INFERRING INDOOR FLOORPLANS FROM WIRELESS SIGNALS

BY

CHAITANYA AMBALLA

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois Urbana-Champaign, 2025

Urbana, Illinois

Adviser:

Professor Romit Roy Choudhury

ABSTRACT

Neural Radiance Fields (NeRFs) have been remarkably successful at synthesizing novel views of 3D scenes by optimizing a volumetric scene function. This scene function models how an optical ray accumulates colors on its path and eventually delivers this color to the camera pixel it impinges upon. Radio frequency (RF) or audio signals can also be viewed as a vehicle for delivering information about the environment to a sensor. However, unlike camera pixels, an RF/audio sensor receives a mixture of signals that contains many environmental reflections. Is it still possible to infer the environment using such mixed signals? We show that with redesign, the core NeRF framework has the potential to solve this inverse problem. We focus on a specific application of inferring the indoor floorplan of a home from WiFi measurements made at multiple locations inside the home. Our inferred floorplans look promising, and benefit downstream signal prediction applications. Our work also uncovers a number of problems for continued research.

To my parents and brother, for their love and support.

ACKNOWLEDGMENTS

I would thank my advisor, Prof. Romit Roy Choudhury, for his constant guidance and support throughout my research. I'm especially grateful for his belief in my abilities and for the constructive feedback that consistently pushed me to improve. Working with Romit taught me the value of intuition and clarity when approaching complex problems. He often encouraged me to step back and look at a problem at a higher level, which helped me to guage the problem better and not get lost in the low-level intricacies. Romit also gave me the space and trust to make difficult decisions — I remember a point in my research where I had to take a complete detour and start over. His support during that time was reassuring, and it helped us arrive at meaningful results.

Beyond research, Romit has had a lasting impact on how I think and work. His teaching style, which emphasizes getting the fundamentals right and building ideas step by step, helped me understand the value of a strong foundation. Outside of academics, his encouragement towards physical fitness made a real difference in my daily routine, helping me become more proactive and balanced in both work and life. I had originally planned to pursue only a Master's thesis, but Romit's mentorship and encouragement gave me the confidence to continue into the PhD program. I'm truly thankful for that opportunity.

I am grateful to Debottam, Sattwik, and Alan for the countless brainstorming sessions and random discussions. Their camaraderie has been a great source of support, and I've gained valuable insights from their perspective. A special thanks to Yu Lin Wei for being a second pillar of support, guiding me through crucial times and helping me navigate towards right research directions. I would also like to thank my group mates, Sahil and Rajalaxmi, for their assistance throughout this journey, as their insights in our group discussions have been immensely helpful. I am also thankful to

Akash and Debottam for being the best roommates I could have asked for. Grateful for Hamid at Jeruselum restaurant for his delicious food over the past two years.

To my friends Jeevan and Bhanu, I am especially thankful for making me feel at home. Our random trips across the country during every break have been a source of joy and a perfect escape from academia. To Venkatesh, Yuthsavi, and Sreeja, thank you for being there for me over the years, through long phone calls and for always being a source of strength. Sravanthi, I deeply appreciate our candid conversations about life and the shared struggles of PhD journey.

Finally, I am grateful to my family. Their constant support has been the anchor throughout this journey, giving me strength during both the highs and lows. To my brother, thank you for your steady presence and the encouragement along the way. I would like to express my deepest gratitude to my parents. I'm truly thankful for their love, sacrifices, and unwavering support over the years.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Indoor floorplans play a crucial role in a wide range of applications, including digital twins, context-aware systems like virtual assistants, indoor navigation, and augmented reality. For instance, virtual assistants can provide spatially-aware services, such as guiding users to specific rooms or controlling devices based on their physical location within a building. Estimating indoor floorplans has been extensively studied, with many existing solutions relying on visual sensors like cameras [1, 2, 3, 4, 5, 6], or LIDARs [7, 8]. These technologies typically require a robot or user to traverse the space, collecting measurements from the environment to build a map. While these methods provide accurate results, they come with significant limitations. Visual sensors can inadvertently capture user-sensitive information, raising privacy concerns. Additionally, capturing a comprehensive floorplan using visual data often requires exhaustive measurements, as every wall and corner of the indoor space must be viewed.

This thesis presents a novel approach to indoor floorplan estimation that mitigates both privacy and measurement efficiency challenges. More specifically, we are interested in inferring the floorplan of an indoor environment by measuring wireless signals from a mobile sensor. Using wireless sensing modalities such as RF or audio signals offers advantages in both the aforementioned challenges. Rich details of the user's home are not captured with RF/audio signal measurements (or at least not at the extreme resolution of visual sensors). Moreover, radio/audio waves propagate through walls, allowing MapFill to operate with fewer measurements – the user does not need to collect measurements from every room and corner. One could envision this as a user walking around with a phone in a home; can the phone record ambient WiFi signals (or audio music played from loudspeakers) and learn the home's floorplan?

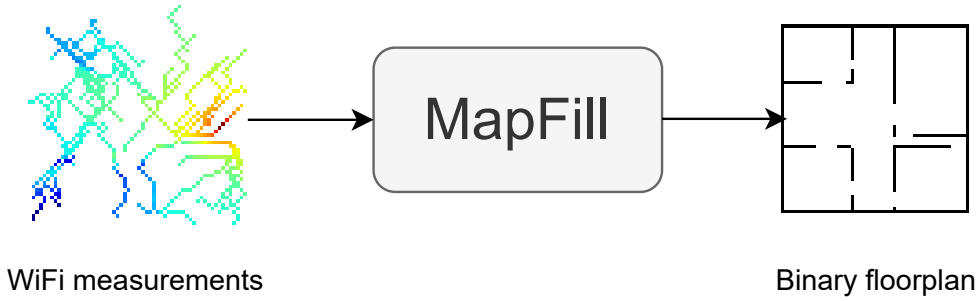

Figure 1.1: Overview of the problem

Figure 1.1 illustrates our "MapFill" ing problem where we observe the signal power at a series of locations in the environment as our input shown as a heat map and the output is the floorplan of the environment. This seems plausible, as the signal that arrives at the phone is a function of the environment's layout, the phone's location, and the signal propagation model. Signals passing through walls experience attenuation, and reflected signals arrive at the receiver with reduced power compared to those traveling via a direct, unobstructed path. As a result, the received signal at any location is effectively a summation of multiple components—some traveling directly, others passing through walls, and yet others arriving after one or more reflections. In essence, the spatial layout of the indoor environment shapes the observed signal. Figure 1.2 illustrates the signal propagation process in an indoor floorplan with one transmitter and one receiver. To achieve this, we provide two solutions, one that invloves learning the underlying signal propagation model implicitly from data and the other where the signal power is explicitly modeled as a function of the floorplan using Neural Radiance Fields (NeRFs) [9].

We briefly outline the two approaches in this Chapter. In the following chapters, we provide an in-depth discussion of these two approaches, with particular focus on the NeRF-based solution, which we consider the most innovative contribution of this thesis.

