effective unsupervised domain adaptation method designed to address domain shifts by aligning the second-order statistics (covariance) of feature distributions between the source and target domains [40]. By minimizing the difference in covariance matrices, CORAL ensures that features from the two domains are statistically aligned, thereby improving model generalization in case of domain mismatch.

In JRC systems, domain shifts refer to variations in radar signatures caused by dynamic changes in the operating environment. These shifts are critical to address, as they arise from factors such as imperfections and errors inherent in channel estimation and radar signal processing. Even when the target user remains in the same physical location, the estimated radar signatures can exhibit significant variations. These variations arise from factors such as estimation imperfections, changes in environmental conditions, or even fluctuations in device temperatures. Such dynamic changes in the domain make traditional CNN models, which assume a static feature distribution, unsuitable for real-time JRC applications. To address this, we integrate CORAL with a multi-layered CNN to bridge the distribution gap

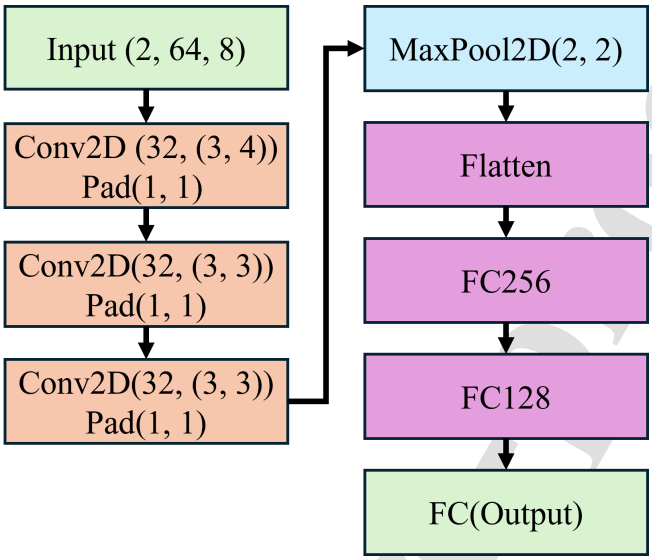


Figure 7: CORAL-CNN Model Architecture

between data collected at different times. CNNs are powerful for learning hierarchical feature representations directly from raw data, capturing spatial and temporal dependencies crucial for complex tasks like beamforming and radar signal processing. However, data collected at different times can exhibit domain mismatches due to dynamic changes in the operating environment. By embedding the CORAL loss into a CNN framework, we create a unified approach where the CNN extracts robust, domain-invariant features, and the CORAL component ensures alignment between the distributions of these temporally separated datasets. This combination effectively addresses the dual challenges of robust feature learning and domain adaptation, enabling reliable performance under the estimation errors and imperfections that are unavoidable in practical real-time JRC deployments.

The CORAL loss is defined as:

$$\mathcal{L}_{CORAL} = \frac{1}{4d^2}\|\mathbf{C}_S - \mathbf{C}_T\|_F^2, \tag{13}$$

where:

- $\mathbf{C}_S$ and $\mathbf{C}_T$ are the covariance matrices of the features extracted from datasets collected at different times, representing the source and target domains, respectively.
- $\|\cdot\|_F$ denotes the Frobenius norm, and d is the feature dimensionality.

The CORAL loss minimizes the distance between the covariance matrices of the source and target feature distributions, aligning the two domains in feature space.

We implemented a CORAL-CNN designed to handle domain adaptation in real-time JRC systems as shown in Fig. 7. The network consists of three 2-dimensional convolutional layers, each followed by batch normalization and ReLU activation functions, to extract hierarchical spatial features from the radar signatures. The convolutional layers are configured with 32 filters and use kernel sizes of 3×3 Conv2D(32,(3,3)) or 3×4 Conv2D(32,(3,4)) with appropriate padding Pad(1,1) to maintain spatial consistency. Additionally, a max-pooling layer with 2×2 window size MaxPool2D(2,2) is applied after the convolutional operations to reduce spatial dimensions, ensuring essential features are preserved while decreasing computational complexity.

After the convolutional layers, the output is flattened and passed through two fully connected layers FC256 and FC128 with 256 and 128 units, respectively, each equipped with ReLU activation functions, for high-level feature

representation learning. A dropout layer with a rate of 0.1 is employed to mitigate overfitting. The final output layer is a fully connected layer FC(Output) that maps the extracted features to the target classes (beamforming angles).

