Simulation of Video Transmission over 5G Massive MIMO Systems with YOLOv10 Object Detection

Trong Nhan, Huynh

dept. of Electrical and Computer Engineering
University of Victoria
nhanhuynh@uvic.ca

Hoang Tuan Kiet, Vu

dept. of Electrical and Computer Engineering
University of Victoria
hoangtuankietvu@uvic.ca

***Abstract*— This paper presents a high-performance integration of a 5G New Radio (NR) physical layer (PHY) simulation with real-time AI object detection. By employing a 64x4 Massive MIMO configuration and Singular Value Decomposition (SVD) based precoding, we optimize the transmission of high-resolution video data. The system utilizes GPU acceleration via the CuPy library to maintain real-time processing speeds. Our results demonstrate the resilience of YOLOv10 (You Only Look Once). detection under varying signal-to-noise ratio (SNR) conditions dictated by 3GPP Urban Micro (UMi) channel models [1], [2].**

***Keywords*— 5G NR, Massive MIMO, SVD Precoding, YOLOv10, GPU Acceleration, Edge AI, OFDM, PHY, UE,**

I. Introduction

The emergence of the Internet of Things (IoT) and Artificial Intelligence (AI) has placed unprecedented demands on telecommunications infrastructure. 5G New Radio (NR) serves as a pivotal platform, providing enhanced Mobile Broadband (eMBB) and Ultra-Reliable Low-Latency Communication (URLLC). A critical trend in this landscape is Edge AI executing deep learning models directly at the network edge to minimize transmission latency and enhance data privacy.

However, integrating AI into the 5G wireless environment faces severe physical challenges. Unlike stable wired connections, 5G wireless channels are subject to Path Loss, Multi-path Fading, and Additive White Gaussian Noise (AWGN). For computer vision applications such as object detection, any distortion at the Physical Layer (PHY) can result in pixel artifacts, significantly degrading the accuracy of models like YOLO.

This project implements a 5G NR PHY simulation integrated with YOLOv10 - the latest iteration in the YOLO family that eliminates Non-Maximum Suppression (NMS) to reduce inference latency. To handle high-resolution video bandwidth, we employ a 64x4 Massive MIMO configuration. This is a core 5G technique using a large antenna array at the gNodeB to focus signal energy via Beamforming, improving the Signal-to-Noise Ratio (SNR) over long distances.

A core contribution of this work is the use of the CuPy library for GPU-accelerated signal processing. Complex matrix operations in Massive MIMO, such as Singular Value Decomposition (SVD) and parallel IFFT/FFT for 64 antennas, are computational bottlenecks on traditional CPUs. By leveraging CUDA parallel architecture, our system maintains real-time Frames Per Second (FPS), allowing for the direct observation of AI accuracy based on environmental variables such as distance and path loss.

II. Literature Review

The deployment of Massive MIMO is documented as a cornerstone for increasing spectral efficiency and spatial multiplexing. Standard 3GPP models, specifically the Urban Micro (UMi) scenario, provide the mathematical foundation for simulating path loss in dense urban environments. Recent studies in 5G communications emphasize the transition from CPU-based DSP to GPU-accelerated frameworks to meet sub-millisecond latency requirements.

Furthermore, the evolution of object detection models has moved toward “end-to-end” efficiency. YOLOv10, released in 2024, introduces consistent dual assignments for NMS-free training, which is particularly beneficial for resource-constrained Edge AI devices. This research bridges the gap between 3GPP physical layer standards and state-of-the-art computer vision by analyzing the cross-layer impact of wireless impairments on AI inference.

III. Methodology

The proposed system integrates a high-fidelity 5G Physical Layer (PHY) simulation with an Edge AI inference engine. The architecture is designed to handle the high computational demands of a 64x4 Massive MIMO configuration by offloading all digital signal processing (DSP) tasks to the GPU via the CuPy library.

