NeRF^2: Neural Radio-Frequency Radiance Fields

Xiaopeng Zhao, Zhenlin An*, Qingrui Pan, Lei Yang*
Department of Computing, The Hong Kong Polytechnic University
{zhao,an,pan,young}@tagsys.org

ABSTRACT

Although Maxwell discovered the physical laws of electromagnetic waves 160 years ago, how to precisely model the propagation of an RF signal in an electrically large and complex environment remains a long-standing problem. The difficulty is in the complex interactions between the RF signal and the obstacles (e.g., reflection, diffraction, etc.). Inspired by the great success of using a neural network to describe the optical field in computer vision, we propose a neural radio-frequency radiance field, NeRF^2, which represents a continuous volumetric scene function that makes sense of an RF signal's propagation. Particularly, after training with a few signal measurements, NeRF^2 can tell how/what signal is received at any position when it knows the position of a transmitter. As a physical-layer neural network, NeRF^2 can take advantage of the learned statistic model plus the physical model of ray tracing to generate a synthetic dataset that meets the training demands of application-layer artificial neural networks (ANNs). Thus, we can boost the performance of ANNs by the proposed turbo-learning, which mixes the true and synthetic datasets to intensify the training. Our experiment results show that turbo-learning can enhance performance with an approximate 50% increase. We also demonstrate the power of NeRF^2 in the field of indoor localization and 5G MIMO.

KEYWORDS

Wireless Channel Prediction, Deep Learning, Wireless Localization, MIMO

1 INTRODUCTION

In free space, the propagation of an RF signal can be precisely modeled by Maxwell's equation and the Friis equation. However, when objects protrude into the first Fresnel zone defined by the TX and the RX locations, the free-space model fails [1]. As shown in Fig. 1, the whole radiance field is disturbed by absorption, reflection, diffraction, and/or scattering effects, making electromagnetic ray tracing become extremely complicated. Specifically, (1) reflection occurs on some quasi-specular surfaces (e.g., walls, ground, ceiling, etc.) and follows the law of reflection, i.e., the ray is reflected with the angle the same as the incident angle. (2) Diffraction is present at the edges of obstacles and follows the uniform theory of diffraction [2], i.e., a single incident ray upon the edge may create thousands of new rays on Keller cone. (3) The incident ray may also be scattered at small obstacles with rough surfaces where the ray may be reflected toward many angles.

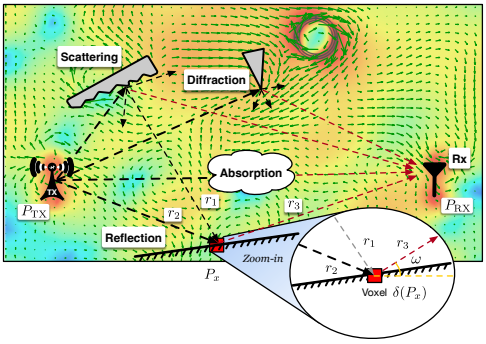


Fig. 1: Illustration of a radio-frequency radiance field. The ideal distribution of RF radiance is disturbed by the obstacles, which cause the RF signals to be reflected, scattered, diffracted, or absorbed.

To resolve the above issues, conventional algorithms conduct electromagnetic (EM) ray tracing with a given 3D scene model scanned by LiDAR [3, 4]. They simulate real EM rays to trace the path that an actual RF signal would take in the real world, which allows better simulation of how the signal interacts with the obstacles in the scene. The tracing quality highly depends on how deep the ray is traced and how realistic the scene model is built. Unfortunately, RF propagation depends not only on the locations and sizes of obstacles but also on their materials and physical characteristics. Precisely modeling the real-life world using the LiDAR technique is a nearly impossible task in practice.

Recently, Google researchers proposed the neural radiance fields (NeRF) [5] to address the ray tracing of light. As one of the important breakthroughs in computer vision, the NeRF has demonstrated great successes in the view synthesis [5–7], 3D model rendering [8, 9], and immersive street view [10, 11]. A large number of demos can be found at [12]. The basic idea of NeRF is to capture a few images in the scene from different angles as input and then train an MLP (i.e., a fully connected class of feedforward artificial neural network) to fit the optical radiance field. The NeRF regards each pixel of an image as a result of one ray tracing, which reflects the feature of the scene-dependent optical radiance field. After training well with a few images, NeRF can exactly predict the result of ray tracing from any other direction and further synthesize an entire image from a given observing direction.

*Zhenlin An and Lei Yang are corresponding authors.

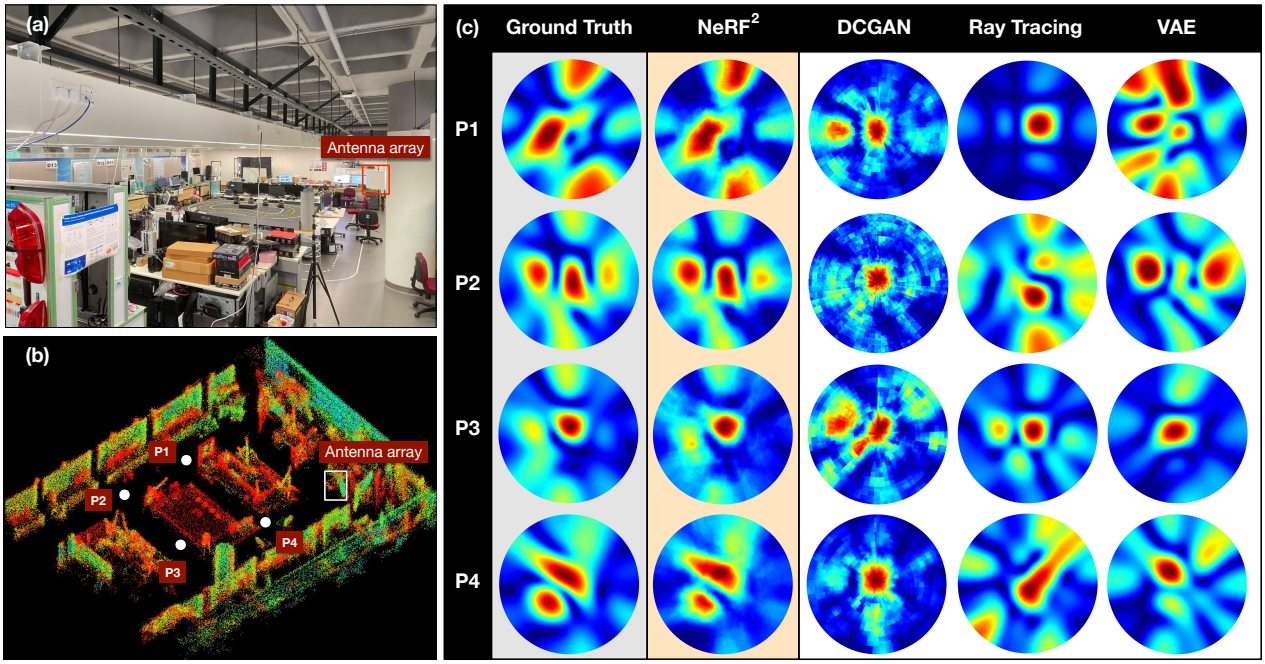


Fig. 2: Synthesis of spatial spectrums. The spatial spectrums, also known as the multipath profile, show how strong the RF signal is from a particular direction composed of the azimuthal and elevation angles. The formal definition refers to Eqn. 12. (a) shows the scene, in which the TX may be located at any position, but the RX equipped with a 4×4 antenna array is fixed at a corner; (b) shows the point cloud created by LiDAR, which is only used for the conventional ray-tracing algorithm; (c) compares the synthesis spectrums generated by different algorithms when the TX is located at four different positions. The ground truth is obtained using the antenna array.

Based on the fact that light is a kind of electromagnetic wave, we propose the Neural Radio-Frequency Radiance Fields (NeRF^2), which extends the neural radiance fields from optics to electromagnetism. Similarly, NeRF^2 represents scenes as neural radiance fields by optimizing an underlying continuous volumetric scene function using a sparse set of input signal measurements. Specifically, NeRF^2 can predict what and how an RF signal is received when the transmitter (TX) is located at a known position. To intuitively understand the capability of NeRF^2, we show an example in Fig. 2. Four algorithms are used to synthesize (or predict) the spatial spectrums (i.e., multipath profile) that reflect how the RX received the signal from different directions when the TX is located at four different positions. Evidently, the prediction of NeRF^2 is most similar to the ground truth, which is generated through the true signals received by the antenna array. More examples can be found in our demo video https://xpengzhao.github.io/NeRF2.

Yet, translating the NeRF to the RF domain requires addressing many challenges. First, the RF signals operating at UHF or microwave spectrum (e.g., 800MHz, 2.4GHz, or 6GHz) are more prone to be reflected, diffracted, and scattered because their frequencies are far lower than the visible light. Second, only the amplitude (i.e., light strength) is considered in the optical NeRF. The phase of light is neglected because it repeats every 600–800 nm propagation. By contrast, the phase cannot be disregarded anymore in cm- or mm-wavelength RF signals for its crucial role in constructive or destructive superimposing owing to the multipath effects. Third, the measurement of visible light is taken by using a million-pixel camera, but an RF RX is usually equipped with either a single antenna or a small antenna array because of the size limitation (i.e., the size of an antenna is proportional to the wavelength). To address these issues, we first update the physical tracing model to fit the characteristic of RF signals. We then introduce the phase apart from the amplitude to set up a complex-valued MLP for NeRF^2. We finally propose two training approaches for the single-antenna and array-antenna receivers.

As a physical-layer neural network, NeRF^2 can promote the performance of many key RF applications, such as indoor localization, channel estimation, wireless power transmission, 5G base station deployment, wireless sensing, and so on. To meet the various application-layer demands, we propose *turbo-learning*, which takes advantage of the physical nature of NeRF^2 to generate a vast number of synthetic datasets in accordance with the physical model. This synthetic dataset is mixed with the true dataset together to intensify the training of application-layer artificial neural networks (ANNs). Turbo-learning not only allows ANNs to collect fewer training datasets but also promises a high-level learning accuracy.

Summary of results. We use a 4×4 antenna array as the RX to predict the spatial spectrums (i.e., multipath profile) in the micro benchmark. The results show that the median similarity of the spatial spectrums generated by NeRF^2 and the ground truth is up to 82%, which is far higher than other

synthetic algorithms. We also use light-of-sight (LOS) AoA estimation as an application to quantify the benefit of turbo-learning. The experiment results show that the accuracy can be raised by 47.9% using only a 10% true training dataset. Our large-scale experiments in which we collect RF signals at 530K positions in 14 scenes further verify the great power of turbo-learning. Overall, the average AoA accuracy is improved by 49.5% across 14 scenes.