The input to the network is a radar signature represented as a two-channel tensor Input(2,64,8), where 2 corresponds to the real and imaginary components of the complex radar CSI, 64 represents the 64 OFDM subcarriers, and 8 denotes the 8 TX-RX pairs of JRC station. The output of the network is the predicted beamforming angle, corresponding to one of the predefined classes. This design leverages the feature extraction capabilities of CNNs combined with the domain adaptation power of CORAL, ensuring robust performance across varying conditions and mitigating the impact of channel estimation errors and imperfections.

• **Offline Learning**. The CORAL-CNN model is trained using a supervised learning approach to classify CSI data into predefined classes. During training, the input data is first processed through a series of convolutional layers with ReLU activations and batch normalization. Specifically, the model applies two convolutional blocks with 32 filters each, followed by a max-pooling operation to reduce spatial dimensions. The output from the final convolutional layer is flattened and passed through two fully connected layers, with dropout applied to prevent overfitting.

The final classification layer outputs predictions corresponding to the original number of labels, while the intermediate features extracted from the convolutional layers are also made available for potential auxiliary tasks. The model is optimized using CORAL loss, with gradients computed via backpropagation. Training is performed by feeding batches of labeled CSI

data, minimizing the loss through iterative updates using stochastic gradient descent or other optimizers.

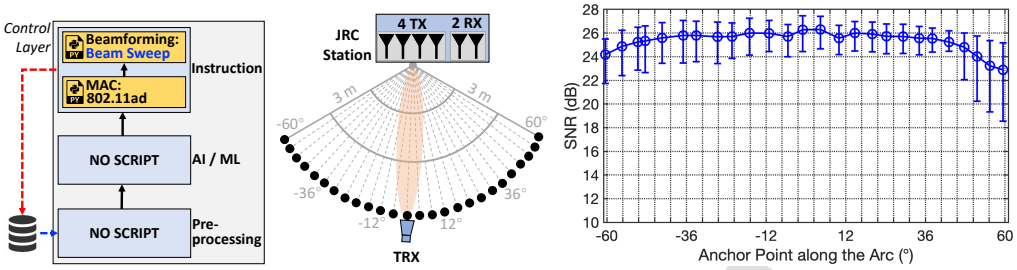
• **Online Decision-Making**. In an online scenario, the pre-trained CORAL-CNN model receives real-time radar CSI inputs and outputs a predicted class. The predicted class is then mapped to a specific predefined beamforming angle, enabling dynamic adjustment of the JRC's beam direction to optimize performance in time-varying channel conditions.

6.5. Metrics

We evaluate the communication performance using four key metrics: SNR, PER, Data Packet Ratio, and Packet Reception Ratio (PRR). The Data Packet Ratio is defined as the proportion of data packets successfully transmitted by the JRC station to the total number of packets sent by the JRC station including both Data and NDP packets. The data packet ratio is used to evaluate how much of the available resources the control algorithm allocates to communication, since NDP packets are primarily used for sensing. This ratio can be mathematically represented as follows:

$$\text{Data Packet Ratio} = \frac{\text{Number of Data Packets Transmitted}}{\text{Total Packets Sent (NDPs + Data Packets)}}$$

The packet reception ratio is defined as the proportion of data packets successfully received by the communication user to the total number of data packets transmitted by JRC station.



(a) Control Layer Scripts Configuration (b) Beam Training Experimental Setup (c) SNR at Each Anchor Point with 90% Confidence Intervals

Figure 8: Validating Sweeping-based mmWave Communication with O-JRC

7. Evaluation

The primary objective of these experiments is to illustrate the efficiency and simplicity of implementing and evaluating diverse algorithms on O-JRC, rather than directly comparing their performance. The experimental results range from sweeping-based mmWave communication as a baseline, to radar-aided mmWave communication, and finally to more complex machine learning-based beamforming control algorithms such as `GP-UCB` and CORAL-CNN. O-JRC facilitates the seamless integration of these approaches, thereby enabling the development and evaluation of innovative strategies. Its ability to support real-time decision-making, leverage radar-assisted functionalities, and adapt to evolving scenarios demonstrates its practical applicability in addressing the challenges of dynamic and complex ISAC tasks.

7.1. Sweeping-based mmWave Communication

Our initial experiment aims to verify that the O-JRC system can be utilized for traditional mmWave communication without radar assistance and is compatible with standard protocols. Fig. 8a displays the scripts used

by the control layer, where only a MAC layer script and a traditional `beam sweep` script for beam discovery are added to the instruction block.