1.1 PhyMap

Floorplans inherently contain structural elements such as walls, doors, and furniture. These structures directly influence wireless signals, as the received

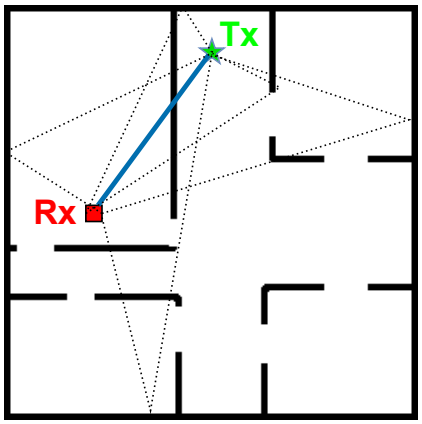

Figure 1.2: Signal propagation in a floorplan with one transmitter and one receiver. Direct path is shown in blue line, while the reflected paths are with black dotted lines.

signals undergo interactions with the environment during propagation. Consequently, estimating floorplans from wireless signals constitutes an inverse problem, where the forward process describes signal propagation by modeling signal power as a function of the environment.

In this work, we explore a data-driven approach to establish a mapping between measured signal power and the corresponding floorplan. Our proposed model, `PhyMap`, first learns a compact representation of the environment and subsequently relates input signal measurements to these learned representations. Given that limited measurements can lead to multiple plausible floorplans generating the same signal power, our mapping function is designed to produce a distribution over potential floorplans, thereby capturing this ambiguity.

The key challenge in this approach is make the mapping function learn the signal propagation model. While data-driven models excel at capturing structures in floorplans, directly learning a mapping from the signal power to the floorplan seems difficult – many functions can generate the same signal power given the limited number of measurements. To overcome this limitation, `PhyMap` incorporates a learnable physics-based model, `PhyModel`, which approximates the forward signal propagation process. Once learned, `PhyModel` facilitates physically-informed guidance to the inverse mapping function, ensuring that the generated floorplan is consistent with the observed signal power. Moreover, `PhyModel` is fully differentiable, enabling its integration into a test-time optimization framework and downstream tasks,

such as signal prediction.

Our physics-based model `PhyModel` requires the transmitter's location, receiver locations, and the environmental floorplan as inputs to predict the signal power at the receiver locations. However, since no prior knowledge about the transmitter's location is assumed, we design `PhyMap` to jointly estimate the transmitter location as part of the floorplan estimation process. Further details regarding `PhyMap` and `PhyModel` are discussed in Chapter 4. However, our experiments reveal that relying solely on the physics model `PhyModel` is insufficient for accurately solving the inverse problem. Specifically, certain floorplans may better fit the observed signal power than the actual floorplan, indicating that `PhyModel` does not perfectly capture the true underlying ray tracing dynamics. Consequently, while `PhyModel` demonstrates potential for modeling signal propagation with good performanace, it alone cannot reliably guide the inverse mapping function towards learning the true floorplan. Thus, explicit modeling of signal propagation is essential to solve the inverse problem, laying the foundation for our NeRF-based approach.

1.2 `NeRFMap`

Our second approach to "MapFill" problem builds on the idea of Neural Radiance Fields (NeRFs), and using them to implicitly learn the floorplan from wireless signals. We view our NeRF-based solution as the main contribution of this thesis and spend most of the thesis discussing this approach.

NeRFs [9, 10, 11, 12] have delivered impressive results in solving inverse problems, resulting in 3D scene rendering. While NeRFs have mostly used pictures (from cameras or LIDARs) to infer a 3D scene, we ask if the core ideas can generalize to the case of wireless signals as well (such as RF or audio). Generalizing optical NeRFs to wireless sensing seems plausible since wireless measurements are just observations from inside the scene, while NeRF images are typically taken from the outside. A growing body of work [13, 14, 15, 16, 17, 18] is investigating this connection between NeRFs and wireless. While none have predicted floorplans, NeRF2 [15] and NeWRF [16] have adopted NeRFs to predict WiFi signals at different locations inside an indoor space. However, we find that such predictions can be successful without necessarily solving the floorplan inference problem. In other words, it is

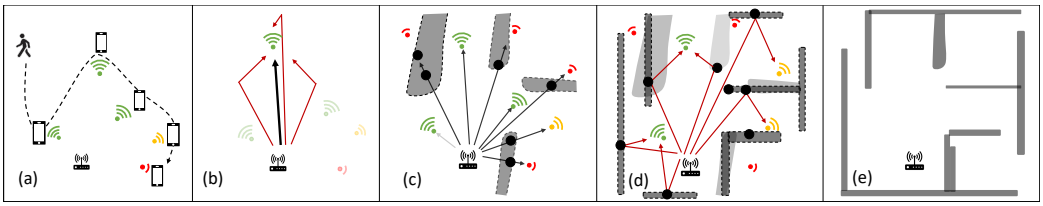

Figure 1.3: Overview: (a) User's device measures wireless signal power. (b) Transmitted signals arrive at the device along a line of sight (LoS) path and after reflections from surrounding walls. (c) Our LoS model estimates crude walls. (d) Our first-order reflection model refines the inner and outer walls. (e) Together, the floorplan is estimated which can then be used to predict signals at new locations.

possible to learn an implicit representation of the scene that does not align with signal propagation models. Our goal is to design an objective function that models indoor signal propagation and, in turn, trains the NeRF to learn the environment's floorplan. Once learnt, the floorplan can be utilized for various downstream tasks, such as signal prediction or basic ray tracing.

In our model, each NeRF voxel is defined by its opacity $\delta \in [0, 1]$ and orientation $\omega \in [-\pi, \pi]$. When trained perfectly, free-space air voxels should be transparent ($\delta = 0$), wall voxels should be opaque ($\delta = 1$), and each opaque voxel's orientation should match its wall's orientation. As measurements, we use the received *signal power*, easily available from any receiver's hardware. Thus, the input to our `NeRFMap` model is the transmitter (Tx) location, a sequence of known receiver (Rx) locations, and the signal power measured at each Rx location (Figure 1.3(a)). The output of `NeRFMap` is an (implicitly learnt) floorplan of the indoor environment. We expect to visualize the floorplan by plotting the learnt voxel opacities.

The key challenge in learning floorplans is in modeling the correct reflections since these reflections help reveal where the walls are. However, without knowledge of the walls, the reflections are not easy to model, leading to a type of chicken and egg problem. Additionally, the number of wireless measurements are relatively sparse since we envision users walking around with their phones only for a few minutes. Finally, measured signals will have "blind spots", meaning that rays that bounced off certain regions of the walls may not have arrived at any of the Rx locations. This leaves gaps or holes in the floorplan and NeRF's interpolation through these gaps will produce error or blur.