Field Study. We present two field studies to demonstrate how the NeRF^2 benefits two classical applications: (1) BLE Localization. Pinpointing an RF device indoors is challenging [13–31], particularly when the line-of-sight propagation is blocked. Similar to the problem of ray tracing in graphics, localization accuracy can be improved greatly if the propagation of RF signals is deeply traced using NeRF^2. Our experiment results show that the NeRF^2 enabled turbo-learning can reduce the median error by 50% and the standard variance by 40%. (2) 5G MIMO. Massive MIMO (i.e., 5G network) heavily relies on accurate channel state information (CSI) for beamforming, i.e., the base stations must know the downlink wireless channel from their antennas to every client devices [32]. To this end, the client devices should transmit the channel estimation results back to the base station and thus cause huge overheads. The demand channel estimation can be exactly met by the NeRF^2 in that it can predict the CSI at any position derived from the learned radiance fields. Our experiment results show that NeRF^2 is 5.97 dB better SNR and 4.32 dB better SINR than the state-of-the-art work in terms of channel prediction and MU-MIMO performance.

Contribution. Our contributions are summarized as follows.

- We translate the NeRF from optics to the RF domain. Specifically, (1) We update the neural networks for complex-valued input parameters; (2) we replace the light propagation model with the Friis equation; (3) we invented the single-antenna and multiple-antenna-based electromagnetic ray-tracing approaches.
- We propose the NeRF^2 enabled turbo-learning. The benefit of turbo-learning is not only limited to the performance enhancement but also, more importantly, addresses the pain point of deep learning – significant reduction of the quantity of training dataset and the corresponding workload on collecting dataset.
- We conduct real-life field studies in terms of RFID, BLE, and 5G systems. The proposed turbo-learning is evaluated on indoor localization and FDD massive MIMO channel prediction.

2 NeRF^2 DESIGN

Following a common practice of NeRF, we make the following similar assumptions: (1) The receivers (e.g., 5G base station, Bluetooth station, and RFID reader) are located at known positions, whereas the transmitter (e.g., smartphones, iBeacon, and RFID tags) are movable within a limited range. (2) Major obstacles (e.g., buildings, walls, and furniture) in each scene remain unchanged. (3) The moving obstacles may introduce temporary minor perturbance on the radiance field, which can be smoothed through the upper-layer filtering algorithm (e.g., Kalman filter), so their influence is not considered.

At the heart of NeRF^2 is the two key components: the neural radiance network and the ray tracing algorithm:

- **Neural Radiance Network**: This network is used to represent the scene and the radiance field using two MLPs. It can predict how RF signals are distributed in the scene.
- **Ray Tracing**: Given the RF distribution, we must trace the signals transmitted from all potential directions to know what signal is received at the RX.

In this section, we elaborate on the details of the above two components and present the training approaches. Finally, we present NeRF^2 enabled turbo-learning.

2.1 Neural Radiance Network

To model the radiance field, we discretize the scene of interest into a finite number of small 3D voxels in the space. The Huygens–Fresnel principle suggests that when the original RF signal arrives at it from all possible paths, a voxel can be considered as a new radiance source retransmitting the RF signal. To better understand this principle, we show an example voxel in the zoom-in of Fig. 1. Let the subscript x denote an arbitrary voxel in the scene. The voxel x at position P_x receives the RF signal from two paths r_1 and r_2, and it becomes a new TX that retransmits the RF signal along the path r_3 to the RX. In our model, each voxel is described with three properties, the position $P_x = (X, Y, Z)$, the attenuation $\delta(P_x) = \Delta a(P_x)e^{\mathbf{J}\Delta\theta(P_x)}$ and the retransmitted RF signal $S(P_x)$. The $\delta(P_x)$ is a material-dependent variable, which indicates that the amplitude is degraded by $\Delta a(P_x) = |\delta(P_x)|$ and the phase is rotated by $\Delta\theta(P_x) = \angle\delta(P_x)$ if an RF signal passes through the voxel at P_x. As a new RF transmitter, the voxel at position P_x retransmits a new complex-valued signal S_x, i.e., $S(P_x) = a(P_x)e^{\mathbf{J}\theta(P_x)}$ where $\theta(P_x)$ and $a(P_x)$ are the initial phase and the initial amplitude. The voxel cannot be simply modeled as an omnidirectional radiance source. Instead, it may radiate EM waves unevenly in angles. To address this issue, we introduce another variable called measuring direction $\omega = (\alpha, \beta)$ where α and β are the azimuthal and elevation angles. As shown in the zoom-in of Fig. 1, the voxel is located in the direction ω relative to the RX position.

NeRF^2 aims to predict the RF signal S_x retransmitted from the voxel at P_x toward the direction ω when given the TX's position P_{TX}. To do so, we take advantage of the neural

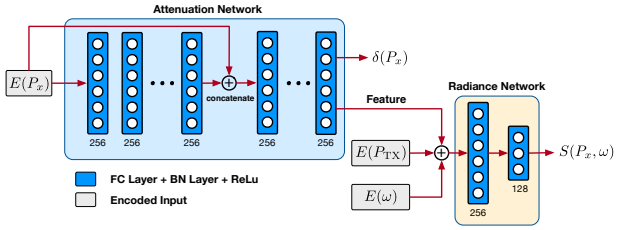


Fig. 3: Architecture of the neural network. NeRF^2 consists of two MLPs, the attenuation network, and the radiance network. The attenuation network can predict the attenuation δ of any voxel. Given the TX position and a measuring direction, the radiance network can predict the signal transmitting from an arbitrary voxel.

network to fit the radiance field. Formally, the radiance field **F** is represented as follows:

$$\mathbf{F}_\Theta : (P_{\text{TX}}, P_x, \omega) \rightarrow \Big(\delta(P_x), S(P_x, \omega)\Big) \tag{1}$$

where Θ indicates the learnable neural network weights. Unlike the visual NeRF that assumes the ambient light remains unchanged, we introduce the position of the TX as an additional input because our transmitters (e.g., smartphones or IoT devices) are moveable. In this way, we can create a dataset for a scene by placing the TX at different and sufficient positions. The neural network contains two outputs. One is the attenuation $\delta(P_x)$ of the voxel at P_x, which is highly related to the voxel's physical characteristics. Another is the RF signal $S(P_x, \omega)$ retransmitted from the voxel at P_x toward the direction ω. Thus, the neural network represents not only the scene but also the RF distribution.

Network Architecture. To build the neural network, we adopt two MLPs: the attenuation network and the radiance network, as shown in Fig. 3. The attenuation property is highly related to the materials of the voxel and independent of the incoming signals, so we separate the attenuation network to predict the attenuation $\delta(P_x)$ as a function of the position P_x. The attenuation network is composed of eight fully connected layers (using ReLU activations and 256 channels per layer) and outputs $\delta(P_x)$ and a 256-dimensional feature vector. This feature vector is then concatenated with the RX direction ω related to P_x, and the TX position P_{TX}. The combination is passed to the radiance network, another two fully connected layers (using a ReLU activation and including 256 and 128 channels), which outputs the direction-dependent RF signal $S(P_x, \omega)$, which is retransmitted from the voxel along the direction ω. The network architecture is similar to the optical NeRF but differs in two aspects. First, the visual NeRF assumes that the location of TX (i.e., light source) remains unchanged, whereas our TX is moveable. Second, our two networks are complex-valued, considering both magnitude and phase.

Discussion. The radiance field is only relevant to the scene, including the obstacles and the position of TX, but irrelevant to the position of RX. One may be concerned about how to deal with the multiple reflections of the RF signal. The trick of NeRF^2 is that each voxel is considered as a new transmitter that "retransmit" a combined signal received from all possible paths. Such a model simplifies the subsequent calculation of ray tracing.

2.2 Electromagnetic Ray Tracing

To train the NeRF^2, a naive approach is to probe the RF signals at a vast number of RX positions. Evidently, this approach is unscalable in practice. The visual NeRF views each image of the scene as a result of ray marching [1], where each pixel reflects the intensity of the light propagated from a particular direction due to the pinhole model of cameras. Similarly, the signal received at the RX is a result of electromagnetic ray tracing, where the signal is a combination of signals transmitted from all possible directions. Next, we introduce how we trace the signal from a particular direction.

The propagation of an RF signal S from a transmitter (TX) to a receiver (RX) conforms to the Friis equation as follows:

$$R = H_{\text{TX}\rightarrow\text{RX}} S = a_{\text{TX}\rightarrow\text{RX}} e^{\mathbf{J}\theta_{\text{TX}\rightarrow\text{RX}}} S \tag{2}$$

where R is the received signal, $H_{\text{TX}\rightarrow\text{RX}}$ is the channel attenuation. Particularly, $a_{\text{TX}\rightarrow\text{RX}}$ and $\theta_{\text{TX}\rightarrow\text{RX}}$ are the amplitude degradation and the phase rotation caused by the distance from the TX to the RX. Mathematically, a direction ω related to the RX can be modeled as a ray, which starts from the RX and directs toward ω. The points on this ray are correspondingly described as follows:

$$P(r, \omega) = P_{\text{RX}} + r\omega \tag{3}$$

where r is the radial distance from the RX to the point on the ray. Note that $P_{\text{RX}} = P(0, \omega)$. The purpose of ray tracing is to accumulate the RF signals emitted from all voxels on this ray. Namely, the received signal at the RX from the direction ω can be expressed as:

$$R(\omega) = \int_0^D H_{P(r,\omega)\rightarrow P_{\text{RX}}} S\Big(P(r, \omega), -\omega\Big)\, \mathrm{d}r \tag{4}$$

In the above equation, $S(P(r, \omega), -\omega)$ represents the signal transmitted from the voxel at $P(r, \omega)$ to the RX at P_{RX}. Its transmission direction is opposite to the ray's direction, so we take the negative of ω in the equation. D is the maximal distance across the scene. The above equation suggests that the final signal received by the RX from the direction ω is the accumulation of the RF signals transmitted from all voxels on the ray, i.e., from $P(0, \omega)$ to $P(D, \omega)$. The $H_{P(r,\omega)\rightarrow P_{\text{RX}}}$ is the attenuation of the signal propagated from the point $P(r, \omega)$

[1] Ray tracing and ray marching are two rendering techniques in computer graphics. Ray tracing computes the resulting color by tracing rays and accounting for object interactions, while ray marching estimates the color and opacity of the scene by evaluating a function along a ray.