Fig. 8b depicts the experimental setup. The station is centrally positioned while the transceiver is placed along an arc with a radius of 3 meters, covering a total angle of 120°. The transceiver's horn antenna always points towards the station, creating a single-path beam in LOS. The arc consists of 26 anchor points, dividing it into 25 segments, with anchor points located every 4.8°. Angles cover from $-60°$ to 60° counterclockwise. The transceiver is moved to each of the 26 anchor points to conduct beam training with the station performing beam sweeping. At each point, the station executes 50 beam sweeps to collect SNR data from the transceiver. This process is repeated across all anchor points 10 times, yielding a total collection of over 1.5 million SNR data points.

The SNR curve of Fig. 8c exhibits a trend consistent with the antenna's directional characteristics, with the highest SNR values at the center positions directly facing the JRC station antenna and lower SNRs at off-center positions. The highest SNR value, approximately 26 dB, is achieved when the transceiver is directly facing the center. However, when the angle is shifted to the 60° position, the SNR value decreases to about 23 dB, indicating a drop of approximately 3 dB from the center point to this position. This marks the boundary from the point of maximum gain to where the signal strength drops to half. Based on this criterion, we can determine the effective coverage range of the beam. Observing a similar SNR decrease at the $-60°$ position, we can conclude that the beam angle range covered by the antenna is approximately 120 degrees (from $-60°$ to 60°), within which

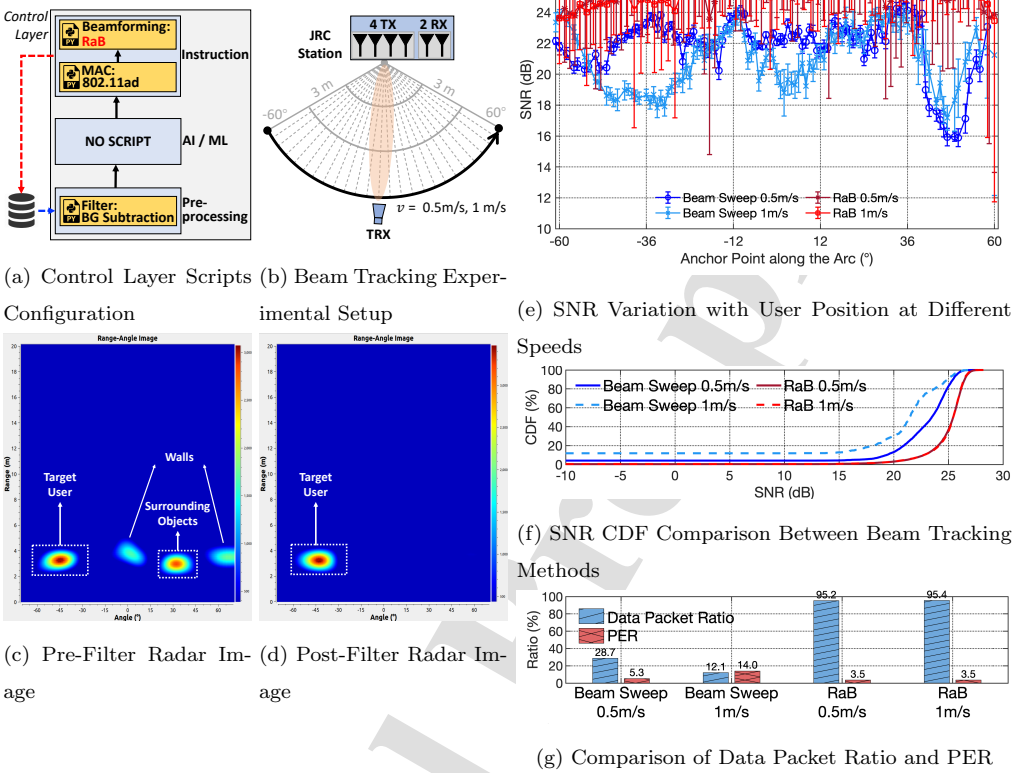
the signal strength remains relatively high.

This experiment demonstrates that, in a more ideal case—where the communication user is always facing the JRC station and remains static—a sweeping-based algorithm can achieve the best possible results even without leveraging radar functionality, serving as a baseline. Such algorithms can still be implemented and evaluated directly on the O-JRC platform without requiring modifications to the communication layer's code for different control algorithms.

7.2. Radar-Aided mmWave Communication

The O-JRC's capability for radar-aided mmWave communication is assessed next, particularly focusing on beam tracking under mobile scenarios, while ensuring compatibility with standard protocols.