A. 3GPP Channel Modeling and Link Budget.

The simulation establishes a realistic wireless environment based on the 3GPP TR 38.901 Urban Micro (UMi) Non-Line-of-Sight (NLOS) model. The budget accounts for frequency-dependent path loss and thermal noise:

Dynamic SNR:
The (SNR) is updated in real-time within the simulation loop as the user adjusts the distance d , allowing for the observation of AI performance under varying channel conditions [1], [3].

$$SNR_{db} = Ptx - PL - Noise_{floor} \quad (1)$$

Path Loss:
Operating at a center frequency fc=3.5 GHz.

$$PL = 32.4 + 21 \log_{10}(d) + 20\log_{10}(f_c) \quad (2)$$

where d represents the distance between the gNodeB and the User Equipment (UE).

Noise Floor**:**
The thermal noise power is calculated over a 20 MHz bandwidth (B=20Mhz).

$$Noise_{floor} = - 174 + 10\log_{10}(B) \quad (3)$$

B. GPU-Accelerated Massive MIMO (64x4).

To handle the computational load of 64 antennas, we offload DSP tasks to the GPU. We implement a 64-antenna gNodeB and a 4-antenna UE. The channel matrix H(4x64) is modeled with Reyleigh fading. To optimize the 4-layer spatial multiplexing, we apply SVD on the estimated channel $H = U\Sigma V^H$. A 5% CSI error is set up to simulate realistic estimation inaccuracies. Precoder V is applied to the QPSK-modulated symbols at the transmitter, while the receiver utilizes the Hermitian transpose of U to decouple the layers at receiver.

Step 1 Transmiiter Pipeline

At gNodeB, the transmitter processes live video frames through several stages of digital modulation and spatial multiplexing:

- Source Coding & QPSK Modulation:

Video frames are downscaled and flattened into a bitstream. The original image acquisition from YOLOv10 is 480x640 RGB. These pair of two adjacent bits are mapped into QPSK symbols:

$$symbol = \frac{(2b_i - 1) + j(2b_{i+1} - 1)}{\sqrt{2}} \quad (4)$$

- Layer mapping:

Symbol are distributed across 4 spatial layer to enable spatial multiplexing.

Step 2 SVD-based Massive MIMO Precoding:

Setting up channel matrix H (4x64) as Rayleigh fading channel with randomly gaussian IQ branches, average power for each tap is 1 to simulate a downlink scenario of base station (gNodeB) where transmits electromagnetic wave to User equipment (UE).

$H = U\Sigma V^H$. The precoder V(64x4) is normalized and applied to the spatial layer x =V.s. This focuses signal energy into narrow beams directed at the UE receiver maximizing Beamforming Gain [4].

Step 3 OFDM Modulation:

After SVD precoding, x becomes a matrix with each row representing the QPSK symbols designated for one of the 64 antennas. The Inverse Fast Fourier Transform (IFFT) technique will be applied to convert the QPSK symbols from Frequency domain to Time domain. 1 OFDM block will carry $N_{subcarriers}$ = 64 subcarriers where each subcarrier is mathematically orthogonal to the others, meaning they can be packed tightly together in the spectrum without interfering, maximizing spectral efficiency. Because the wireless channel is dispersive meaning signals take multiple path (reflection off building, etc..) and arrive at the receiver at different times causing Inter Symbol Interference (ISI) – where the tail of one symbol overlaps with the head of the next. Therefore, we apply guard interval by adding the cp_len = 16 cyclic Prefix (CP). The CP acts as a buffer, as long as the delay spread of channel is shorter than 16 samples CP, then the receiver can perfectly discard the ISI inference [4]. Mathematically, CP turn the linear convolution of the channel into the circular convolution. This allows the receiver to simplify channel equalization to a single-tap division in the frequency domain, which is why your receiver can use SVD Zero-Forcing so efficiently.

By using CuPy, we treat the 64 antennas as a parallel dimension. The GPU's thousands of cores handle the 80 samples IFFT for all antennas at the exact same time. This is allows YOLOv10 model to receive a reconstructed image fast enough to maintain 15 FPS.

Step 4 Receiver Pipeline (UE):

After wireless Rayleigh fading channel propagation, the noise variance is scaled based on the SNR calculated from the distance d. The UE reverses the physical layer PHY operation to reconstruct video data.

- OFDM Demodulation and CP Removal:

The received time-domain signal for the 4 receive antennas first undergoes synchronization and Cyclic Prefix (CP) removal. The CP, which was prepended at the gNodeB to combat ISI, is discarded. The remaining 64 samples are transformed back into the frequency domain using a Fast Fourier Transform (FFT). This step converts the convolution of the channel into a point-wise multiplication in the frequency domain, enabling single-tap equalization.