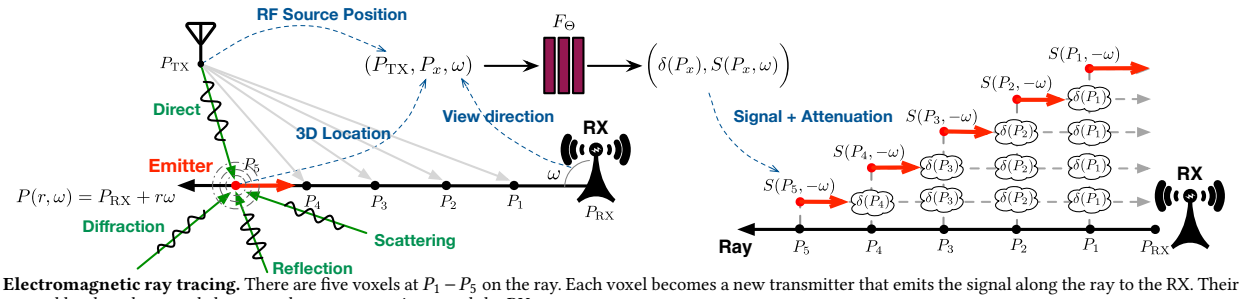


Fig. 4: Electromagnetic ray tracing. There are five voxels at $P_1 - P_5$ on the ray. Each voxel becomes a new transmitter that emits the signal along the ray to the RX. Their signals are attenuated by the other voxels between the new transmitters and the RX.

to the RX. It is defined as follows:

$$\begin{aligned} H_{P(r,\omega)\to P_{\text{RX}}} &= \prod_{\tilde{r}=0}^{r} \delta\big(P(\tilde{r},\omega)\big) \\ &= \left(\prod_{\tilde{r}=0}^{r} \Delta a_{P(\tilde{r},\omega)} e^{\mathrm{J}\Delta\theta_{P(\tilde{r},\omega)}}\right) \end{aligned} \tag{5}$$

The above equation means that the total attenuation equals the product of all attenuations caused by the voxels between the voxels at $P(r,\omega)$ and at $P(0,\omega)$, i.e., $0 \le \tilde{r} \le r$. To facilitate the calculation, we transform the above equation to an equivalent log-scale form as follows:

$$\begin{aligned} H_{P(r,\omega)\to P_{\text{RX}}} &= \exp\left(\ln\left(\prod_{\tilde{r}=0}^{r} \delta(P(\tilde{r},\omega))\right)\right) = \exp\left(\int_0^r \ln\Big(\delta\big(P(\tilde{r},\omega)\big)\Big)\,\mathrm{d}\tilde{r}\right) \\ &= \exp\left(\underbrace{\int_0^r \hat{\delta}\Big(P(\tilde{r},\omega)\Big)\,\mathrm{d}\tilde{r}}_{\text{Sum of attenuations}}\right) \end{aligned} \tag{6}$$

where $\hat{\delta}(\cdot)$ denotes the log-scale attenuation of $\delta(\cdot)$, which is defined as follows:

$$\hat{\delta}(P(\tilde{r},\omega)) = \ln \delta(P(\tilde{r},\omega)) \tag{7}$$

The log-scale form makes the product become a sum of all attenuations between two voxels, which greatly facilitates the calculation. Substituting Eqn. 6 into Eqn. 4, the signal coming from the direction ω is given by

$$R(\omega) = \int_0^D \underbrace{\exp\left(\int_0^r \hat{\delta}(P(\tilde{r},\omega))\,\mathrm{d}\tilde{r}\right)}_{\text{Attenuation Network}} \overbrace{S(P(r,\omega),-\omega)}^{\text{Radiance Network}}\,\mathrm{d}r \tag{8}$$

where the terms engaged in the previous part are predicted by the attenuation network, and the terms engaged in the last part are predicted by the radiance network. Briefly, the result of ray tracing along a direction is to aggregate the signals retransmitted from the voxels on this ray, each of which is regarded as a new source. Meanwhile, each transmission from a voxel must be attenuated by other voxels between the current voxel and the RX. Suppose there are N voxels on the ray, the ray tracing will take $O(N^2)$ aggregations.

To visually understand the ray tracing algorithm, we show an example in Fig. 4. Assuming the horizontal ray is from the RX to the left (i.e., $\omega = 180°$). On the ray, there are five voxels at P_1, P_2, P_3, P_4, and P_5, all of which are considered as new transmitters regardless of how these voxels are lighted up. As a result, the signal received by RX along the opposite direction of the ray (i.e., $-\omega$) is a combination of the five signals retransmitted from these five voxels. Particularly, the signal S_5 retransmitted from the voxel at P_5 is attenuated by the voxels at P_4, P_3, P_2, and P_1 in sequence. The accumulated attenuation equals $(\hat{\delta}_{P_1} + \hat{\delta}_{P_2} + \hat{\delta}_{P_3} + \hat{\delta}_{P_4})$. Similarly, the signals retransmitted from the voxels at P_1, P_2, P_3, and P_4 are attenuated by 0, $\hat{\delta}_{P_1}$, $\hat{\delta}_{P_1} + \hat{\delta}_{P_2}$, and $\hat{\delta}_{P_1} + \hat{\delta}_{P_2} + \hat{\delta}_{P_3}$, respectively.

Summary. The NeRF^2 does not completely depend on the neural network but combines the physical model and the statistic model. Specifically, ray tracing takes a well-known physical model of signal propagation, meanwhile, deep learning offers a statistical model of the complicated interactions between the RF signal and the surrounding obstacles.

2.3 Network Training

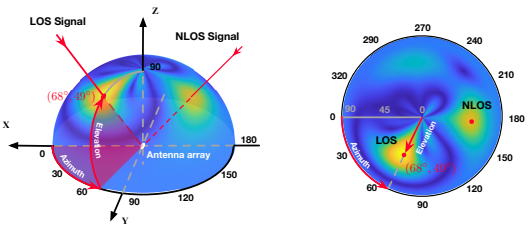
The previous describes the ray tracing algorithm, by which we can use the NeRF^2 to predict the signal received by the RX from a particular direction. Regarding which type of antenna is equipped at the RX, we introduce two types of training approaches.

2.3.1 Case I: Single-Antenna RX Model. We consider a simplified case where the RX is equipped with a single omnidirectional or a single directional antenna. Evidently, a single antenna has no discernibility in directions. Thus, the eventually received signal by the RX is a combination of the signals from all potential directions as follows:

$$\begin{aligned} R &= \int_\Omega \sqrt{G_{\text{RX}}(\omega)} R(\omega)\,\mathrm{d}\omega \\ &= \int_\Omega \int_0^D \exp\left(\int_0^r \hat{\delta}(P(\tilde{r},\omega))\,\mathrm{d}\tilde{r}\right) S(P(r,\omega),-\omega)\,\mathrm{d}\tilde{r} \end{aligned} \tag{9}$$

where $G_{\text{RX}}(\omega)$ indicates the antenna directivity (i.e., the gain that the antenna provides in each direction), and Ω denotes the directions that the antenna can cover. Let R and $\widetilde{R}$ denote the predicted signal by NeRF^2 with the ray tracing and the true received signal, respectively. We then can use the following loss function to train NeRF^2:

$$\mathcal{L} = |R - \widetilde{R}|^2 \tag{10}$$



(a) 3D Spatial Spectrum (b) 2D Spatial Spectrum

Fig. 5: Illustration of spatial spectrum

The loss function aims to reduce the gap between the true signal and the predicted one.

2.3.2 Case II: Multi-Antenna RX Model. Next, we consider the second case where the RX is equipped with a phased antenna array, which can form a very narrow beam and steer it to receive signals from a particular direction [33]. The RX can then discriminate the signal in directions. Suppose the antenna array is equipped with $K \times K$ elements uniformly. Choosing the element $A_{1,1}$ as a reference, we can compute the following relative power of projecting the received signal into the direction of $\omega = (\alpha, \beta)$:

$$\Psi(\omega) = \frac{1}{(K^2-1)} \left| \sum_{i=1}^{K} \sum_{j=1}^{K} w_{i,j}(\omega) \cdot e^{\mathrm{J}\Delta\widetilde{\theta}_{\mathrm{i,j}}} \right| \tag{11}$$

where $w_{i,j}(\omega) = e^{\mathrm{J}-\Delta\theta_{\mathrm{i,j}}}$ is the complex weight for steering a beam to a certain angle of (α, β). In the above, $\Delta\widetilde{\theta}_{i,j}$ is the phase difference computed by using the received signals at $A_{i,j}$ and $A_{1,1}$, whereas $\Delta\theta$ is their theoretical phase difference [34]. The sum aggregates the relative power across the (K^2-1) pairs of elements, i.e., $(A_{1,2}, A_{1,1}), (A_{1,3}, A_{1,1}), \cdots$. When $\Delta\widetilde{\theta}_{i,j}$ aligns with $\Delta\theta_{i,j}$, i.e., the signal comes from the direction of (α, β), the normalized relative power $\Psi(\alpha, \beta)$ should achieve the maximum. A heatmap can then be generated to show the relative power at N possible directions that the received RF signal might come from. We call such a 2D heatmap *spatial spectrum*, denoted by Ψ. The N is a custom parameter depending on the angle resolution. If the one-degree resolution is accepted, $N = 360 \times 90$, and the spatial spectrum is defined as follows:

$$\Psi = \begin{pmatrix} \Psi(0°, 0°) & \Psi(1°, 0°) & \cdots & \Psi(360°, 0°) \\ \Psi(0°, 1°) & \Psi(1°, 1°) & \cdots & \Psi(360°, 1°) \\ \vdots & \vdots & \vdots & \vdots \\ \Psi(0°, 90°) & \Psi(1°, 90°) & \cdots & \Psi(360°, 90°) \end{pmatrix} \tag{12}$$

Sometimes, the spatial spectrum is also called multipath profile [30] because it reflects how the signal comes from multiple directions. Fig. 5(a) shows the spatial spectrum in 3D where all directions are uniformly distributed; Fig. 5(b) shows the 2D spectrum by projecting the 3D onto the X-Y plane, in which the radial distance represents $\cos(\beta)$ so the elevation angle distributes non-uniformly.

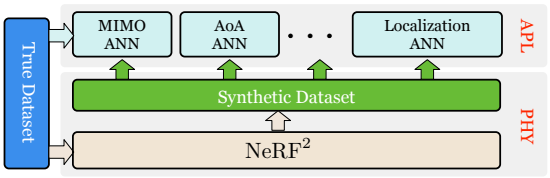


Fig. 6: Illustration of turbo-learning

It is spontaneous for NeRF^2 to predict the power of the signal coming from a particular direction and generate a predicted spatial spectrum Ψ' as follows:

$$\Psi'(\omega) = |R(\omega)|^2 \tag{13}$$

The relative power is directly proportional to the true power computed above. Even though a constant offset may exist between them, it does not affect the training of the network by using the following loss function:

$$\mathcal{L} = \sum_{\omega \in \Omega} |\Psi(\omega) - \Psi'(\omega)|^2 \tag{14}$$

The training aims to reduce the difference in power of the signal received from all possible directions.