Fig. 9a presents the scripts utilized by the control layer. Given that radar-aided mmWave communication does not rely on feedback from the transceiver, we introduce a background subtraction filter script in the preprocessing block to eliminate static background, thus highlighting moving targets. The configuration in the instruction block remains essentially similar to that of native mmWave communication, with the MAC script unchanged and simply replacing the `beam sweep` script with the RaB script. The experimental setup illustrated in Fig. 9b is similar to that of Fig. 8b, but with beam tracking experiments being conducted as the transceiver moves along the arc at two distinct average speeds—0.5 m/s and 1 m/s—with the horn antenna consistently aligned towards the station. To highlight the enhancement of mmWave communication by radar functionality, we also conduct comparative beam tracking experiments using a native mmWave communica-



(a) Control Layer Scripts Configuration

(b) Beam Tracking Experimental Setup

(c) Pre-Filter Radar Image

(d) Post-Filter Radar Image

(e) SNR Variation with User Position at Different Speeds

(f) SNR CDF Comparison Between Beam Tracking Methods

(g) Comparison of Data Packet Ratio and PER

Figure 9: Validating Radar-Aided mmWave Communication with O-JRC

tion configuration (Section 7.1). To highlight the enhancement of mmWave communication by radar functionality, we also conduct comparative beam tracking experiments using a native mmWave communication configuration (Section 7.1). Each beam tracking method is repeated 20 times at both speeds, resulting in a total of 37,000 SNR data points.

Fig. 9c displays a pre-filter radar image, where walls and surrounding objects produce reflections of varying intensities on the radar image. However, as these are stationary, they are filtered out by the background subtraction

filter script, leaving only the moving targets visible, as shown in Fig. 9d.

Fig 9f illustrates the CDF (cumulative distribution function) of SNR. The CDF corresponding to `beam sweep`-based beam tracking at 1 m/s is the farthest to the left among all, followed by the `beam sweep` at 0.5 m/s. This is due to the `beam sweep` requiring time to probe all possible angles, and during which time, the transceiver moves to a new position, rendering the information from previous probes outdated. This lag is exacerbated at higher speeds, leading to worse SNR. Such lags are detrimental to achieving comprehensive real-time contextual awareness, as they hinder the system's ability to accurately and promptly adapt to fast environmental changes.

Conversely, the `RaB` curves exhibit superior performance compared to `beam sweep`, especially at higher speeds. This observation validates O-JRC's ability to rapidly respond and process real-time data in dynamic scenarios. Particularly, the `RaB` curves highlight the rapid response capability of O-JRC to movement changes, as shown by their shift to the right of the CDF plot at both 0.5 m/s and 1 m/s speeds, with the curves nearly overlapping. This effectiveness stems from `RaB`'s use of immediate location information by real-time radar processing, which minimizes the lag. Although the immediate location may not ascertain the optimal beam direction, it still maintains a relatively high SNR.

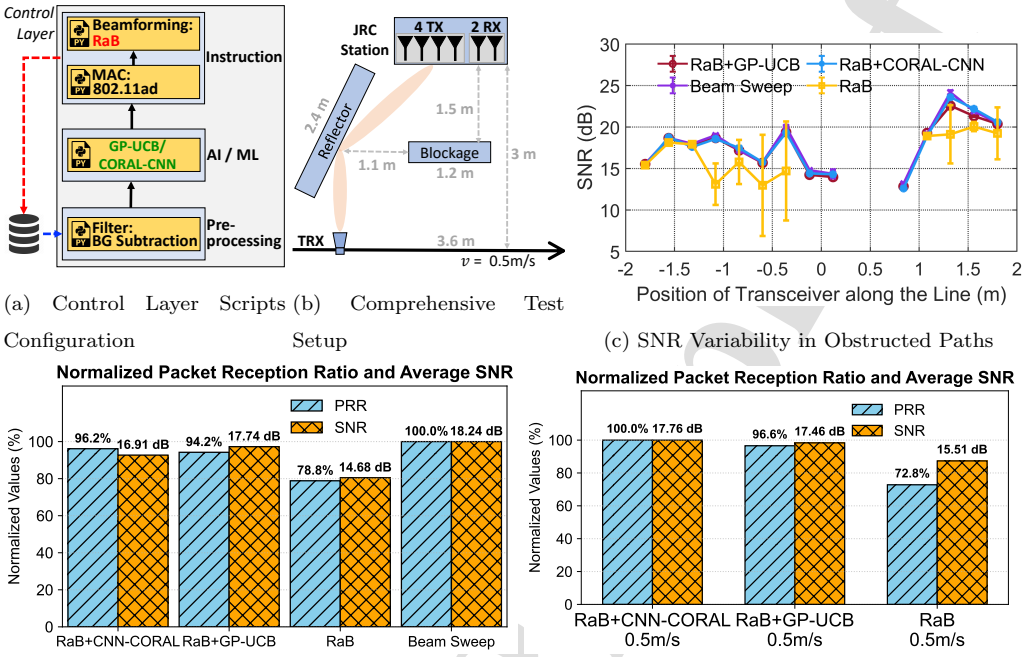
We also assessed the efficiency of beam alignment by examining the data packet ratio, as demonstrated in Fig. 9g. The `RaB` method consistently uses a higher proportion of data packets for transmission, enhancing the continuity and efficiency of communication in dynamic environments. Unlike `beam sweep`, which relies heavily on NDPs to probe all possible directions