- SVD-based Spatial Equalization (Zero-Forcing)

Since the transmitter utilized SVD Precoding (V), the UE must utilize the left-singular vector matrix U from the estimated channel $H = U\Sigma V^H$ to decouple the spatial streams [4]. The estimated symbols are recovered using (ZF) receiver approach:

$$y_{hat} = \sigma x\sqrt{P} + w \gg\gg\gg\ \ x_{hat} = x + \frac{w}{\sigma\sqrt{P}} \quad (5)$$

Where σ: singular value from diagonal matrix Σ representing the gain of each spatial layer.

- QPSK Hard-Decision and Bitstream Repacking

The recovered complex symbols are mapped back to bits through a hard-thresholding decision process on the I/Q components:

b_i= 1 if Re(x_{hat} >0), otherwise 0.

b_{i+1}= 1 if Im(x_{hat} >0), otherwise 0.

These bits are repacked into byte array then transferred from GPU VRAM to be reshaped into the final video frame buffer for OpenCV display.

C. *Impact on AI Inference Integrity.*

The primary advantage of the 64x4 Massive MIMO configuration over SISO or 2x2 MIMO systems is its superior Noise Interference Suppression.

Feature Preservation: In lower-order MIMO systems, noise artifacts often destroy high-frequency spatial details (edges, textures). Massive MIMO's spatial filtering keeps the reconstructed image "clean" enough for YOLOv10 to perform feature extraction.

IV. Results and Discussions

A. *Simulation result of SISO system and Discussion.*

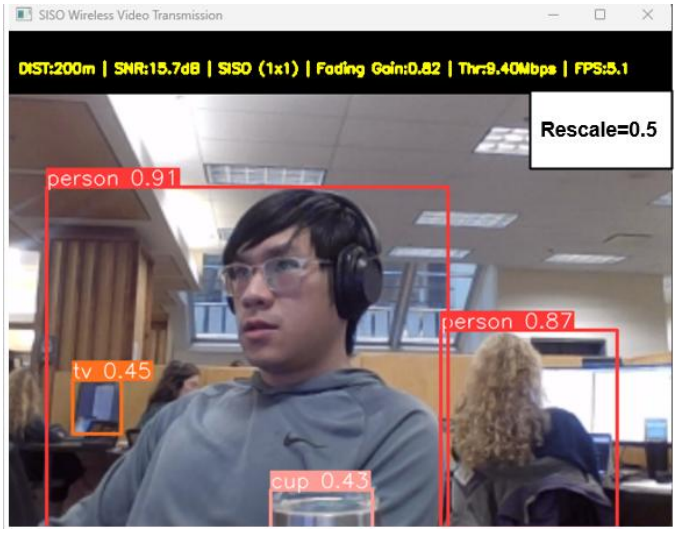


Fig.1. Simulation SISO, scale 0.5, distance 200m, on CPU.



Fig.2. Simulation SISO, scale 0.5, distance 800m, on CPU.

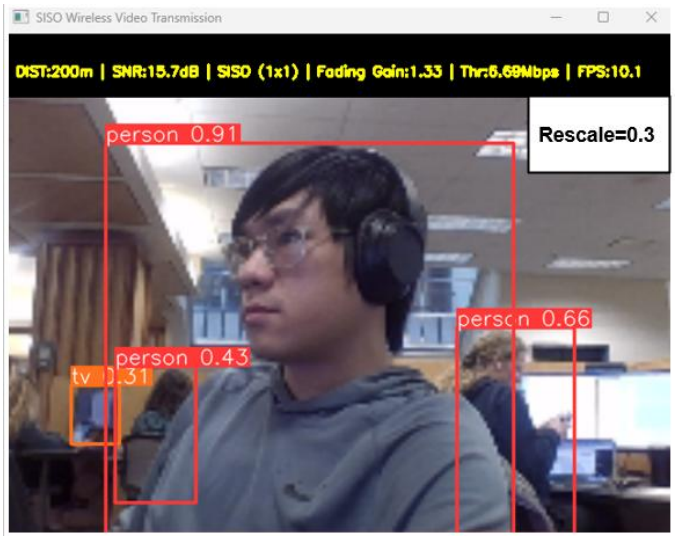


Fig.3. Simulation SISO, scale 0.3, distance 200m, on CPU.