NeRFMap approaches this problem by modeling the received signal power as a sum of the line of sight (LoS) power and the power from first-order reflections (Figure 1.3(b)). The LoS model is inherited from classical NeRFs. The main departure from past work is in modeling the reflections. Since opaque voxels are unknown during training, the reflection surfaces are not known; hence, the reflection power at the Rx is modeled as an aggregate over all plausible reflections. Fortunately, for a given $\langle Tx, Rx \rangle$ pair, the voxels that can cause reflections lie on a geometric manifold (under reasonable assumptions), reducing the plausible set. The reflections are aggregated over this set to finally model the total (LoS + reflection) power.

NeRFMap trains to minimize the loss between the total and measured power across all Rx locations, and in the process, learns the voxel's opacities that best explain the measured dataset. To cope with sparse measurements, regularizations are added to enforce smoothness among local voxels; a penalty is imposed to prevent learning multiple reflections from one manifold. Lastly, NeRFMap freezes the LoS model once it converges (Figure 1.3(c)), and uses this intermediate state to partly supervise the reflection model (Figure 1.3(d)).

To evaluate NeRFMap, we train on 2.4 GHz WiFi measurements from NVIDIA's Sionna simulator [19], with floorplans from the Zillow's Indoor Dataset (ZIND) [20]. Results show consistent improvement over baselines in terms of the estimated floorplan's IoU and F1 score. Qualitative results show visually legible floorplans without any post-processing. As a derivative of the floorplan, NeRFMap can compute the received signal power for new $\langle Tx, Rx \rangle$ locations (outperforming existing baselines), and also shows basic ray tracing to explain this power, offering interpretability to its predictions.

The remainder of the thesis is organized as follows: In Chapter 2, we review the necessary background. In Chapter 3, we discuss the related work. In Chapter 4, we discuss the PhyMap and PhyModel framework, followed by the results, challenges and discussion. In Chapter 5, we present the NeRF-based solution NeRFMap. In chapter 6, we show experimental results of NeRFMap with baselines, few downstream applications, and limitations. Finally, we conclude the thesis in Chapter 7.

CHAPTER 2

BACKGROUND

2.1 Channel Impulse Response

When a wireless signal propagates, it is typically influenced by multipath effects such as reflections and scattering, as well as attenuation caused by the surrounding environment—collectively referred to as the channel. The overall impact of these phenomena on the signal is characterized by a linear model known as the Channel Impulse Response (CIR) [21].

Mathematically, the CIR is expressed as a sum of scaled and delayed impulses as shown in Equation (2.1).

$$h(t) = \sum_{i=1}^{N} \alpha_i e^{j\phi_i} \delta(t - \tau_i), \tag{2.1}$$

where N is the number of multipath components, α_i denotes the amplitude (attenuation factor) of the i-th path, ϕ_i represents the phase shift of the i-th path, and τ_i is the delay of the i-th path.

For an input signal $x(t)$ transmitted through the channel $h(t)$, the output signal measured at *Rx*, $y(t)$ is obtained by the convolution:

$$y(t) = x(t) * h(t) + n(t), \tag{2.2}$$

where $n(t)$ represents additive noise. For a simple two-path channel with a line of sight (LoS) path and one reflected path, the CIR $h(t)$ is given as

$$h(t) = \alpha_1 \delta(t) + \alpha_2 e^{j\phi_2} \delta(t - \tau_2), \tag{2.3}$$

The received signal $y(t)$ would then be:

$$y(t) = x(t) * \left[\alpha_1 \delta(t) + \alpha_2 e^{j\phi_2} \delta(t - \tau_2)\right] + n(t) \tag{2.4}$$

In a complex indoor environment, the number of multipath components N increases significantly due to multiple reflections, diffractions, and scattering caused by walls, furniture, and other obstacles. Each path differs in amplitude, delay, and phase, resulting in a CIR that encodes information about the spatial geometry of the environment.

To analyze the received signal strength at a particular location, the total received power is typically considered, computed as the squared magnitude of the CIR:

$$P_{\text{recv}} = |h(t)|^2 \tag{2.5}$$

The relative signal strength indicator (RSSI) is a common metric used to quantify this power, which is often expressed in decibels (dBm). The RSSI can be calculated as:

$$\text{RSSI} = 10 \log_{10} \left(P_{\text{recv}}\right), \tag{2.6}$$

These scalar power values P_{recv} and RSSI, derived from the CIR and used interchangeably, serve as the input measurements in our work for inferring the indoor floorplan.

2.2 Neural Radiance Fields

Neural Radiance Fields (NeRF)[9] is a method for synthesizing novel views of complex scenes by learning a continuous volumetric scene function from a set of posed images. The scene function takes a 3D spatial coordinate and a viewing direction as input and outputs the radiance and opacity at that point. NeRFs is trained by optimizing the weights of a neural network to minimize the error between the rendered images and the ground truth views. Once trained, the NeRF can be used to render novel viewpoints of the scene by querying the scene function at different 3D coordinates and viewing directions.

More formally, the color observed along a camera ray $r(t) = o + td$ is computed using the volume rendering equation:

$$C(r) = \int_{t_n}^{t_f} T(t)\sigma(r(t))c(r(t), d)dt, \tag{2.7}$$

where σ is the volume density, c is the emitted color, and $T(t)$ is the accumulated transmittance along the ray.

To make the integral tractable, NeRF employs a quadrature approximation from [22], allowing the integral to be approximated as:

$$C(r) = \sum_{i=1}^{N} T_i \left(1 - \exp(-\sigma_i \delta_i)\right) c_i, \quad \text{where} \quad T_i = \exp\left(-\sum_{j=1}^{i-1} \sigma_j \delta_j\right), \tag{2.8}$$

and $\delta_i = t_{i+1} - t_i$ is the distance between adjacent sample points along the ray. NeRF is tasked to learn the volume density σ_i and color c_i at any sample point $r(t_i)$ along a ray and is trained using the following loss function:

$$\mathcal{L} = \sum_{r \in R} \left\| \hat{C}(r) - C(r) \right\|_2^2, \tag{2.9}$$

where $\hat{C}(r)$ is the predicted color and $C(r)$ is the ground truth color along ray r. In practice, NeRF uses two networks—a coarse network to estimate a rough global structure of the scene, and a fine network that focuses on higher-resolution details by sampling more densely in regions of interest identified by the coarse model.

CHAPTER 3

RELATED WORK

3.1 Wireless (WiFi) channel prediction using NeRFs

`NeWRF`[16] and `NeRF2`[15] are recent papers that have used NeRFs to predict the wireless channel impulse response (CIR) [23] at unknown locations inside a room. Drawing a parallel to optical NeRFs, a voxel's color in optics becomes a voxel's transmit power in wireless. The voxel's density in optics remains the same in wireless, modeling how that voxel attenuates signals passing through it. NeRF2 and NeWRF *learn to assign transmit power and attenuation to each voxel* such that the measured CIR is matched (i.e., L_2 loss minimized) over training locations inside a room. The authors explain that voxels assigned non-zero transmit power will be called *virtual transmitters*, and will model the reflections in the environment. However, *many assignments are possible that fit the CIR training data*, especially when the data is sparse (Figure 3.1 illustrates two possible assignments). While the predicted CIRs could achieve low error for all such assignments, only one of the assignments will model the true reflections, forcing the virtual transmitters to be located on the wall surfaces (Figure 3.1(b)). We have plotted `NeRF2` and `NeWRF`'s assignment of voxel densities (see Figure 3.1(c,d)) to confirm that their high accuracy in CIR prediction is not an outcome of correctly learning the wall layout. `NeRFMap`'s goal is to assign the voxel densities correctly using the simple NeRF framework, which should naturally yield the floorplan.