to obtain optimal beamforming angle upon cyclic redundancy check (CRC) failures, `RaB` can update radar images and maintain user location using both NDPs and data packets. This dual capability allows `RaB` to reserve NDP usage for instances when the user moves outside the range of the current data packet beam, leading to higher data packet ratios and lower PER compared to `beam sweep`. The ability of `RaB` to rapidly update its position ensures a more efficient payload transmission and a significant reduction in PER.

This experiment demonstrates that, compared to the scenario in Section 7.1, dynamic environments present challenges where there is insufficient time to perform complete beam sweeping. Leveraging radar functionality enables rapid adjustments to beamforming angles, thereby maintaining high communication efficiency even under changing conditions. The results highlight that in dynamic scenarios, control algorithms can utilize the radar functionality of the O-JRC platform to assist communication, leading to enhanced performance. This underscores the platform's capability to implement and evaluate radar-aided control algorithms, showcasing its utility in optimizing communication performance with the help of radar in challenging and dynamic environments.

7.3. Incorporation of Learning Algorithms

A key objective of developing O-JRC is to provide a software platform that enables the rapid implementation and evaluation of AI/ML algorithms in real-time to address complex tasks in JRC systems. This is achieved by decoupling the control algorithms from the communication functionality, eliminating the need to modify the communication layer's code for each algorithm. Instead, ML-based control algorithms can be directly integrated



(a) Control Layer Scripts Configuration

(b) Comprehensive Test Setup

(c) SNR Variability in Obstructed Paths

(d) Average SNR and Normalized PRR in Static User Case

(e) Average SNR and Normalized PRR in Moving User Case

Figure 10: Validating the Incorporation of Learning Algorithms with O-JRC

into the AI/ML block, with only minimal adjustments to the corresponding interface. To demonstrate this flexibility, we implemented two distinct ML methods: a `GP-UCB` algorithm and a CORAL-CNN approach. These algorithms were seamlessly integrated within the AI/ML block, as shown in Fig. 10a, showcasing the efficiency of deploying and validating advanced algorithms on the O-JRC platform.

Comprehensive tests are carried out in dynamic situations involving blockages and multipath conditions. As depicted in Fig. 10b, the transceiver moves along a straight line perpendicular to the JRC station, maintaining a distance of 3 m. This line has a total length of 3.6 m, extending 1.8 m to

either side of the perpendicular, thus creating a segment ranging from -1.8 m to 1.8 m. A blockage, measuring 1.2 m in width, is situated 1.5 m away from the station's perpendicular. Additionally, a reflector, 2.4 m long and positioned at a 30° angle to the vertical, is installed to create a multipath scenario. The transceiver traverses this 3.6 m line at average speeds of 0.5 m/s for data gathering. To train the `GP-UCB` and CORAL-CNN models, we first collected data in the environment depicted in Fig. 10b. The movement path was divided into 16 anchor points, and the communication user was positioned at each anchor point sequentially. At each anchor point, the JRC station transmitted NDP packets to obtain radar CSI, followed by a beam sweeping process to identify the optimal beamforming angle. The beamforming angles obtained through sweeping were added as labels to the radar CSI collected via NDP packets for training the CORAL-CNN model. Additionally, the sweeping data were recorded and utilized for training the `GP-UCB` model. To enable CORAL-CNN to align domains effectively, we repeated the experiments across multiple days at each anchor point, collecting data under varying conditions. This approach allowed the model to leverage data collected from different days to address domain shifts more robustly. In total, we gathered 951,986 samples for training the `GP-UCB` model and 616,152 samples for training the CORAL-CNN model.

Fig. 10c illustrates the SNR performance and 90% confidence intervals for four methods—`beam sweep`, `RaB`, `RaB` +`GP-UCB`, and `RaB` +CORAL-CNN—at each anchor point in a static communication user scenario. `beam sweep` serves as the baseline, providing an exhaustive search to identify the best possible SNR by evaluating all beamforming angles. The results indicate that

both RaB +GP-UCB and RaB +CORAL-CNN achieve SNR levels comparable to those obtained through beam sweeping. In addition, the RaB +GP-UCB and RaB +CORAL-CNN methods utilize radar information to achieve similar SNR performance without the need for a complete beam sweep, highlighting their efficiency. Notably, the confidence interval for RaB is significantly wider compared to the other three methods. This is because the RaB method does not always successfully identify the communication user's location in complex environments, making its decisions more susceptible to environmental factors. In contrast, the GP-UCB and CORAL-CNN-based methods leverage training to learn the environmental impact on radar signatures at each anchor point, enabling them to make more fast and reliable decisions.