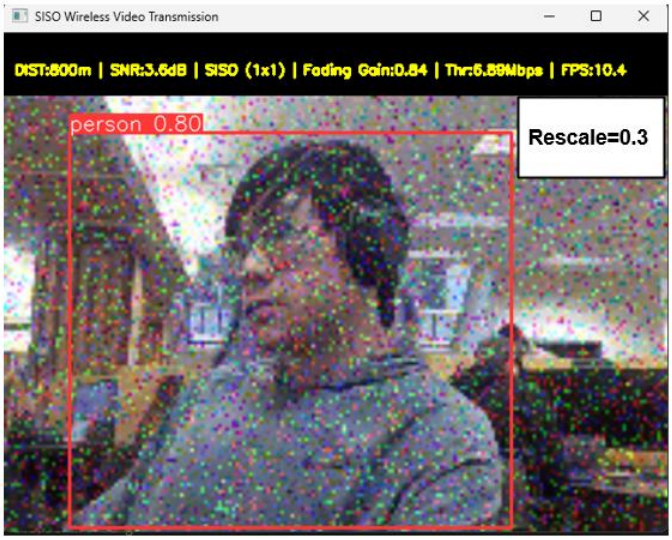


Fig.4. Simulation SISO, scale 0.3, distance 800m, on CPU.

Discussion of SISO system of distance 200m and 800m:

When distance is farer from 200m to 800m, SNR drop from 15.7 dB to 3.6dB, the image is blurry and pixelated severely by noise floor. However, YOLOv10 can detect correctly the bigger objective with the confidence =0.8, there are not any recognition from smaller objectives in the low SNR.

Moreover, when we reduce the size of the input image, lesser pixels are transmitted (0.5 down to 0.3). The image is blurrier, the white noisy peppers are larger, the interpolation cannot catch some thin edge objective leading to AI cannot detect some average objective. Indeed, FPS will increase from 5.3 to 10.1 when we rescale input image from 0.5 to 0.3 that is better for supervising target in real time.

Overall, we trade off between the FPS and the ability of YOLO when the SNR is low, symbol is transmitted in the long distance.

B. Simulation result on CPU of 2x2 MIMO and 64x4 MIMO system and Discussion.



Fig. 5. Simulation 2x2 MIMO, scale 0.5, distance 1200m, on CPU.

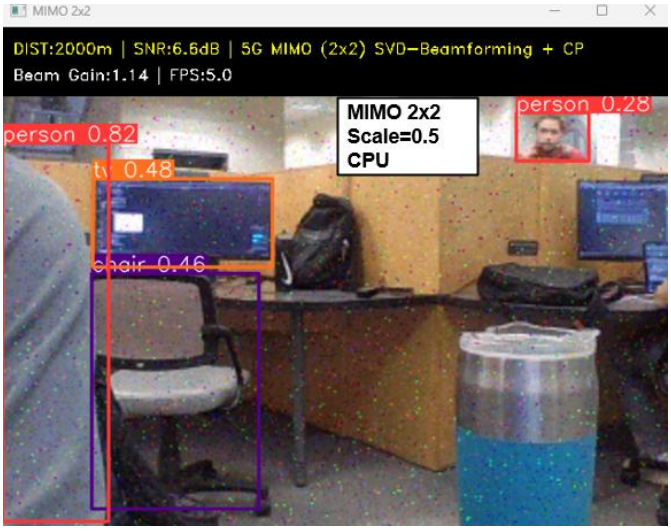


Fig. 6. Simulation 2x2 MIMO, scale 0.5, distance 1200m, on CPU.