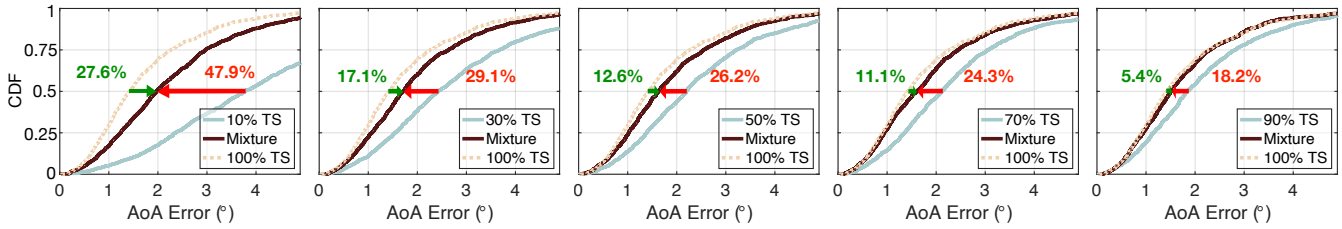
2.4 Turbo-Learning

As a physical-layer neural network, NeRF^2 describes the distribution of the radiance field. It cannot directly meet the application-layer demands, such as predicting the location of a receiver or beamforming parameters. Usually, extra neural networks are set up to address the specific application demand (e.g., MIMO ANN, AoA ANN, localization ANN, etc.). Instead, we employ NeRF^2 as a reinforcer to boost the performance of the application-layer ANNs. Fig. 6 illustrates this basic idea. First, we train NeRF^2 with the true training dataset. Second, NeRF^2 generates a vast number of synthetic dataset which meets the demand of the application-layer ANNs. Finally, we mix the true dataset and the synthetic dataset together to train the upper-layer ANNs. We call this training approach *turbo-learning*, i.e., applying more additional synthetic data to intensify the learning. Turbo-learning is also termed data augmentation in the field of data science. In the following sections, we will elaborate on turbo-learning case by case.

3 NeRF^2 IMPLEMENTATION

We train a separate NeRF^2 for each scene. This requires a dataset of RF signals or spatial spectrums captured in the scene, the corresponding locations of TX and RX, and scene bounds (i.e., Ω and D). The location-related parameters are acquired by a high-precision infrared positioning system named OptiTrack [35]. At each iteration, we make the following optimizations:

(1) Positional Encoding: NeRF^2 accepts two 3D positions and one 2D direction as the inputs. Following the practice from the optical NeRF, which uses the encoded positions,



(a) 10% TS + 90% SS (b) 30% TS + 70% SS (c) 50% TS + 50% SS (d) 70% TS + 30% SS (e) 90% TS + 10% SS

Fig. 7: CDFs of AoA error. The ANN is trained by the naive-learning (in light blue) and the turbo-learning (in dark red), respectively. We quantify the benefits of NeRF^2 with different mixture percentages.

we also raise the dimensions of the inputs to L using the subsequent encoding function:

$$E(x) = \left(\sin\left(2^0\pi x\right), \cos\left(2^0\pi x\right), \cdots, \sin\left(2^{L-1}\pi x\right), \cos\left(2^{L-1}\pi x\right)\right) \quad (15)$$

This function is applied separately to each of the three coordinate values in the P_{TX} or the P_x, and to the three components of the Cartesian direction unit vector ω. In our experiments, we set $L = 10$ for P_{TX} and P_x, and $L = 4$ for ω.

(2) Voxel Size: There is a trade-off in setting the size of the voxel. On the one hand, fine-grained voxels can provide higher resolution for NeRF^2 and accuracy in ray tracing. On the other hand, the number of voxels has a major impact on computational complexity. In our experiments, we set the size of the voxel to the 1/8 of the wavelength.

(3) Network Configuration: In each dataset, we randomly sample 80% samples to train the neural network and use the remained 20% for testing. We adopt a similar configuration as NeRF [5]. Specifically, the batch size is set to 4096. The Adam optimizer [36] is adopted. The learning rate begins at $3e-4$ and decreases exponentially to $3e-5$. Other hyper-parameters remain at default values (e.g., $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\varepsilon = 10^{-7}$). The network training for a single scene typically takes around 300-500k iterations to converge on a single NVIDIA 3080Ti GPU (about 10 hours). In contrast, testing for a single sample can be accomplished in about 0.2 seconds.

4 MICROBENCHMARK

We start with a microbenchmark experiment to provide insights into the working of NeRF^2 in this section.

4.1 Experimental Setup

We deploy a USRP-based RX equipped with a 4×4 antenna array. The RX operates at 915 MHz and targets to receive the signal backscattered from a moving RFID tag. The RFID tag is activated by a nearby reader (i.e., 1 m away) and repeatedly transmits `RN16` replies. Figs. 2-(a) and (b) show the photo of the scene and the corresponding 3D model (composed of a point cloud) scanned by LiDAR. This is a demo room full of reflectors such as metal desks, shelves, tables, computers, and so on. The dataset is created by placing the tag at random

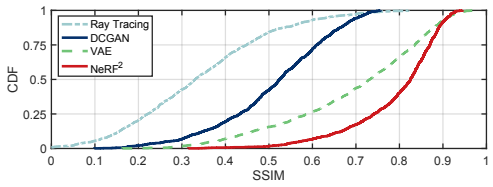


Fig. 8: SSIM Comparison

positions. For each position, the antenna array generates a spatial spectrum using Eqn. 11, which is represented by 360× 90 pixels from viewpoints sampled on the front hemisphere of the antenna array. We collected a total of 10 K data in this scene, where 8 K are used for training and 2 K for testing. We use the approach introduced in [37] to estimate the phase and amplitude of the received backscatter signals and employ Eqn. 14 as the loss function to train the neural radiance field.

4.2 Spectrum Synthesis

The goal of the original optical NeRF is to synthesize the photo of the scene taken from an arbitrary direction. Similarly, NeRF^2 possesses the ability to synthesize RF spatial spectrums when the TX is located at an arbitrary position. To visually understand such a purpose, we leverage NeRF^2 to synthesize the spatial spectrums that the antenna array receives. The synthesized spatial spectrum helps us intuitively verify whether the neural radiance field can successfully predict the signal propagations in the scene. We compare NeRF^2 with the other four baseline schemes.

- **Ground truth**: The true spatial spectrums are computed by using the Eqn. 12 across the real signals received by the antenna array. The spectrums are desired to peak at the LOS direction. Unfortunately, Fig. 2-(c) (1st column) shows possible multiple peaks because of the multipath propagations in such a complex environment.
- **RayTracing**: We employ the RayTracking toolbox in Matlab [38] to generate the spatial spectrums. Particularly, this toolbox requires importing the 3D model of the scene (i.e., Fig. 2-(b)). Given the locations of TX, the toolbox can predict the RF signals received by the RX.
- **Deep Convolutional Generative Adversarial Network (DCGAN)**. DCGAN is one of the most popular GANs wherein two models (i.e., generator and discriminator) are

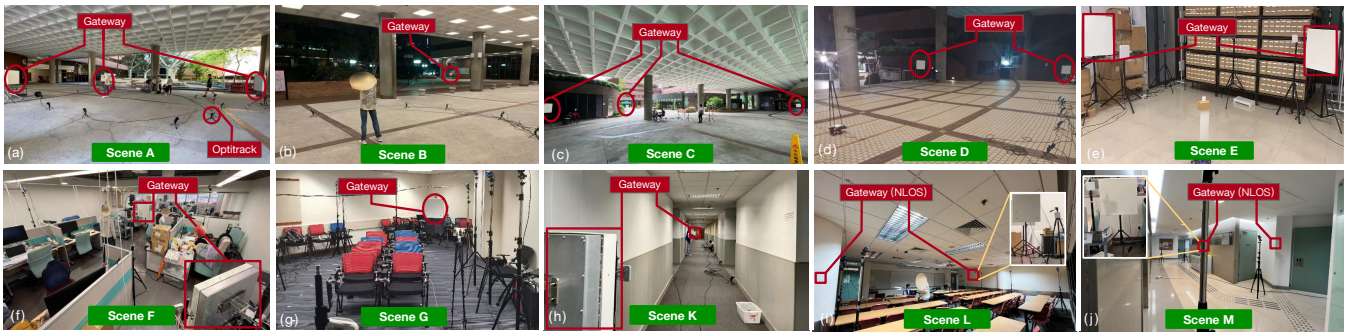


Fig. 9: Illustration of example scenes. (a)-(d) shows the semi-indoor environment, which is large-sized and semi-closed halls. (e)-(j) show the full-indoor environment.

trained simultaneously by an adversarial process. The generator model spawns "fake" images that look like the training images. The discriminator model determines whether an image is a real training image or a fake image from the generator. We view the predicted spatial spectrums as images and use DCGAN to learn and generate the spectrums with given TX's locations.

- **Variational Autoencoder (VAE)**. VAE is one of the famous generative models. It is used to resolve similar issues in wireless systems, such as liquid sensing [39] and channel estimation [32]. Adopting the similar architecture in FIRE [32], an encoder network learns the probability distribution of the training set in a lower dimensional latent space. Subsequently, the samples drawn from the decoder network are decoded to generate the data in accord with the learned distribution.

The results are shown in Fig. 2-(c), where the spatial spectrums are generated using the above schemes when the TX locates at four positions. Visually, the spatial spectrums generated by NeRF^2 are evidently more similar to the ground truth than other generative models. We further use a common criterion called *structural similarity index measure* (SSIM) to quantify the similarity of two images. Owing to the page limit, we omit the definition of SSIM but encourage the reader to refer to [40] for details. A higher SSIM indicates the two images are more similar. We randomly choose 100 positions to synthesize the spatial spectrums using the four algorithms. The CDF of the SSIM between those synthetic spatial spectrums and the ground truth is shown in Fig. 8. Particularly, the median SSIM of RayTracking, DCGAN, VAE, and NeRF^2 are 0.33, 0.52, 0.73, and 0.82, respectively, and their 90th percentiles are 0.56, 0.67, 0.89, and 0.91. The RayTracing underperforms because it is short of the material information, even though the geometric model of the scene is provided. DCGAN and VAE view spatial spectrums as a kind of signature related to the TX's location, so they do not really "understand" the rationale behind it. The outperformance of NeRF^2 is in the accurate model of radiance field in accordance with the underlying physical laws.