Furthermore, near 0m, the RaB method fails to establish a communication link with the communication user, while the other three methods successfully maintain the connection. This is because RaB fails to identify the reflected path created by the reflector. In contrast, the other methods either perform a full beam sweep to find the angle of reflected path or leverage learning-based approaches to identify the reflected path effectively. Around 0.5m, none of the four methods are able to establish an effective link. This is due to the communication user moving behind the obstruction, where no viable path exists for communication, direct or reflected. Additionally, the horn antenna on the communication user's side can only receive signals from specific angles, further preventing communication in this scenario.

Fig.10d presents the overall average SNR performance and packet reception ratio (PRR) for the static communication user scenario. In the figures, we normalize the PRR data by taking the highest PRR as the baseline for

comparison. In the SNR comparison, the average SNR data for different methods is labeled in dB units, and the bar heights are normalized for comparison. The results align with the observations from Fig.10c, showing that the average SNR for the `beam sweep`, `RaB` +`GP-UCB`, and `RaB` +CORAL-CNN methods are comparable, with similar PRR values. However, the `GP-UCB` and CORAL-CNN methods do not always make accurate decisions, leading to a slightly lower PRR compared to `beam sweep`. On the other hand, `RaB` frequently fails to correctly identify the communication user's location, resulting in lower average SNR and PRR performance. These findings highlight the relative strengths and limitations of each approach in maintaining robust communication under static conditions.

To evaluate the performance of different beamforming control algorithms in dynamic and complex JRC scenarios, we utilized radar signatures as reference information to guide decision-making. In the setup shown in Fig. 10b, the communication user moved at an average speed of 0.5 m/s. For comparison, `RaB`, `RaB` +`GP-UCB`, and `RaB` +CORAL-CNN were configured to make decisions, i.e., adjust beamforming angles, every 0.5 seconds. Both `GP-UCB` and CORAL-CNN leveraged pre-trained models, and to ensure real-time decision-making, model updates were disabled during testing. Each method was tested ten times, with each trial involving the communication user moving along the line segment from one end to the other and then returning. The results are presented in Fig.10e. The results indicate that `RaB` +`GP-UCB` and `RaB` +CORAL-CNN achieved similar performance, whereas `RaB` showed relatively lower performance. Notably, `RaB` demonstrated comparable performance in both static and dynamic communication user scenarios. This

consistency can be attributed to the preprocessing step, which analyzes temporal variations in radar CSI to mitigate self-interference and enhance the clarity of radar signatures. These findings demonstrate the advantages of integrating trained control algorithms in dynamic scenarios to enhance communication performance.

As a reference, we repeat one static experiment and recorded radar CSI data under Fig. 10b setup and used this data to compare the validation accuracy of CNN models with identical layer structures and training data but different loss functions. Specifically, we evaluated a traditional CNN trained with standard cross-entropy loss and a CORAL-CNN trained with CORAL loss. The validation accuracy improved significantly from 18.21% with standard cross-entropy loss to 80.98% when CORAL was applied.

Through these experiments, we demonstrate the versatility and robustness of the O-JRC platform in enabling the rapid implementation and evaluation of a diverse set of control algorithms for ISAC systems. The platform's modular and flexible architecture allows control algorithms to be seamlessly integrated into the corresponding control layer's block with minimal changes to the interface, eliminating the need for modifications to the underlying hardware or communication functionalities. This flexibility ensures that O-JRC supports a wide variety of solutions, making it a developer-friendly tool for advancing research and development.

8. Conclusion

In this paper, O-JRC, an open-source JRC software platform, is introduced to support efficient algorithm development and validation across vari-

ous scenarios. Its layered and modular architecture effectively separates the communication and control layers, streamlining the integration of AI/ML algorithms. This design not only simplifies the process of swapping out or upgrading algorithms within the control layer but also facilitates easy modification of communication functions.