3.2 Neural radiance fields for audio

Another exciting line of research focuses on predicting room impulse response (RIR) for audio [17, 24, 18, 25]. Neural Acoustic Field (NAF) [17] extended

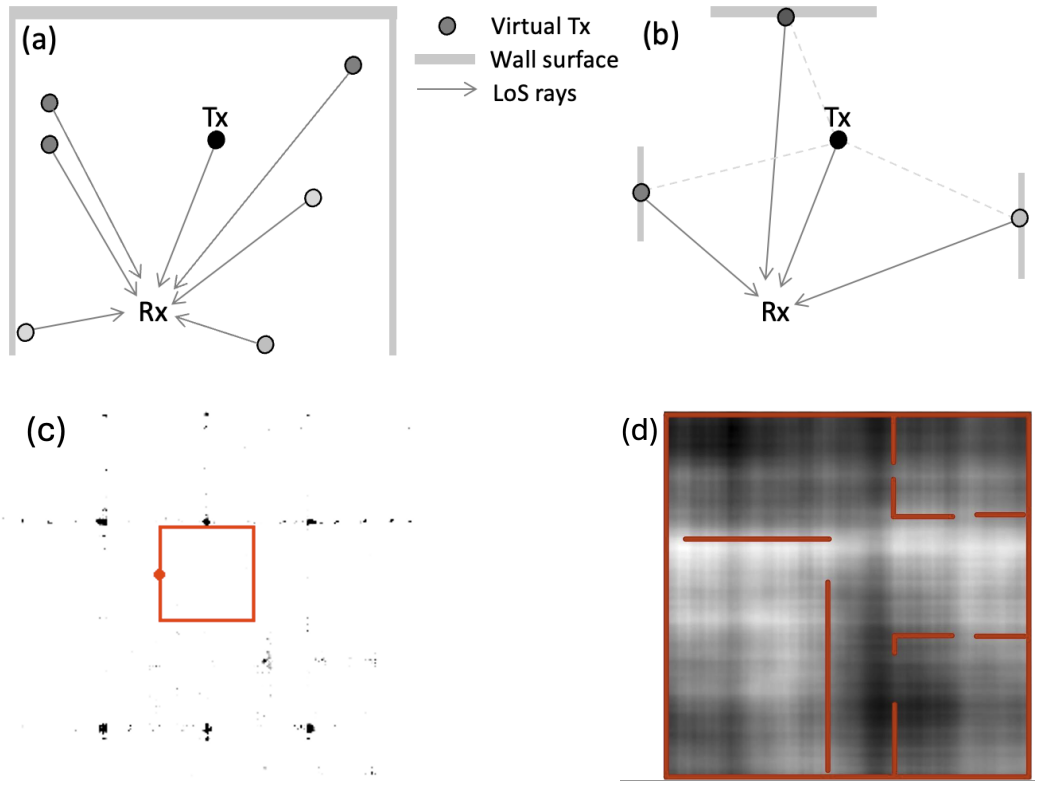

Figure 3.1: (a) Fitting signal power using virtual Txs. (b) Ideally, virtual Txs should be located on wall surfaces. (c) Sparse virtual Txs learnt by NeWRF shown in black/gray dots. (d) Dense virtual Txs learnt by NeRF2. Neither correspond to the true floorplan (shown in red).

the classical NeRF to train on RIR measurements in a room and predict the RIR (magnitude and phase) at new $\langle Tx, Rx \rangle$ locations. NAF identified the possibility of overfitting to the RIR and proposed to learn, jointly, the local geometric features of the environment (as spatial latents) and the NAF parameters. The spatial latents embed floorplan information, but a decoder needs to be trained using the partial floorplan data. `NeRFMap` requires no floorplan supervision, and secondly, relies entirely on signal power (less informative than RIR) to estimate the floorplan.

Follow up works have focused more on improving the RIR accuracy and have utilized explicit information about the surrounding, namely pictures [26, 24, 18, 27], LIDAR scans [28], or optical NeRFs [25], to better guide their MLPs. RIR prediction results are steadily improving; however, this sequence of ideas is veering away from the inverse nature of the problem. Our goal is to first invert the signal power to a floorplan, which can then enable CIR/RIR predictions.

3.3 Modeling reflections in optical NeRFs

Optical NeRFs have tackled reflections [29, 30, 31] for synthesizing glossy surfaces and mirrors, and for re-lighting [32, 33]. NeRFRen [34] proposes to decompose a viewed image into a transmitted and a reflected component. To synthesize novel views, the two component images (learnt by separated branches of the MLP) are added with learnt weights to appropriately suppress the reflected component. Ref-NeRF [35] also focuses on reflections through a similar decomposition of the transmitted and reflected color, however, the reflected color is modeled as a function of the viewing angle and the normal to the surface, resulting in accurate models of specular reflection. Several other papers have developed similar ideas [36, 37] and the core insight centers around solving a 2-component decomposition problem. `NeRFMap` faces the challenge of not knowing the number of rays adding up from all possible directions in the environment. Hence, `NeRFMap` must solve a many-component decomposition problem using the physics of signal propagation (an RF specific opportunity).

3.4 Floorplan Estimation

Finally, floorplan estimation, as a technology, has matured significantly with RGBD cameras [38, 1, 2, 3, 4, 6], LIDARs [7, 8], and even smartphone cameras (in Apple's ARKit) [39]. Early methods focused on hand-crafted geometric features and layout priors [40, 41, 38, 42], leveraging cues such as vanishing points, edge maps, and planar surfaces to infer room structures. More recent approaches have shifted toward deep learning-based techniques, using neural networks to directly predict layouts. Some of these method use visual data for segmentation [5], layout regression [3, 1], while others leverage panoromic images [2, 4] and 3D point clouds [43, 44] to perform similar tasks. However, using visual sensors indoors can be invasive to the user's privacy; `PhyMap` and `NeRFMap` aims to infer floorplans through less invasive RF signals. More importantly, for `NeRFMap`, we are keen on helping NeRFs learn wireless signal propagation, especially first-order reflections; floorplan estimation is one application of this learning exercise.