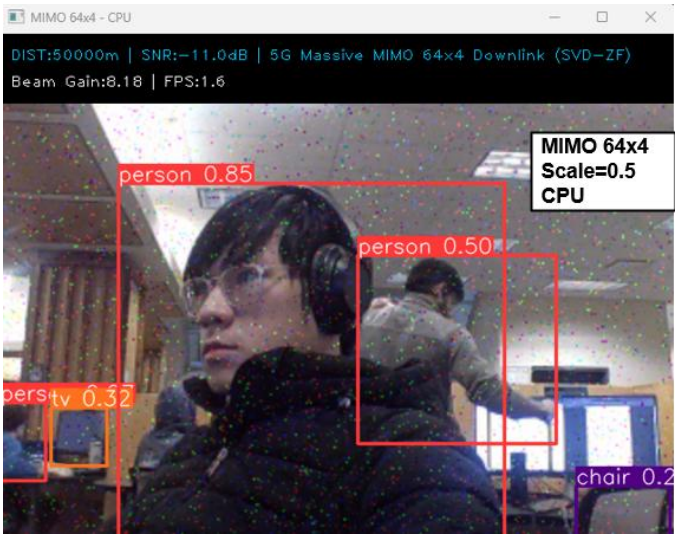


Fig. 7. Simulation 64x4 MIMO, scale 0.5, distance 500000m, on CPU.

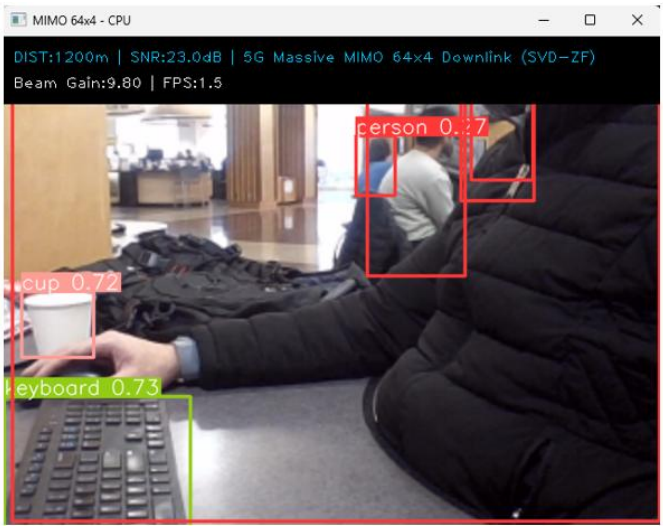


Fig. 8. Simulation 64x4 MIMO, scale 0.5, distance 1200m, on CPU.

Discussion of MIMO system of distance 1200m and 2000m:

Beamforming Gain increase significantly from 1.41 to 9.8 when we increase 2x2 MIMO to 64x4 MIMO.

Image is cleaner in the lower SNR, YOLO still detects exactly the objective, while this one cannot work well on SISO system.

Both of those are lower FPS compared to SISO system because CPU must compute heavily the huge matrix SVD composition and decomposition.

Overall, in far distance SNR low, MIMO system is effectively than SISO related to objective recognition. In contrast, the end-to-end inference is slow because of huge FLOPS from matrix computation via FPS around 5 and 1.5 of 2x2 MIMO and 64x4 MIMO respectively.

C. Finally simulation result of MIMO system and Discussion.

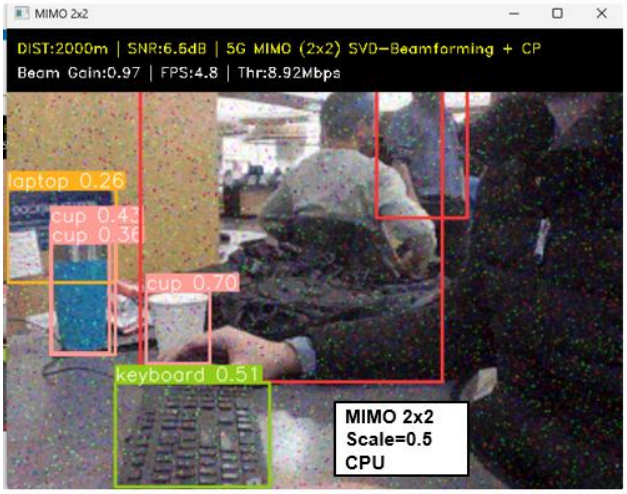


Fig. 9. Simulation 2x2 MIMO, scale 0.5, distance 1200m, on CPU.