We implemented two fundamentally different machine learning algorithms: an MAB-based reinforcement learning algorithm and a CORAL-CNN a CNN-based algorithm. To the best of our knowledge, O-JRC is the first open-source platform capable of simultaneously implementing and validating different types of ML-based control algorithms without requiring reconfiguration of the underlying hardware or communication functionalities. Complex environments demand more sophisticated algorithms, and through experiments in static, dynamic, multipath, and blockage scenarios, we evaluated the performance of various control algorithms under different conditions. These experiments demonstrate O-JRC's ability to support real-time evaluation of diverse control algorithms, highlighting its flexibility and robustness and solidifying its efficacy as a versatile tool for advancing research and development in the JRC domain.

Future work will focus on enhancing the O-JRC platform to support real-time cross-layer optimization and control between the MAC layer and PHY layer, enabling more efficient coordination and adaptability in dynamic RF environments. Additionally, expanding the platform's interface to facilitate the implementation and evaluation of scheduling algorithms in multi-user scenarios will be a critical step toward providing comprehensive support for complex network conditions.

References

[1] P. Kumari, A. Mezghani, R. Heath, Jcr70: A low-complexity millimeter-wave proof-of-concept platform for a fully-digital simo joint communication-radar, IEEE Open Journal of Vehicular Technology (2021).

[2] C. Ozkaptan, H. Zhu, E. Ekici, O. Altintas, A mmwave mimo joint radar-communication testbed with radar-assisted precoding, IEEE Transactions on Wireless Communications (2023).

[3] L. Simić, J. Arnold, M. Petrova, P. Mähänen, Radmac: radar-enabled link obstruction avoidance for agile mm-wave beamsteering, in: HotWireless'16, 2016.

[4] H. Xu, V. Kukshya, T. Rappaport, Spatial and temporal characteristics of 60-ghz indoor channels, IEEE Journal on Selected Areas in Communications (2002).

[5] T. Wei, X. Zhang, Mtrack: High-precision passive tracking using millimeter wave radios, in: MobiCom'15, 2015.

[6] J. Zhang, X. Zhang, P. Kulkarni, P. Ramanathan, Openmili: A 60 ghz software radio platform with a reconfigurable phased-array antenna, in: MobiCom'16, 2016.

[7] S. Saha, Y. Ghasempour, M. Haider, T. Siddiqui, P. De Melo, N. Somanchi, L. Zakrajsek, A. Singh, O. Torres, D. Uvaydov, J. Jornet, E. Knightly, D. Koutsonikolas, D. Pados, Z. Sun, X60: A programmable

testbed for wideband 60 ghz wlans with phased arrays, in: WiNTECH'17, 2017.

[8] Y. Ghasempour, M. Haider, C. Cordeiro, D. Koutsonikolas, E. Knightly, Multi-stream beam-training for mmwave mimo networks, in: MobiCom'18, 2018.

[9] J. Lacruz, D. Garcia, P. Mateo, J. Palacios, J. Widmer, Mm-flex: An open platform for millimeter-wave mobile full-bandwidth experimentation, in: MobiSys'20, 2020.

[10] O. Abari, H. Hassanieh, M. Rodreguiz, D. Katabi, Poster: A millimeter wave software defined radio platform with phased arrays, in: MobiCom'16, 2016.

[11] R. Zhao, T. Woodford, T. Wei, K. Qian, X. Zhang, M-cube: A millimeter-wave massive mimo software radio, in: MobiCom'20, 2020.

[12] M. Polese, F. Restuccia, A. Gosain, J. Jornet, S. Bhardwaj, V. Ariyarathna, S. Mandal, K. Zheng, A. Dhananjay, M. Mezzavilla, J. Buckwalter, M. Rodwell, X. Wang, M. Zorzi, A. Madanayake, T. Melodia, Millimetera: Toward a large-scale open-source mmwave and terahertz experimental testbed, in: mmNets'19, 2019.

[13] J. Lacruz, R. Ortiz, J. Widmer, A real-time experimentation platform for sub-6 ghz and millimeter-wave mimo systems, in: MobiSys'21, 2021.

[14] J. Hu, Z. Zhao, M. McManus, S. K. Moorthy, Y. Cui, N. Mastronarde, E. S. Bentley, M. Medley, Z. Guan, Next: Architecture, prototyping and

measurement of a software-defined testing framework for integrated rf network simulation, experimentation and optimization, Computer Communications 210 (2023) 342–355.

[15] R. Mealey, A method for calculating error probabilities in a radar communication system, IEEE Transactions on Space Electronics and Telemetry (1963).

[16] A. Zhang, M. Rahman, X. Huang, Y. Guo, S. Chen, R. Heath, Perceptive mobile networks: Cellular networks with radio vision via joint communication and radar sensing, IEEE Vehicular Technology Magazine (2021).

[17] F. Liu, C. Masouros, A. Li, H. Sun, L. Hanzo, Mu-mimo communications with mimo radar: From co-existence to joint transmission, IEEE Transactions on Wireless Communications (2018).