CHAPTER 4

PHYMAP

4.1 Setup and Overview

We aim to infer a 2D floorplan from WiFi measurements of received signal strength indicator (RSSI) at various locations within an indoor environment. Let ϕ be a binary matrix of size $L \times L$, representing the indoor floorplan, where L is the maximum length of the floorplan. The training data consists of WiFi RSSI measurements $\{\psi(Rx_i)\}$ at a set of receiver locations $\{Rx_i\}$, corresponding to various trajectory locations traversed by the user. Note that we do not assume the locations of the (fixed) transmitter Tx, in fact, `PhyMap` is designed to predict the transmitter location as well. The goal is to learn a model that can map these measurements to the 2D floorplan ϕ. The signal power ψ, received by a receiver Rx (with the subscripts omitted for generality), can be modeled as:

$$\psi = \psi_{LoS} + \psi_{ref_1} + \cdots + \psi_{ref_n}$$

where ψ_{LoS} is the power from the direct line-of-sight (LoS) path, and ψ_{Ref_k} is the aggregate power from all k^{th} order reflections (i.e., all signal paths that underwent exactly k reflections before arriving at the Rx). We ignore the effects of diffraction and scattering in this work. We assume that the RSSI at location Rx is independent of time and is only a function of the obstacles, the transmitter, and the receiver. Additionally, to leverage the inherent inductive biases through convolutions, we convert these receiver locations $\{Rx_i\}$ and measurements $\{\psi(Rx_i)\}$ into 2D matrices R and Ψ, respectively. To summarize, the task is then to map the WiFi measurements Ψ to the 2D floorplan ϕ through a learning model.

Given the complexity of high-dimensional floorplan images, we first project

these images onto a lower-dimensional latent space. These latent representations capture the critical features of indoor layouts and are subsequently utilized to map received RSSI measurements to feasible floorplans. Since multiple distinct floorplans can produce nearly identical WiFi RSSI measurements, especially when the number of measurements is limited, identifying a unique floorplan is challenging. Additionally, minor structural changes, such as adding or removing a wall far from the transmitter, can often lead to negligible variations in signal strength, necessitating a model capable of generating multiple plausible floorplans.

In `PhyMap`, we train a model that maps RSSI data onto this lower-dimensional latent representation space. This involves learning a direct relationship between observed RSSI values at receiver locations and the corresponding floorplan latents. In order to map to multiple floorplans that can explain the same set of RSSI measurements, we design the mapping to learn a distribution over the floorplan latents that we can sample from. To accommodate multiple floorplans consistent with the same RSSI measurements, we design this mapping to learn a distribution over the latent space, enabling sampling of diverse plausible floorplans. Although the model can infer floorplans upon training, verifying the accuracy of these inferred floorplans, especially in real-world scenarios, remains challenging. To mitigate this, we introduce an additional physics-based model, `PhyModel`, specifically designed to model physical wave propagation. That is, our `PhyModel` is tasked with predicting RSSI values from a given floorplan, receiver trajectories, and transmitter locations. Once trained, this physics-based model serves as guidance to refine latent representations during the `PhyMap`'s training phase. An ideal `PhyModel` should accurately model the physics of wave propagation in the environment, which would effectively resolve the ambiguity in the inverse problem by fully explaining the observed measurements. Since `PhyModel` is differentiable after training, we can optimize the floorplan latents even at inference time to identify the floorplan that best aligns with actual measurements, enabling `PhyMap` to support test-time optimization.

4.2 Encoding Floorplans

Indoor environments are inherently complex, often cluttered with walls, furniture, and other obstacles. Generating accurate floorplan images in high-dimensional space is computationally challenging, and the ambiguity arises when multiple floorplans can match the input set of WiFi RSSI measurements. To address this, our approach focuses on projecting the high-dimensional floorplan into lower-dimensional latent representations that is equipped with the ability to sample these multiple floorplans. To this end, we propose using the Vector Quantized-Variational AutoEncoder (VQ-VAE) [45] to encode the floorplan into a latent space. Given a set of floorplans Φ, we begin by encoding each floorplan $\phi \in \Phi$ onto a lower-dimensional latent space using an encoder E_1. Every floor plan ϕ is represented via a subset of a discrete codebook of vectors. Specifically, let $c_k \in \mathbb{R}^d$ be a latent vector, and let $C \in \mathbb{R}^{K \times d}$ denote a shared embedding matrix of K such latent vectors. The floorplan ϕ is first passed through the encoder E_1 to obtain a continuous latent representation, $E_1(\phi) \in \mathbb{R}^{h \times w \times d}$, which results in $h \times w$ vectors in the latent space. Each latent vector $z_e(j) \in E_1(\phi)$ is then discretized by finding its nearest neighbor from the codebook in the following manner:

$$z_q(j) = c_{k^*} \quad \text{where} \quad k^* = \underset{k \in \{1,\ldots,K\}}{\operatorname{argmin}} ||z_e(j) - c_k||_2 \quad \forall z_e(j) \in E_1(\phi)$$

Both the codebook and the encoder-decoder are learned together during the training process and the gradient non-differentiability that happens due to quantization is approximated with a simple straight-through gradient. The discretized vectors $z_q(j), j = \{1, \ldots, h \times w\}$ are then sent to a decoder D_1 to reconstruct ϕ, and the model is trained using the reconstruction loss along with two latent losses to ensure both the codebook and the encoder's outputs snap into each other simultaneously. Once trained, we freeze the encoder E_1, decoder D_1, and the codebook C. Figure 4.1 represents the VQ-VAE model.

4.3 Method

Floorplans and RSSI measurements are two different representations of the same underlying environment, where the RSSI measurements directly stem

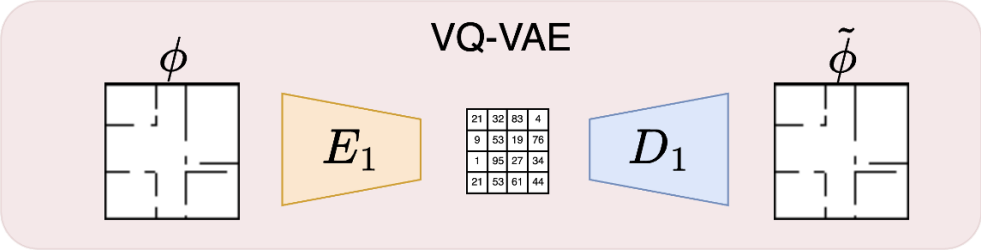

Figure 4.1: VQ-VAE model showing the codebook indices

from the floorplan in which they are measured. For example, recording WiFi measurements and audio measurements in the same floorplan are both observations from the physical space, where RF and audio wave propagation occur within the same environment. These two modalities, though different in nature—one focusing on radio frequencies and the other on sound—are both influenced by the same latent factors, such as the layout of walls, furniture, and obstacles. So once we obtain the latent representation of the floorplans, we utilize an additional encoder, E_2, to project the input WiFi RSSI matrix R onto the latent space of the floorplans. More specifically, E_2 outputs a softmax probability distribution over the codebook elements c_k. The encoder's output is then represented as $E_2(\Psi) \in \mathbb{R}^{h \times w \times K}$, where K denotes the number of codebook elements. This step allows us to map the received WiFi measurements onto the distribution of discrete floorplan latents. We note that this representation of E_2 allows us to sample multiple floorplans that we can decode by first sampling from the softmax probabilities, and using the trained decoder D_1 and codebook C.