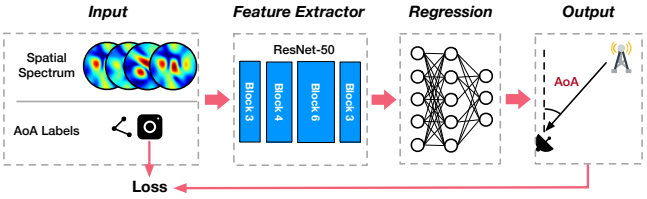


Fig. 10: Architecture of Angular Artificial Neural Network

4.3 Performance of Turbo-Learning

To quantify the benefits of NeRF^2, we apply turbo-learning to the AoA estimation, which aims to determine the direction of the line-of-sight propagation. The AoA is desired to be achieved at the peak of the spatial spectrum. Unfortunately, owing to the multipath propagations and the destructive superposition of signals, the peak deviates substantially from the true LOS direction. To address this issue, angular artificial neural networks (AANNs) are resorted to identifying the AoAs [34, 41–43]. Similar to the iArk [34], we set up an AANN based on the ResNet convolutional network [44], as shown in Fig. 10. The AANN accepts spatial spectrums in the image format and outputs the AoAs. In the AANN, a ResNet-50 network is adopted as the feature extractor, which is followed by a fully connected network for regression. We train the AANN using the following two approaches:

- **Naive Learning**. We use 10% of the true training dataset (TS, total 8 K) to train the AANN straightforwardly. In this approach, NeRF^2 is not involved.
- **Turbo-Learning**. We use the same 10% of the true dataset to train the NeRF^2. Then, we use the well-trained NeRF^2 to generate the rest 90% synthetic dataset (SS). Finally, the 10% true dataset and the 90% synthetic dataset (i.e., turbocharger) are mixed to train the AANN.

These two learning approaches fully use the same 10% of the true training dataset for the sake of fairness, i.e., *both hold the same amount of information from the true dataset.* We also use 100% training set to train the AANN as the baseline. The results are shown in Fig. 7(a). The median errors of naive learning and turbo-learning are 3.78° and 1.96°, respectively. The result of naive learning is enhanced by

Table 1: Summary of Experiment Scenes

Env. (#)	Scene (#)	RSS (dBm)	Total (#)	Density (p/m^3)	Space (m^2)	Distance (m)
Semi	A	−62.5	84,392	3,843.0	78.5	5
	B	−88.6	50,186	10,490.4	7,854.0	50
	C	−68.3	18,726	6,079.9	1,256.6	20
	D	−68.9	77,538	4,345.1	530.9	13
Full	E	−66.2	78,635	27,924.4	314.2	10
	F	−65.1	48,467	22,627.0	153.9	7
	G	−61.4	10,521	4,911.8	78.5	5
	H	−61.7	5,102	912.7	113.1	6
	I	−60.9	7,466	823.0	113.1	6
	J	−61.6	25,543	1,576.7	78.5	5
	K	−71.0	28,882	1,380.6	1256.6	20
	L	−77.7	52,634	935.9	1963.5	25
	M	−79.9	21,683	1,335.2	3217.0	32
	N	−68.5	21,729	848.1	254.5	9

NeRF^2 with 47.9%. On the other hand, the accuracy of turbo-learning is extremely approaching the 1.42° error that the baseline achieves. This result demonstrates that the quality of the synthetic dataset generated by NeRF^2 is as good as the true dataset. Clearly, the quantity of true training dataset required by turbo-learning is far less than the baseline, but the accuracy remains at a comparably high level. This feature is useful because collecting a training dataset is an important but cumbersome and painful task for today's deep learning. The power of NeRF^2 is in the significant reduction of the quantity of true training set and the corresponding workload.

We also test other mixture ratios (30% TS+70% SS, 50% TS+50% SS, 70% TS+30% SS, and 90% TS+10% SS) using the same ways. The results are shown in Fig. 7(b)-(e). As desired, the error of turbo-learning is reduced from 1.96° to 1.72°, 1.62°, 1.59°, and 1.50°. Evidently, the accuracy is increased with an increasing quantity of true datasets. This is understandable because the accuracies of NeRF^2 and AANN improve as the amount of true information increases. On the other hand, turbo-learning outperforms naive learning by 47.9% to 29.1%, 26.2%, 24.3%, and 18.2%. This demonstrates that more benefits can be gained when more percent of synthetic data is given. Even if only 10% synthetic data is fed, the median error can be reduced by 18.2% compared with naive learning. One may wonder why do not try the mixture of 0% TS plus 100% SS. It is impossible because the training of NeRF^2 must require a few numbers of true datasets. To achieve the trade-off between the accuracy and the quantity, it is advisable to take the mixture of 30% TS plus 70% SS in practice.

4.4 Large-scale Experiments

Regardless of NeRF or NeRF^2, both are scene-dependent because the radiance field is highly related to the scene layout. Whether the outperformance of turbo-learning can still be achieved in different scenes is unclear. Thus, we conduct large-scale experiments. We use the same antenna array to collect a huge dataset at 531,504 positions from 14 scenes (labeled A∼N). The settings are listed in Table 1. Fig. 9 shows eight of them (owing to the space limit). We first collect the data in a large-area semi-indoor environment with the purpose of quantifying the impact of the distance. In such an environment, the antenna array is deployed in four scenes labeled A, B, C, and D, which are large-area and semi-closed halls, as shown in (a)-(d) of Fig. 9. In these scenes, the distance varies from 5 to 50 m. The distance is the mean value between the scene center and the antenna array. We then collect the data in the full-indoor environment. We deploy the platform in 10 rooms (i.e., Scenes E-N). Scene E is a warehouse, Scene F is a lab room, and Scenes G-I are classrooms. Scenes J and N are offices, Scene K is the hallway, Scene L is a meeting room, and Scene M is a lift lobby, as shown in (e)-(j) of Fig. 9. The coverage of the scene ranges from 5 to 32 m. Particularly, the gateways are deployed behind the wall in Scenes L, M, and N. Majority of these data are collected in the scenes full of people passing by and various reflectors.

Similarly, we choose the 80% dataset for training and the 20% dataset for testing in each scene. Naive learning with the entire 80% true dataset is used for the baseline. Turbo-learning is conducted with 10% (out of the 80%) of the true training dataset plus 90% synthetic dataset. The AoA accuracy results are shown in Fig. 11. From the figure, we have the following two findings:

- Compared with naive learning (NL), the turbo-learning (TL) can offer 33%-70% improvement. The average is 49.5%. This shows that the performance enhancement by turbo-learning is a general phenomenon across scenes.
- Compared with the baseline (BL), the turbo-learning can hold −27.5% gap, where the minus sign denotes the "lower accuracy than". However, turbo-learning saves 90% workload for the dataset collection because only 10% training set is used.

Our experiments reveal two key factors influencing turbo-learning performance: (1) Quantity of the dataset. NeRF^2 has the ability to reduce the requirement for data collection in application-layer NN tasks. However, if adequate data is provided, the application-layer NNs can train the model effectively, thus reducing the benefits of NeRF^2. (2) Quality of the dataset. The performance of NeRF^2 can also be affected by environmental interference, such as passing by people or other signals. Despite this, our results demonstrate that turbo-learning still improves the performance of application-layer NNs by over 30%. In summary, the outperformance of turbo-learning is mainly derived from the physical model provided by NeRF^2. Naive learning models the "signature (feature)-based" relationship between the AoA and the spectrums, but NeRF^2 learns the physical rationale behind the relationship so it can provide more reasonable samples for the learning.

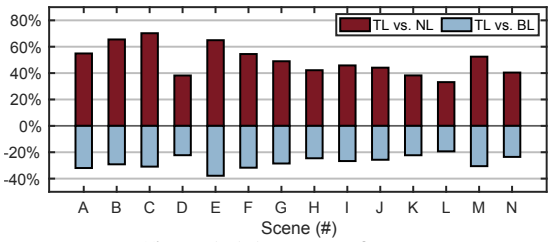


Fig. 11: AoA Accuracy vs. Scenes

5 FIELD STUDY: BLE LOCALIZATION

In this section, we discuss how NeRF^2 helps indoor localization in the scenario where no antenna array is available at a receiver. We conduct a large-scale experiment with 50 BLE gateways in an elderly nursing home. The project aims to track the potential spread of COVID-19 to protect elderlies from the infection better.

5.1 Experiment Setup

Fig. 12 shows the floor plan of the facility, which occupies 15,000 ft^2. A total of 50 BLE gateways (red circles) are deployed to collect the ID and RSSI of BLE beacons. Each gateway is $72\times7\times20$mm^3 in size, operates at 2.4 GHz and adopts an NRF52832 Bluetooth SoC [45] from Nordic Semiconductor. Redundant gateways are deployed to ensure that each location can be covered by at least 3 gateways. The BLE nodes are embedded into the visitor cards or elderlies' wristbands. They broadcast every 500 ms with 4 dBm transmitting power.

Ground Truth. The Velodyne VLP-16 LiDAR plus a 9-axis IMU are used to serve LIO-SAM (i.e., a publicly available SLAM algorithm [46]) for localization and map construction. The gateways and nodes are located by the LiDAR system as the ground truth. Taking 30 BLE nodes, we randomly walk into the house and totally create a dataset involving 6 K positions in the scene. Each dataset item is a 50-dimensional tuple, including the RSSI values detected by the 50 gateways, plus the position of the BLE node. The RSSI value is set to -100 dB by default if the gateway does not detect any signal from the node. 70% (4.2 K) and 30% (1.8 K) of the dataset are chosen from training and testing datasets.