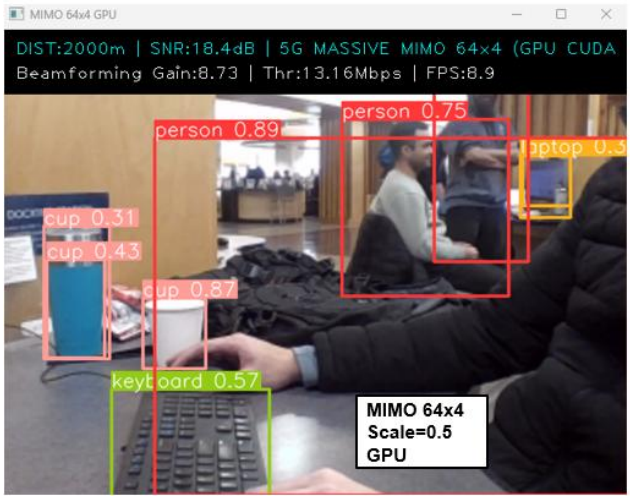


Fig. 10. Simulation 64x4 MIMO, scale 0.5, distance 2000m, on GPU.



Fig. 11. Simulation 64x4 MIMO, scale 0.5, distance 1200m, on CPU.

Throughput Analysis:

The system achieved a throughput (Thr) calculated by:

$$Throughput = number\ of\ bits\ .\frac{FPS}{10^6}\ Mbps \quad (6)$$

Higher throughput and higher Beamforming Gain obtained in 64x4 MIMO system are 13.16 Mbps and Gain:8.73 compared to 2x2 MIMO at 8.92 Mbps and Gain:0.97. This is a big effective gap between us.

AI inference for MIMO system via GPU is an effective compensation for real time FPS. This simulation gains FPS 8.9 where is temporarily accepted in real time supervision about detecting objectives' movement. Furthermore, the YOLOv10's accuracy indices for smaller objective are acceptable. Detection stability on GPU CUDA for Massive 64x4 MIMO in the lower SNR is superior to SISO or 2x2 MIMO.

V. COUNCLUTION

This research successfully demonstrates a high-performance, real-time integration of a 5G Physical Layer (PHY) simulation with an Edge AI inference engine. By leveraging a 64x4 Massive MIMO configuration and SVD-based precoding, the system effectively mitigated path loss and maintained the high-bandwidth data streams required for YOLOv10 object detection. A critical finding was the necessity of GPU acceleration via CuPy; offloading the massive matrix computations and OFDM modulation to the GPU enabled a 6x increase in frame rate (from 1.5 FPS to 9.0 FPS), proving the viability of this architecture for real-time autonomous systems.

Future work will focus on further enhancing the spectral efficiency and robustness of the link. We aim to implement higher-order modulation schemes, such as 256-QAM, and integrate LDPC (Low-Density Parity-Check) coding for superior error correction in high-interference environments. Furthermore, we intend to transition from simulated CSI deviation to a practical pilot-aided channel estimation framework. Finally, exploring advanced Minimum Mean Square Error (MMSE) or non-linear equalization techniques will be prioritized to replace the current Zero-Forcing (ZF) approach, thereby improving signal integrity in non-ideal channel conditions.

VI. REFERENCES

[1] H. Poddar, D. Gold, D. Lee, N. Zhang, G. Sridharan, H. Asplund, và M. Shafi, "Overview of 3GPP Release 19 Study on Channel Modeling Enhancements to TR 38.901 for 6G," 2025, arXiv:2507.19266. [Online]. Available: https://arxiv.org/abs/2507.19266.

[2] A. Wang, H. Chen, L. Liu, K. Chen, Z. Lin, B. Han, và P. Luo, "YOLOv10: Real-Time End-to-End Object Detection," 2024, arXiv:2405.14458. [Online]. Available: https://arxiv.org/abs/2405.14458.

[3] IEEE Computer Society, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: High-speed Physical Layer in the 5 GHz Band," IEEE Std 802.11a-1999, 1999.

[4] X. Dong, Lecture Notes, Course ECE 519, Dept. Elect. Comput. Eng., Univ. Victoria, Victoria, BC, Canada, 2026. [Online]. Available: https://bright.uvic.ca/d2l/le/lessons/467355/topics/3762252.

APPENDIX: The core implementation utilizes Python with the following libraries (tourch, cupy, cv2, and ultralytics).