We train the encoder E_2 using supervision. Additionally, the encoder is tasked with predicting the transmitter's (Tx) location within the floorplan. This prediction is essential, as detailed in the subsequent section. We define the effective loss function $\tilde{\mathcal{L}}$ as follows:

$$\tilde{\mathcal{L}} = \mathcal{L}_\phi + \lambda_1 \mathcal{L}_{WiFi} \tag{4.1}$$

where $\mathcal{L}_\phi$ is the loss for snapping onto the codebook indices, $\mathcal{L}_{WiFi}$ is the error in Tx's location, and λ_1 is a hyperparameter. By optimizing the loss function $\tilde{\mathcal{L}}$, we obtain estimates of the floorplan through training encoder E_2. This involves sampling the set of argmax indices for each latent dimen-

sion $j \in \{1, \ldots, h \times w\}$, retrieving the corresponding latent vectors from the codebook C to form $Z_q \in \mathbb{R}^{h \times w \times d}$, and decoding the estimated floorplan $\hat{\phi}$ through the decoder D_1 as $\hat{\phi} = D_1(Z_q)$. However, we observe that the resulting floorplan estimates may not always accurately reflect the true environmental layout (see details in Figure 4.6 in Section 4.7). Moreover, without additional supervision or explicit constraints on the encoder E_2, the model can adapt to fit any arbitrary function that minimizes the error, potentially leading to overfitting. This situation can lead to a model that fails to capture the forward propagation process i.e., the physical ray-tracing interactions inherent to the environment. To mitigate this, we introduce a separate physics-informed model, denoted as `PhyModel`, designed to guide encoder E_2 toward predicting more physically accurate floorplans.

4.4 `PhyModel`

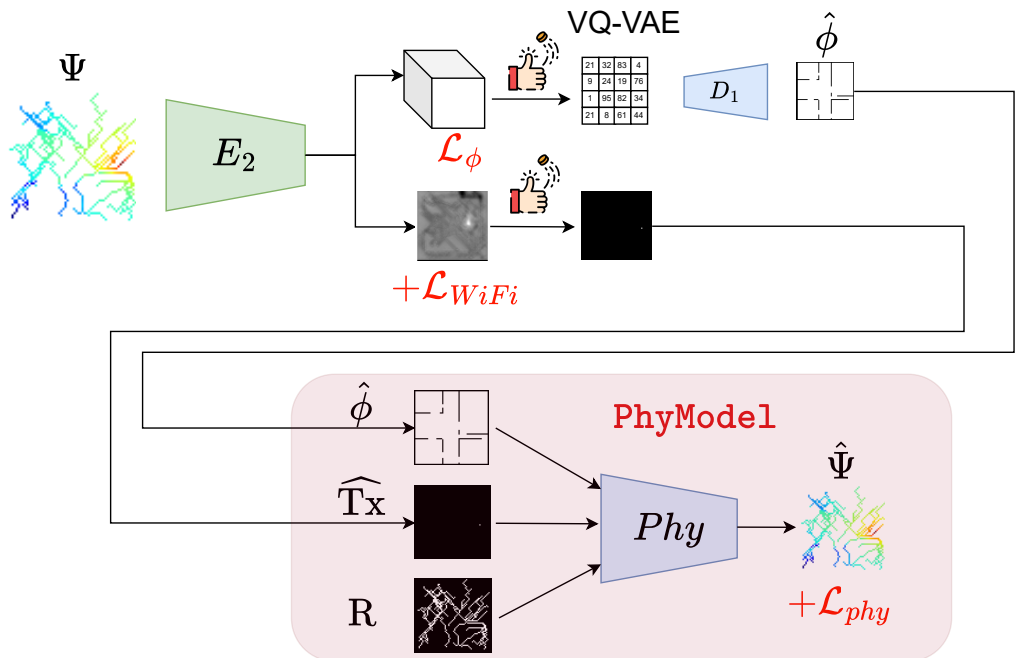

Figure 4.2: `PhyMap` model architecture

The core idea behind the `PhyModel` is to design a module that understands the physics of wave propagation, including modeling line-of-sight attenuation, reflection across wall surfaces. This physics-informed understanding is then used to guide the encoder E_2 in solving the true inverse problem: inferring

more accurate floorplans. Ideally, if one had access to a differentiable emulator that accurately mirrors the forward signal propagation—including both penetration and reflection effects—and supports backpropagation, it could be used to refine the floorplan estimate $\hat{\phi}$. Such an emulator would take the current estimate $\hat{\phi}$ and the ground truth transmitter location Tx as inputs (assuming the oracle has access to the ground truth Tx), and output the predicted RSSI values at receiver locations. The resulting gradients could then be propagated back through the encoder E_2 to iteratively minimize the RSSI prediction error. However, access to such an oracle that supports differentiable ray tracing and knowledge of the transmitter's location is not feasible in practice. As a result, we pursue an alternative approach: training a model end-to-end without relying on explicit ray tracing. The goal of our `PhyModel` is to predict RSSI values at the receiver locations given the floorplan and transmitter location. We train the `PhyModel` using the following loss function:

$$\mathcal{L}_{phy} = ||\Psi - \hat{\Psi}||_2^2 \tag{4.2}$$

where λ_1, λ_2 are hyperparameters.

Intuitively, `PhyModel` can be interpreted as a discriminator, analogous to the critic in Wasserstein GANs [46]. Rather than merely assessing whether the predicted floorplan $\hat{\phi}$ resembles the actual floorplan ϕ, it evaluates how well the prediction aligns with the underlying RSSI measurements. We also note that given a floorplan and transmitter location, every point within the environment implicitly defines an RSSI value, whether observed or not. In this setup, receiver locations serve as evaluation points for a spatial RSSI function conditioned on the floorplan and transmitter location. Thus, RSSI values corresponding to different receiver trajectories can be derived by applying relevant spatial masks to this continuous function. So our input to `PhyModel` comprises the floorplan and transmitter location, and the model outputs the RSSI values at all potential receiver positions. Once this physical model is trained, specific receiver locations query the predicted RSSI field using a binary mask, extracting the expected RSSI values for those positions of interest. We present two variants of the `PhyModel` model: one based on a ResNet architecture [47] and another utilizing a Transformer architecture [48].

Figure 4.2 represents our PhyMap model. Thus, the loss function for training the PhyMap model is given by:

$$\mathcal{L} = \mathcal{L}_{\phi} + \lambda_1 \mathcal{L}_{WiFi} + \lambda_2 \mathcal{L}_{phy} \tag{4.3}$$

where λ_1, λ_2 are hyperparameters.

4.4.1 Estimating Tx

PhyModel requires the transmitter location as an input along with the floorplan to predict the RSSI values any receiver location. However, during test time, the true transmitter location is not available. To address this, we predict the transmitter location during training as a distribution over the floorplan using the encoder E_2. We then sample from this distribution to obtain a single point as the estimated transmitter location. To allow gradients to propagate through this sampling step, we utilize the Gumbel-softmax operation [49], which approximates the non-differentiable argmax with a differentiable alternative through a temperature controlled softmax function. See Figure 4.2.