5.2 RSSI Prediction

Data-driven approaches are emerging as promising solutions for BLE localization, such as KNN, SVM, and MLP. These methods require an accurate dataset for fingerprint matching or network training. Here, we apply turbo-learning for BLE localization, where the NeRF^2 is trained using the single-antenna RX model (see §2.3.1). The training process is more complicated than in previous cases in that we have 50 RXs here. The beacon of the same BLE node may be received by multiple gateways simultaneously. In this case, we must take ray tracing multiple times, in each of which the result is an aggregation of signals arriving at the corresponding RX

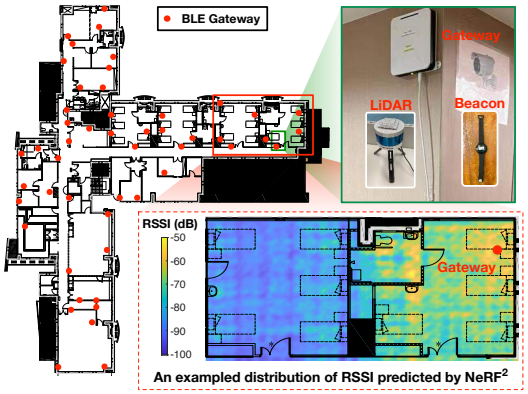


Fig. 12: The floor plan of the nursing home and deployment of BLE gateways.

from all possible directions (Eqn. 9). Similarly, given a position that a BLE node locates in the scene, we must take the ray tracing to predict the RSSI of the signal received at any gateway with the help of NeRF^2. Fig. 12 shows an example distribution of the predicted RSSI across the two rightmost rooms. It can be seen that the coverage of a gateway is not as good as that the manual claims (i.e., 10 m). The signal becomes very weak after walls. Thus, we deployed 3-4 gateways in each room to ensure full coverage. For comparison, we also adopt two other proposed prediction approaches, MRI [47] and CGAN [48]. MRI interpolates the RSSI values at the unsampled location using a basic radio propagation model. CGAN uses the conditional generative adversarial network to predict the RSSI values straightforwardly without regarding any physical model. The prediction error is defined as the difference between the predicted and the collected RSSI values at 1.8 K tested positions. The CDFs of the prediction error are shown in Fig. 13. As a result, the median of NeRF^2 is 2.6 dB (10$^{\text{th}}$ percentile: 0.5 dB; 90$^{\text{th}}$ percentile: 5.7 dB). By contrast, the median errors of CGAN and MRI are 4.5 dB and 6.2 dB, respectively. Evidently, NeRF^2 performs far better than the two others because it combines the advantages of deep learning (e.g., CGAN) and the physical model (e.g., MRI). The physical model provides prior knowledge about signal propagation, while deep learning uses statistical models to depict complicated RF interactions.

5.3 Localization Results

We use the well-trained NeRF^2 to generate a 20 K synthetic dataset at random locations and feed them to the following two localization algorithms.

■ **Turbo-KNN (T-KNN)**: We first evaluate the fingerprint-based localization approach, which assumes the RSSI values are highly related to a node's location. The K-nearest positions are chosen to compute the target node's location, where the RSSI values collected from these K positions (saved in a database) are most close to the RSSI value collected from the

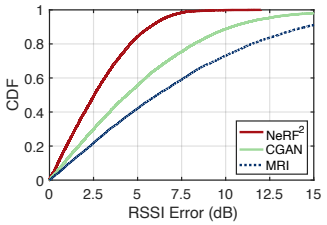


Fig. 13: RSSI Prediction

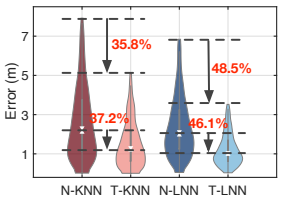


Fig. 14: Localization Result

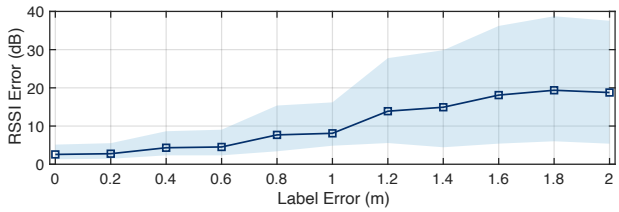


Fig. 15: RSSI vs. Label error

unknown position. The node is located at the weighted average of the K positions [49]. Fig. 14 shows the localization accuracy of naive-KNN (N-KNN) and T-KNN. The N-KNN only adopts the 4.2K true dataset only, whereas T-KNN uses the 20 K synthetic dataset. The median error of T-KNN is 1.41 m (10^{th} percentile: 0.27 m; 90^{th} percentile: 3.3 m), whereas that of N-KNN is 2.52 m (10^{th} percentile: 0.61 m; 90^{th} percentile: 5.38 m). Turbo-learning helps the KNN-based localization approach reduce the error by 44%.

■ **Turbo-LNN (T-LNN)**: We build another neural network to learn the mapping between an RSSI tuple and a position. We call this network *localization neural network* (LNN), which accepts the 50-dimensional RSSI tuple as input and outputs the position. The LNN consists of five-layer fully connected layers with the ReLU activation function. Similarly, the LNN is trained by using the 4.2 K true dataset and 20 K synthetic dataset, respectively. We call them naive-LNN (N-LNN) and T-LNN. Fig. 14 shows their results. The median error of T-LNN is 1.11 m (10^{th} percentile: 0.34 m, 90^{th} percentile: 3.46 m), whereas that of N-LNN is 2.26 m (10^{th} percentile: 0.75 m, 90^{th} percentile: 6.78 m). The error is reduced by turbo-learning by 50.8% (i.e., 1.2 m error). Particularly, T-LNN further reduces the 30 cm median error than T-KNN.

In summary, NeRF²-powered turbo-learning can effectively reduce the localization errors by ∼ 50% regardless of which data-driven approach is used. It also decreases the standard variance by ∼ 40% because the scale of training data is enlarged by 5× and a larger number of samples clearly benefit the convergence.

5.4 Impact of Label Errors

Label accuracy critically affects deep learning algorithm performance, so we investigate the impact of beacon location label errors on RSSI prediction accuracy. Fig. 15 shows the results, representing median prediction error and quartiles. We introduce uniformly distributed noise to the location label to simulate errors, whereby the label is reported with the circular error of radius r. The error r is increased from 0 m to 2 m in a step of 0.2 m, and the corresponding median RSSI prediction error degrades from 2.6 dB to 18.8 dB, with a near 7× increase. The standard deviation also increases from 1.9 dB to 9 dB. When the label error exceeds 1 m, the prediction accuracy of RSSI decreases severely. This is primarily because when the error surpasses 1 m, the label may be inaccurately situated in another room. On the other hand, when is label error is less than 0.6 m, the RSSI prediction error is below 4.5 dB, which is only 1.9 dB when compared to the absence of error. As such, we adopt the SLAM algorithm, which guarantees a label error of less than 0.2 m, as our preferred method of collecting ground truth data.

6 FIELD STUDY: 5G MIMO

In this section, we discuss how NeRF² helps massive MIMO channel estimation of the Frequency Domain Duplex (FDD) system for 5G, where the uplink and downlink transmissions operate at different frequencies. Therefore, the principle of reciprocity that two link channels are equal no longer holds [32]. To estimate the downlink CSI for beamforming, the client devices must receive extra symbols transmitted by the base station equipped with a massive antenna array and then send the estimated result back to the base station, leading to unsustainable overheads. To solve this problem, substantial research is devoted to predicting the downlink channel state by observing the uplink channel, based on the path-sharing assumption that both link channels are created by the same underlying physical environment and the identical paths being traveled [32, 50, 51]. For example, FNN [51] and FIRE [32] make use of a fully connected network and a VAE to transfer the estimated CSI from the uplink to the downlink, respectively.

This problem naturally falls into the domain of NeRF² because NeRF² represents the scene by using an RF radiance field. Given the position of the client device, NeRF² can exactly predict what kind of signal will be received at (or transmitted from) the base station. The key question is how we know the position of the client device. As studied previously [15, 52], the CSI is highly related to the physical environment, so a certain and unique mapping supposedly exists between the position of the client device and the CSI of its uplink signal, which is similar to fingerprint-based localization. Unlike previous works, which attempt to transfer the uplink CSI to the downlink CSI, we use the uplink CSI as a position indicator to train NeRF², which then predicts the downlink CSI directly. Formally, Eqn. 16 is rewritten as:

$$\mathrm{F}_\Theta : (I_{\text{uplink}}, \omega, P_x) \rightarrow (\delta_x, S_x) \tag{16}$$

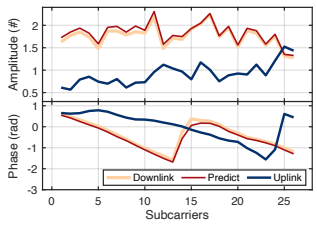


Fig. 16: Channel Amplitude & Phase

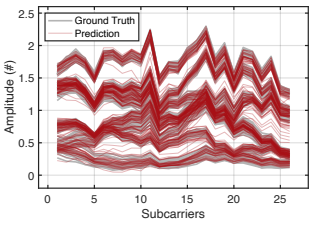


Fig. 17: Channel Amplitude in 2s

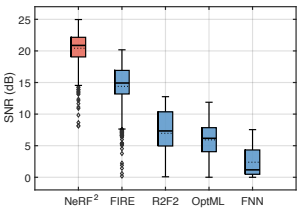


Fig. 18: Prediction SNR

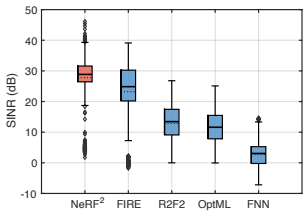


Fig. 19: MU-MIMO SINR

where I_{uplink} is the position indicator (i.e., uplink CSI). However, the following ray tracing works at the downlink frequencies, and the network is trained with the collected downlink CSI. Thus, our solution does not rely on the path-sharing assumption anymore.

6.1 Experiment Setup

We choose the publicly available Argos channel dataset [53] for our evaluation. Argos dataset is a real-world multi-user MIMO (MU-MIMO) dataset that contains 104 antennas at the base station and eight users, including mobile and static traces collection. The dataset contains two different working frequency versions, i.e., 2.4 GHz and 5 GHz. We need to train two separate NeRF^2 networks for them. The dataset is collected on ArgosV2 platform [54], which uses omnidirectional monopole antennas with the spacing of a half wavelength at 2.4 GHz (i.e., 63.5 mm). The system has up to 20 MHz bandwidth with 64 OFDM subcarriers. To estimate the CSI, the system sends 802.11 Long Training Symbols pilots at the beginning of each frame with 52 subcarriers. Similar to the previous work [32], we use the first half of the 52 subcarriers (i.e., 26 subcarriers) for the uplink channel, whereas the remained half is the downlink channel. Furthermore, the uplink and downlink channels are separated by guard bands. We aim to predict the downlink CSI from the uplink CSI without any feedback. The dataset is collected in a complex environment with numerous non-line-of-sight propagations. The dataset contains a total of 100 K items; 70% are used for the training, and 30% are used for the testing. We choose FIRE [32], R2F2 [50], OptML [51], and FNN [55] for comparison. For fairness, the experimental results of these four algorithms (including SNR and SINR) are taken from [32], which are also measured based on the same dataset.