[18] A. Martone, K. Gallagher, K. Sherbondy, Joint radar and communication system optimization for spectrum sharing, in: RadarConf'19, 2019.

[19] F. Liu, C. Masouros, A. Li, T. Ratnarajah, Robust mimo beamforming for cellular and radar coexistence, IEEE Wireless Communications Letters (2017).

[20] M. Rahman, J. Zhang, X. Huang, Y. Guo, R. Heath, Framework for a perceptive mobile network using joint communication and radar sensing, IEEE Transactions on Aerospace and Electronic Systems (2020).

[21] F. Liu, L. Zhou, C. Masouros, A. Li, W. Luo, A. Petropulu, Toward

dual-functional radar-communication systems: Optimal waveform design, IEEE Transactions on Signal Processing (2018).

[22] A. Ali, N. Gonzalez-Prelcic, R. Heath, A. Ghosh, Leveraging sensing at the infrastructure for mmwave communication, IEEE Communications Magazine (2020).

[23] P. Kumari, S. Vorobyov, R. Heath, Adaptive virtual waveform design for millimeter-wave joint communication–radar, IEEE Transactions on Signal Processing (2020).

[24] C. Sturm, W. Wiesbeck, Waveform design and signal processing aspects for fusion of wireless communications and radar sensing, Proceedings of the IEEE (2011).

[25] P. Kumari, J. Choi, N. González-Prelcic, R. Heath, Ieee 802.11ad-based radar: An approach to joint vehicular communication-radar system, IEEE Transactions on Vehicular Technology (2018).

[26] J. Zhang, X. Huang, Y. Guo, J. Yuan, R. Heath, Multibeam for joint communication and radar sensing using steerable analog antenna arrays, IEEE Transactions on Vehicular Technology (2019).

[27] D. Ma, N. Shlezinger, T. Huang, Y. Liu, Y. Eldar, Joint radar-communication strategies for autonomous vehicles: Combining two key automotive technologies, IEEE Signal Processing Magazine (2020).

[28] C. Ozkaptan, E. Ekici, O. Altintas, Adaptive waveform design for communication-enabled automotive radars, IEEE Transactions on Wireless Communications (2022).

[29] C. Ozkaptan, E. Ekici, C. Wang, O. Altintas, Optimal precoder design for mimo-ofdm-based joint automotive radar-communication networks, in: WiOpt'21), 2021.

[30] C. Ozkaptan, E. Ekici, C. Wang, O. Altintas, Neighbor discovery and mac protocol for joint automotive radar-communication systems, in: 2021 IEEE 94th Vehicular Technology Conference, 2021.

[31] C. Wang, O. Altintas, C. Ozkaptan, E. Ekici, Multi-range joint automotive radar and communication using pilot-based ofdm radar, in: 2020 IEEE Vehicular Networking Conference, 2020.

[32] C. Ozkaptan, E. Ekici, O. Altintas, C. Wang, Ofdm pilot-based radar for joint vehicular communication and radar systems, in: 2018 IEEE Vehicular Networking Conference, 2018.

[33] C. Wang, C. Ozkaptan, E. Ekici, O. Altintas, Poster: Multi-carrier modulation on fmcw radar for joint automotive radar and communication, in: 2018 IEEE Vehicular Networking Conference, 2018.

[34] E. Cianca, M. De Sanctis, S. Di Domenico, Radios as sensors, IEEE Internet of Things Journal (2017).

[35] O-JRC: Open-source Joint Radar-Communication Testbed, `https://github.com/mmWave-MIMO-Testbed/O-JRC` (2024).

[36] Ieee standard for information technology–telecommunications and information exchange between systems–local and metropolitan area networks–specific requirements-part 11: Wireless lan medium access

control (mac) and physical layer (phy) specifications amendment 3: Enhancements for very high throughput in the 60 ghz band, IEEE Std 802.11ad-2012 (2012).

[37] Y. Deng, X. Zhou, A. Ghosh, A. Gupta, N. Shroff, Interference constrained beam alignment for time-varying channels via kernelized bandits, in: WiOpt'22, 2022.

[38] A. Krause, C. Ong, Contextual gaussian process bandit optimization, in: NeurIPS'11, 2011.

[39] GPy, GPy: A gaussian process framework in python, `http://github.com/SheffieldML/GPy` (since 2012).

[40] B. Sun, J. Feng, K. Saenko, Return of frustratingly easy domain adaptation, in: AAAI, 2016.

Declaration of interests

☐ The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☒ The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

Eylem Ekici reports financial support was provided by National Science Foundation. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.