4.5 Test Time Optimization

Once PhyMap is trained, it can be used to infer the floorplan given the WiFi measurements. Since PhyMap is capable of generating multiple plausible floorplans, we can sample a set of candidates floorplans and evaluate them against the measurements using the pretrained PhyModel. Additionally, one can also perform test-time optimization by refining the predicted floorplan through backpropagation, minimizing the loss in 4.2, assuming backpropagation through the PhyModel is feasible at test time. This is achieved by first freezing the PhyModel, and then optimizing the encoder E_2 to minimize the RSSI reconstruction loss in 4.2 with respect to the predicted floorplan $\hat{\phi}$.

4.6 Implementation Details

In this section, we provide an overview of the training pipeline and the key design choices underlying our implementation. The training process involves three main stages: (1) pretraining the VQ-VAE model, (2) training the `PhyModel` to predict RSSI values, and (3) training the final `PhyMap` model to predict the floorplan from RSSI measurements.

We assume the floorplan is of an unknown shape inside a 256×256 grid. No assumptions are made regarding the trajectory lengths taken by users collecting the RSSI measurements. Trajectories are collected using the astar algorithm [50] to simulate a walking trajectory that traverses all rooms. Throughout all stages, we use the Adam optimizer [51] with a learning rate of 10^{-4}, a batch size of 32, weight decay of 10^{-4}. All models are implemented in PyTorch and trained on NVIDIA A100 GPUs.

4.6.1 Pretraining VQ-VAE

In the first stage, we train the VQ-VAE to encode floorplan images $\{\phi\}$ into a discrete latent space. We use a codebook of size 128 and set the latent dimension to 20. Results and ablations are in Section 4.7. The VQ-VAE is optimized to reconstruct input floorplans along with a codebook loss (to train the codebook) and a commitment loss (to ensure the encoder's output snaps to the codebook). To address the non-differentiability of the quantization step, we use the straight-through estimator [52] to allow gradient flow during training.

4.6.2 `PhyModel` Training

The second stage involves training `PhyModel`, which learns to predict the RSSI values given a floorplan and transmitter location at the receiver locations. As mentioned previoisly in Section 4.4, rather than restricting predictions to receiver locations alone, we predict the RSSI field across the entire floorplan. This dense prediction strategy stabilizes training by capturing the overall spatial structure of signal propagation and makes the RSSI field smoother. To maintain differentiability, we sample from the latent space

using Gumbel-softmax, enabling gradients to pass through the otherwise discrete latent selections.

4.6.3 Training `PhyMap`

In the final stage, we train `PhyMap` by optimizing the encoder E_2 to map RSSI measurements to the corresponding latent floorplan representations. We choose $\lambda_1 = 0.1$ and $\lambda_2 = 1$ for our training.

4.7 Results

Floor plan and Trajectory. Floorplans are drawn from the HouseExpo [53], a large-scale dataset of indoor layouts derived from the SUNCG dataset [54]. We also generate synthetic floorplans from apartment layouts and a few layouts with random objects placed inside a room with an outer corridor. Figure 4.3 and 4.4 show the synthetic and HouseExpo floorplans, respectively. In each floorplan, an A^* algorithm [50] is employed to simulate a walking trajectory that traverses all rooms. We pick a random start and end point in each room and connect them using the A^* algorithm. This trajectory simulates the path of a user walking while collecting WiFi measurements on a mobile device.

Wireless Simulation Dataset. We utilize the NVIDIA Sionna RT [55, 19], a differentiable ray tracer for radio propagation modeling, as the platform to compute the ground truth signal power (also known as *received signal strength index* (RSSI)). A transmitter (Tx) is randomly placed within the floorplan, and omnidirectional transmissions are simulated at a frequency of 2.4 GHz. For receiver locations, we sample the user trajectory at a fixed time interval

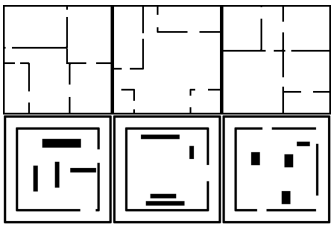

Figure 4.3: synthetic floor plans

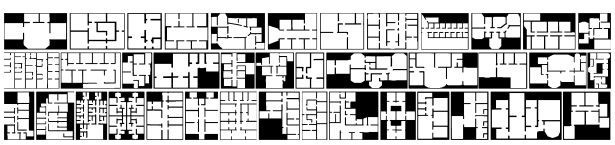

Figure 4.4: HouseExpo floor plans

to obtain R = $\{Rx_i\}$ at N Rx locations. We shoot 10^7 rays into the given floorplan and consider only the rays that reach the receiver location within 5 bounces (i.e., up to 5th-order reflections); others attenuate excessively and are negligible. The default materials for the walls are used in the simulation. As with most WiFi simulators, Sionna does not model signal penetration through walls – this means that a Tx and Rx located on opposite sides of a wall will not receive any line of sight signal. Using Sionna, we obtain signal power matrix $\Psi = \{\psi(Rx_i)\}$ by inputting (ϕ, Tx, R). In total, we collect roughly 100,000 samples for training and 10,000 samples for testing.

Metrics. We evaluate using 3 metrics:

■ **Mean Square Error (MSE)**: This metric measures the average squared difference between the predicted and true floorplan images.

■ **Intersection over Union (IoU)** [56]: This metric measures the degree to which the predicted empty regions (non-walls) and the true empty regions (non-walls) superimpose over each other in the 2D floorplan. The following equation defines the metric:

$$\mathrm{IoU} = \frac{EP \cap EP^*}{EP \cup EP^*}$$

where EP denotes the set of predicted empty white pixels and WP^* denotes the true empty white pixels. We note that this is a simpler metric compared to the `Wall_IoU` metric defined in Chapter 5.

■ **F1 score** [57]: Defined as $F1 = \frac{2 \times P \times R}{P+R}$, where P is the *precision* and R is the *recall* of the bitmap. P and R are defined based on wall pixels, similar as above.

4.7.1 VQ-VAE

Figure 4.5 shows the results of the VQ-VAE models for syntheic and House-Expo floorplans with the codebook size 128 and the latent dimension of 20. The β parameter for the commitment loss is set to 0.1. Ablatoins are performed on the codebook size, latent dimension, and the β parameter (see Table 4.1). We also train a vanilla Variational AutoEncoder (VAE) model with a latent dimension of 1000 and 2000 and compare it with the VQ-VAE model. VQ-VAE is able to reconstruct the floorplans better than the VAE

model. The best model is able to reconstruct the floorplan with an MSE of 0.0047, an IoU of 0.94, and an F1 score of 0.97.