6.2 Channel Prediction Accuracy

First, we evaluate the accuracy of downlink channel prediction using NeRF^2. We show a prediction example in Fig. 16. The blue uplink CSI, including the phase and amplitude at 26 subcarriers, is fed into NeRF^2, and the outputted downlink CSI is shown in yellow. The red ground truth is collected by the real antenna array. The two curves are almost overlapped, exhibiting an incredibly high-accuracy prediction. Fig. 17 plots the amplitude of all ground truth and predicted downlink CSI for a mobile client in 2 seconds. Each red curve shows one prediction result of 26 subcarriers at one time. NeRF^2 can still achieve high-accuracy prediction when the client is moving. To quantify the accuracy, we adopt the prediction SNR (proposed in [32]), which compares the predicted channel $\mathbf{H}$ and the ground truth channel $\mathbf{H}_{\text{gt}}$ as follows:

$$\text{SNR} = -10\log_{10}\left\|\mathbf{H} - \mathbf{H}_{\text{gt}}\right\|^2 + 10\log_{10}\left\|\mathbf{H}_{\text{gt}}\right\|^2 \tag{17}$$

A higher positive SNR is obtained when the predicted channel gets closer to the ground truth. Fig. 18 shows the SNR CDF of five prediction algorithms. Clearly, NeRF^2 achieves a far higher SNR compared with others. Specifically, NeRF^2 achieves a median SNR of 20.87 dB (10^{th} percentile: 17.19 dB, 90^{th} percentile: 23.14 dB). The median SNRs of FIRE, R2F2, OptML, and FNN are 14.9 dB, 7.3 dB, 16.1 dB, and 1.2 dB, respectively. NeRF^2 is 5.97 dB better than the VAE-based FIRE, 13.57 dB better than R2F2, 4.77 dB better than OptML, and 19.67 dB better than FNN. This demonstrates the high-accuracy prediction ability of NeRF^2 again.

6.3 MU-MIMO Performance

Finally, we evaluate the performance of NeRF^2 for MU-MIMO. In an MU-MIMO system, the base station transmits to multiple clients simultaneously through the separated beamforming. MU-MIMO requires a highly accurate channel estimation because a small error resulting from the CSI estimation for one client device may cause leaked interference to other client devices. Owing to the space limit, we encourage the reader to refer to [32, 56] for details about how the base station encodes the data to achieve the MU-MIMO. Here, we randomly select two clients to communicate with the base station equipped with 8 antennas, which constructs an 8×2 MU-MIMO system. The signal-to-interference-and-noise-ratio (SINR) is used to indicate the performance of an MU-MIMO system. Fig. 19 shows the results. NeRF^2 can achieve a median SINR of 29.22 dB. By contrast, the median error of FIRE, R2F2, and OptML are 24.90 dB, 13.33 dB, and 11.53 dB, respectively. The outperformance of NeRF^2 in the MU-MIMO is derived from the precise channel estimation.

7 RELATED WORK

Our work falls under broad channel measurements and wireless localization studies.

Optical Neural Radiance Field. NeRF attracts considerable attention in the field of computer vision with the development of related research. It shows great potential for 3D model and environment reconstruction [10, 57, 58], scene relighting and view synthesis [8, 11], and so on. To the best of our knowledge, NeRF^2 is the first to model the neural radiance fields on the basis of RF signals. Unlike optical NeRF, NeRF^2 considers the phase of RF signals, which requires a different physical tracing model. Moreover, owing to the limited size of the antenna array compared with the camera, NeRF^2 proposes two different training approaches. By precisely predicting the RF signal, NeRF^2 benefits many RF applications, such as wireless localization and MIMO channel prediction.

Channel Estimation. Channel estimation is a critical task in wireless systems, with past works employing training pilots [59], feedback [60], parametric or empirical models [50], and blind or opportunistic methods [61]. As the number of antennas skyrockets, the classic feedback-based estimation methods bring excessive overhead [50]. Recent works [32, 51, 55, 62] attempt to use deep learning to fill out the gap. NNCONFIG [62] proposed a scheme for mapping the trained neural networks to intelligent surfaces. State-of-the-art work, FIRE [32], proposes a generative model based on VAE to learn the downlink channel from the uplink feedback. However, they are purely data-driven machine learning models and highly rely on massive data. Unlike them, NeRF^2 incorporates a physical model (RF radiance field) into the learning process, enhancing interpretability and channel learning accuracy through prior wave transmission knowledge.

Wireless Localization. Wireless localization is a long-studied topic with extensive works [13, 49, 63]. Past works locate a device by building the transmission model between the position and various metrics of received RF signals: RSSI [49], phase [13, 64], CSI [15, 63], Time-of-Flight [65], and AoA [16, 34, 66, 67]. They are widely used in Wi-Fi [16, 41, 52], Bluetooth [66, 68], RFID [34, 69, 70], and LoRa [71]. However, classic geometric or empirical localization models suffer from problems caused by the environment (e.g., complex multipath effect, non-line of sight, etc. These problems are extensively studied in literature [30, 41]. Recent state-of-the-art approaches leverage deep learning models to enhance accuracy in complex indoor scenarios [34, 41, 72, 73] but require extensive training data, limiting real-world applications. In contrast, NeRF^2 embeds physical wave transmission knowledge into learning networks, improving model efficiency through turbo-learning and reducing raw training data while maintaining localization accuracy, thus offering a more practical solution.

8 CONCLUSION AND FUTURE WORK

We present NeRF^2, a novel deep learning architecture for wireless channel understanding. NeRF^2 first embeds the entire electromagnetic wave transmission physical model into the channel learning model. Our extensive experiments show that NeRF^2 can boost the performance of many deep-learning based application-layer tasks, including indoor localization and massive MIMO communication. Future work will focus on addressing the following challenges and improvements:

Generalizability. Transferring NeRF^2 across different scenes remains a substantial challenge. Potential solutions include pre-training on large datasets from multiple scenes for improved generality, employing incremental learning to adapt to surrounding noise or dynamic environments, and utilizing transfer learning to maximize model reusability across various scenes.

Time consumption. NeRF^2 needs several hours to train the model. However, recent substantial research has been devoted to reducing the time by compression methods. For example, Instant-NGP [9] can train an optical NeRF in 5 seconds. Consequently, future work will explore optimizing training time consumption through these methods.

ACKNOWLEDGMENTS

This study is supported by NSFC Key Program (No. 61932017), NSFC Excellent Young Scientists Fund (Hong Kong and Macau) (No. 62022003), NSFC General Program (No. 61972331), and UGC/GRF (No. 15204820, 15215421). We thank all the anonymous reviewers and the shepherd, for their valuable comments and helpful suggestions.

REFERENCES

[1] Z. Yun and M. F. Iskander, "Ray tracing for radio propagation modeling: Principles and applications," *IEEE Access*, vol. 3, pp. 1089–1100, 2015.

[2] R. G. Kouyoumjian and P. H. Pathak, "A uniform geometrical theory of diffraction for an edge in a perfectly conducting surface," *Proceedings of the IEEE*, vol. 62, no. 11, pp. 1448–1461, 1974.

[3] "Electromagnetic Simulation Software," https://www.remcom.com/.

[4] E. Egea-Lopez, J. M. Molina-Garcia-Pardo, M. Lienard, and P. Degauque, "Opal: An open source ray-tracing propagation simulator for electromagnetic characterization," *Plos one*, vol. 16, no. 11, p. e0260060, 2021.

[5] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," in *ECCV 2020*. Springer, 2020, pp. 405–421.

[6] M. Tancik, P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. Barron, and R. Ng, "Fourier features let networks learn high frequency functions in low dimensional domains," *Advances in Neural Information Processing Systems*, vol. 33, pp. 7537–7547, 2020.

[7] J. T. Barron, B. Mildenhall, M. Tancik, P. Hedman, R. Martin-Brualla, and P. P. Srinivasan, "Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields," *ICCV*, 2021.

[8] P. P. Srinivasan, B. Deng, X. Zhang, M. Tancik, B. Mildenhall, and J. T. Barron, "Nerv: Neural reflectance and visibility fields for relighting and view synthesis," in *Proc. of IEEE/CVF CVPR*, 2021, pp. 7495–7504.

[9] T. Müller, A. Evans, C. Schied, and A. Keller, "Instant neural graphics primitives with a multiresolution hash encoding," *ACM Transactions on Graphics (ToG)*, vol. 41, no. 4, pp. 1–15, 2022.

[10] M. Tancik, V. Casser, X. Yan, S. Pradhan, B. Mildenhall, P. P. Srinivasan, J. T. Barron, and H. Kretzschmar, "Block-nerf: Scalable large scene neural view synthesis," in *Proc. of IEEE/CVF CVPR*, 2022, pp. 8248–8258.

[11] R. Martin-Brualla, N. Radwan, M. S. Sajjadi, J. T. Barron, A. Dosovitskiy, and D. Duckworth, "Nerf in the wild: Neural radiance fields for unconstrained photo collections," in *Proc. of IEEE/CVF CVPR*, 2021, pp. 7210–7219.

[12] "NeRF Demonstration," https://www.matthewtancik.com/nerf.

[13] L. Yang, Y. Chen, X.-Y. Li, C. Xiao, M. Li, and Y. Liu, "Tagoram: Real-time tracking of mobile rfid tags to high precision using cots devices," in *Proc. of ACM MobiCom*, 2014, pp. 237–248.

[14] Y. Ma, N. Selby, and F. Adib, "Minding the billions: Ultra-wideband localization for deployed rfid tags," in *Proceedings of the 23rd annual international conference on mobile computing and networking*, 2017, pp. 248–260.

[15] Y. Xie, J. Xiong, M. Li, and K. Jamieson, "md-track: Leveraging multi-dimensionality for passive indoor wi-fi tracking," in *Proc. of ACM MobiCom*, 2019, pp. 1–16.

[16] Y. Xie, Y. Zhang, J. C. Liando, and M. Li, "Swan: Stitched wi-fi antennas," in *Proc. of ACM MobiCom*, 2018.

[17] F. Adib, Z. Kabelac, D. Katabi, and R. C. Miller, "3d tracking via body radio reflections," in *Proc. of USENIX NSDI*, vol. 14, 2013.

[18] F. Adib, Z. Kabelac, and D. Katabi, "Multi-person motion tracking via rf body reflections," in *Proc. of USENIX NSDI*, 2015.

[19] M. Zhao, Y. Tian, H. Zhao, M. A. Alsheikh, T. Li, R. Hristov, Z. Kabelac, D. Katabi, and A. Torralba, "Rf-based 3d skeletons," in *Proc. of ACM SIGCOMM*, 2018, pp. 267–281.

[20] M. Zhao, T. Li, M. Abu Alsheikh, Y. Tian, H. Zhao, A. Torralba, and D. Katabi, "Through-wall human pose estimation using radio signals," in *Proc. of IEEE/CVF CVPR*, 2018, pp. 7356–7365.

[21] Y. Ma and E. C. Kan, "Accurate indoor ranging by broadband harmonic generation in passive nltl backscatter tags," *IEEE Transactions on Microwave Theory and Techniques*, vol. 62, no. 5, pp. 1249–1261, 2014.

[22] X. Hui and E. C. Kan, "Radio ranging with ultrahigh resolution using a harmonic radio-frequency identification system," *Nature Electronics*, vol. 2, no. 3, p. 125, 2019.

[23] A. Haniz, G. K. Tran, K. Saito, K. Sakaguchi, J.-i. Takada, D. Hayashi, T. Yamaguchi, and S. Arata, "A novel phase-difference fingerprinting technique for localization of unknown emitters," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 9, pp. 8445–8457, 2017.

[24] M. Youssef and A. Agrawala, "The horus wlan location determination system," in *Proc. of ACM MobiSys*, 2005, pp. 205–218.

[25] S. Sen, B. Radunovic, R. R. Choudhury, and T. Minka, "You are facing the mona lisa: Spot localization using phy layer information," in *Proc. of ACM MobiSys*, 2012, pp. 183–196.

[26] Z. Yang, C. Wu, and Y. Liu, "Locating in fingerprint space: wireless indoor localization with little human intervention," in *Proc. of ACM MobiCom*, 2012, pp. 269–280.

[27] H. Liu, Y. Gan, J. Yang, S. Sidhom, Y. Wang, Y. Chen, and F. Ye, "Push the limit of wifi based localization for smartphones," in *Proc. of ACM MobiCom*, 2012, pp. 305–316.

[28] H. Wang, S. Sen, A. Elgohary, M. Farid, M. Youssef, and R. R. Choudhury, "No need to war-drive: Unsupervised indoor localization," in *Proc. of ACM MobiSys*, 2012, pp. 197–210.

[29] L. Ni, Y. Liu, Y. Lau, and A. Patil, "Landmarc: Indoor location sensing using active rfid," *Wireless networks*, 2004.

[30] J. Wang and D. Katabi, "Dude, where's my card?: Rfid positioning that works with multipath and non-line of sight," in *Proc. of ACM SIGCOMM*, 2013.

[31] S. J. Pan, V. W. Zheng, Q. Yang, and D. H. Hu, "Transfer learning for wifi-based indoor localization," in *Proc. of ACM AAAI workshop*, vol. 6, 2008.

[32] Z. Liu, G. Singh, C. Xu, and D. Vasisht, "Fire: enabling reciprocity for fdd mimo systems," in *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, 2021, pp. 628–641.

[33] P. Stoica, R. L. Moses *et al.*, *Spectral analysis of signals*. Pearson Prentice Hall Upper Saddle River, NJ, 2005, vol. 452.

[34] Z. An, Q. Lin, P. Li, and L. Yang, "General-purpose deep tracking platform across protocols for the internet of things," in *Proc. of ACM MobiSys*, 2020, pp. 94–106.

[35] "OptiTrack," https://optitrack.com/.

[36] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[37] R. Miesen, A. Parr, J. Schleu, and M. Vossiek, "360 carrier phase measurement for uhf rfid local positioning," in *2013 IEEE International Conference on RFID-Technologies and Applications (RFID-TA)*. IEEE, 2013, pp. 1–6.

[38] "RayTracing Toolbox," https://www.mathworks.com/help/antenna/ref/rfprop.raytracing.html.

[39] U. Ha, J. Leng, A. Khaddaj, and F. Adib, "Food and liquid sensing in practical environments using rfids," in *Proc. of USENIX NSDI*, 2020.

[40] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.

[41] R. Ayyalasomayajula, A. Arun, C. Wu, S. Sharma, A. R. Sethi, D. Vasisht, and D. Bharadia, "Deep learning based wireless localization for indoor navigation," in *Proc. of ACM MobiCom*, 2020, pp. 1–14.

[42] P. Babakhani, T. Merk, M. Mahlig, I. Sarris, D. Kalogiros, and P. Karlsson, "Bluetooth direction finding using recurrent neural network," in *Proc. of IEEE IPIN*. IEEE, 2021, pp. 1–7.

[43] M. Comiter and H. Kung, "Localization convolutional neural networks using angle of arrival images," in *Proc. of IEEE GLOBECOM*. IEEE, 2018, pp. 1–7.

[44] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. of the IEEE/CVF CVPR*, 2016, pp. 770–778.

[45] "NRF52832," https://www.nordicsemi.com/products/nrf52832.
[46] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and R. Daniela, "Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping," in *Proc. of IEEE/RSJ IROS*. IEEE, 2020, pp. 5135–5142.
[47] H. Shin, Y. Chon, Y. Kim, and H. Cha, "Mri: Model-based radio interpolation for indoor war-walking," *IEEE Transactions on Mobile Computing*, vol. 14, no. 6, pp. 1231–1244, 2014.
[48] F. Parralejo, F. J. Aranda, J. A. Paredes, F. J. Álvarez, and J. Morera, "Comparative study of different ble fingerprint reconstruction techniques," in *Proc. of IEEE IPIN*. IEEE, 2021, pp. 1–8.
[49] L. M. Ni, Y. Liu, Y. C. Lau, and A. P. Patil, "Landmarc: Indoor location sensing using active rfid," in *Proc. of IEEE PerCom*. IEEE, 2003, pp. 407–415.
[50] D. Vasisht, S. Kumar, H. Rahul, and D. Katabi, "Eliminating channel feedback in next-generation cellular networks," in *Proc of ACM SIGCOMM*, 2016, pp. 398–411.
[51] A. Bakshi, Y. Mao, K. Srinivasan, and S. Parthasarathy, "Fast and efficient cross band channel prediction using machine learning," in *Proc. of ACM MobiCom*, 2019, pp. 1–16.
[52] M. Kotaru, K. Joshi, D. Bharadia, and S. Katti, "Spotfi: Decimeter level localization using wifi," in *Proc. of ACM SIGCOMM*, 2015, pp. 269–282.
[53] C. Shepard, J. Ding, R. E. Guerra, and L. Zhong, "Understanding real many-antenna mu-mimo channels," in *2016 50th Asilomar Conference on Signals, Systems and Computers*. IEEE, 2016, pp. 461–467.
[54] C. Shepard, H. Yu, N. Anand, E. Li, T. Marzetta, R. Yang, and L. Zhong, "Argos: Practical many-antenna base stations," in *Proc. of ACM MobiCom*. ACM, 2012, pp. 53–64.
[55] C. Huang, G. C. Alexandropoulos, A. Zappone, C. Yuen, and M. Debbah, "Deep learning for ul/dl channel calibration in generic massive mimo systems," in *Proc. of IEEE ICC*. IEEE, 2019, pp. 1–6.
[56] Q. H. Spencer, A. L. Swindlehurst, and M. Haardt, "Zero-forcing methods for downlink spatial multiplexing in multiuser mimo channels," *IEEE transactions on signal processing*, vol. 52, no. 2, pp. 461–471, 2004.
[57] L. Liu, J. Gu, K. Zaw Lin, T.-S. Chua, and C. Theobalt, "Neural sparse voxel fields," *Advances in Neural Information Processing Systems*, vol. 33, pp. 15 651–15 663, 2020.
[58] K. Rematas, A. Liu, P. P. Srinivasan, J. T. Barron, A. Tagliasacchi, T. Funkhouser, and V. Ferrari, "Urban radiance fields," in *Proc. of IEEE/CVF CVPR*, 2022, pp. 12 932–12 942.
[59] T. L. Marzetta and B. M. Hochwald, "Fast transfer of channel state information in wireless systems," *IEEE Transactions on Signal Processing*, vol. 54, no. 4, pp. 1268–1278, 2006.
[60] X. Fan, L. Shangguan, R. Howard, Y. Zhang, Y. Peng, J. Xiong, Y. Ma, and X.-Y. Li, "Towards flexible wireless charging for medical implants using distributed antenna system," in *Proc of ACM MobiCom*, 2020, pp. 1–15.
[61] Y. Ma, Z. Luo, C. Steiger, G. Traverso, and F. Adib, "Enabling deep-tissue networking for miniature medical devices," in *Proc. of ACM SIGCOMM*, 2018, pp. 417–431.
[62] C. Liaskos, S. Nie, A. Tsioliaridou, A. Pitsillides, S. Ioannidis, and I. Akyildiz, "End-to-end wireless path deployment with intelligent surfaces using interpretable neural networks," *IEEE Transactions on Communications*, vol. 68, no. 11, pp. 6792–6806, 2020.
[63] Z. Yang, Z. Zhou, and Y. Liu, "From rssi to csi: Indoor localization via channel response," *ACM Computing Surveys (CSUR)*, vol. 46, no. 2, pp. 1–32, 2013.
[64] Y. Ma, N. Selby, and F. Adib, "Drone relays for battery-free networks," in *Proc. of ACM SIGCOMM*, 2017, pp. 335–347.
[65] A. T. Mariakakis, S. Sen, J. Lee, and K.-H. Kim, "Sail: Single access point-based indoor localization," in *Proc. of ACM MobiSys*, 2014, pp. 315–328.
[66] R. Ayyalasomayajula, D. Vasisht, and D. Bharadia, "Bloc: Csi-based accurate localization for ble tags," in *Proc. of ACM CoNEXT*, 2018, pp. 126–138.
[67] J. Xiong and K. Jamieson, "Arraytrack: A fine-grained indoor location system," in *Proc. of USENIX NSDI*, 2013, pp. 71–84.
[68] M. Cominelli, P. Patras, and F. Gringoli, "Dead on arrival: An empirical study of the bluetooth 5.1 positioning system," in *Proc. of WiNTECH*, 2019, pp. 13–20.
[69] J. Wang, D. Vasisht, and D. Katabi, "Rf-idraw: Virtual touch screen in the air using rf signals," *Proc. of ACM SIGCOMM*, vol. 44, no. 4, pp. 235–246, 2014.
[70] J. Wang, J. Xiong, H. Jiang, X. Chen, and D. Fang, "D-watch: Embracing "bad" multipaths for device-free localization with cots rfid devices," *IEEE/ACM Transactions on Networking*, vol. 25, no. 6, pp. 3559–3572, 2017.
[71] N. BniLam, D. Joosens, M. Aernouts, J. Steckel, and M. Weyn, "Loray: Aoa estimation system for long range communication networks," *IEEE Transactions on Wireless Communications*, vol. 20, no. 3, pp. 2005–2018, 2020.
[72] Y. Zheng, Y. Zhang, K. Qian, G. Zhang, Y. Liu, C. Wu, and Z. Yang, "Zero-effort cross-domain gesture recognition with wi-fi," in *Proc. of ACM Mobisys*, 2019, pp. 313–325.
[73] U. Raza, A. Khan, R. Kou, T. Farnham, T. Premalal, A. Stanoev, and W. Thompson, "Dataset: Indoor localization with narrow-band, ultra-wideband, and motion capture systems," in *Proceedings of the 2nd Workshop on Data Acquisition to Analysis*, 2019, pp. 34–36.