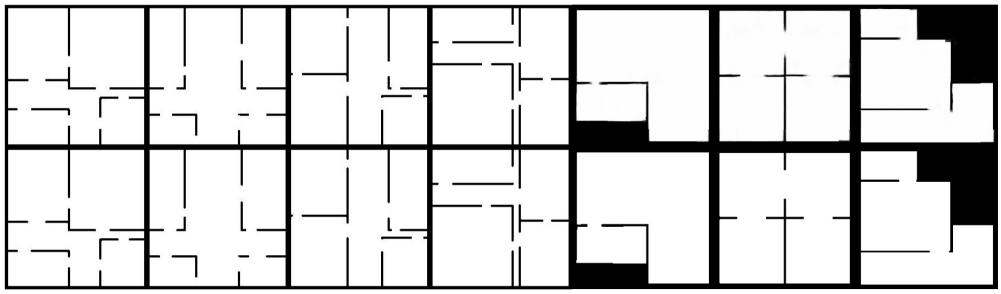

Figure 4.5: VQ-VAE model on syntheic floorplans: On the top is the predicted truth floorplan, at the bottom is the ground floorplan.

Table 4.1: Comparison of MSE, IoU, and Dice Coefficient for VQ-VAE and VAE models.

Method	**MSE**	**IoU**	**F1 Score**
VQ-VAE:			
codebook size = 32×5, $\beta = 0.25$	0.0083	0.9015	0.9482
codebook size = 128×20, $\beta = 0.25$	0.0069	0.9169	0.9567
codebook size = 128×20, $\beta = 1$	0.0066	0.9203	0.9585
codebook size = 128×20, $\beta = 0.1$	**0.0047**	**0.9418**	**0.9700**
codebook size = 128×20, $\beta = 1000$	0.0877	0.2869	0.4458
VAE:			
latent dim = 1000	0.0062	0.9221	0.9595
latent dim = 2000	0.0062	0.9200	0.9584

4.7.2 `PhyMap`

Figure 4.6 illustrates the performance of `PhyMap` when trained without the assistance of the physics-based model 4.6. While the model successfully captures key structures such as corridors, door openings, and few wall segments, it fails to recover the entire floorplan accurately. This limitation arises from directly mapping RSSI measurements to a floorplan without incorporating physical constraints. As a result, the model occasionally introduces artifacts in open spaces that do not intersect receiver trajectories, as seen in the third column of Figure 4.6.

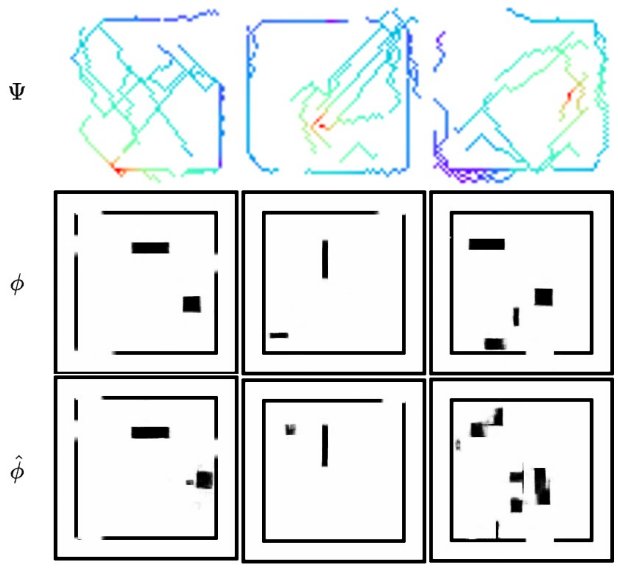

Figure 4.6: `PhyMap` results on synthetic floorplans optimizing $\tilde{\mathcal{L}}$ in 4.1, i.e., without guidance from `PhyModel`.

4.7.3 `PhyModel`

We implement `PhyModel` using a ResNet50 architecture [47]. The model is trained to predict the RSSI field across the entire floorplan, conditioned on the floorplan layout ϕ and the transmitter location Tx provided as binary masks. Predicted RSSI values are then masked using receiver location maps R to compute the training loss. Figure 4.7 shows the predicted RSSI maps by `PhyModel` on synthetic layouts. The results demonstrate that the model is able to predict RSSI at the receiver locations accurately (fourth row) comparable to the ground truth (third row). The last row also showcases model's understanding of line-of-sight propagation and attenuation due to obstructions such as signals on the opposite sides of a wall varying vastly. Additionally, inverse square law behavior is captured as circular attenuation patterns radiating from the transmitter location. Some reflected signal paths are also visible, for example, in the top corridor of the second column, where signals reflect and decay smoothly, although the direct path is blocked by the corridor wall. `PhyModel` achieves a mean squared error (MSE) of 5.18 dB in RSSI prediction.

With `PhyModel` trained, we train `PhyMap` using the composite loss function in Equation (4.3). Figure 4.8 presents the resulting floorplan predictions. Compared to the model trained without `PhyModel` as seen in Figure 4.6,

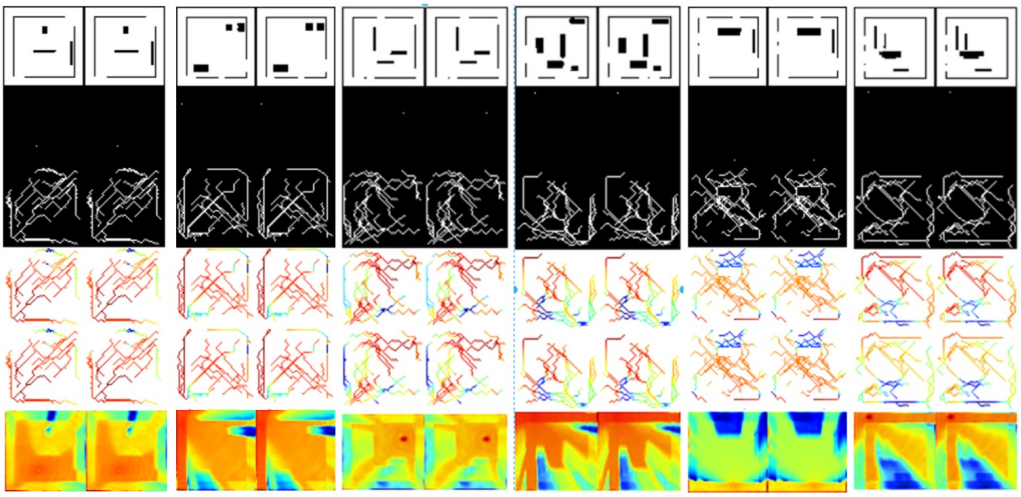

Figure 4.7: `PhyMap` results on synthetic floorplans. Each floorplan has two colunmns: the first column shows results on the predicted floorplan from the VQ-VAE stage and second column is the ground truth floorplan. First row is the ground truth floorplan, second row is the transmitter's location, third row is the receivers locations, fourth row is ground truth RSSI values, fifth row is the predicted RSSI values, and the last row is the predicted RSSI field. Darker red indicates stronger signal, and blue indicates weaker signal.

we observe improved structural consistency and reduced artifacts. Quantitatively, incorporating `PhyModel` leads to a 4% improvement in IoU and a 2.1% boost in F1 score (Table 4.2). Though the gain may appear modest, the IoU metric tends to favor large empty regions, which both models handle well.

Observe that there are still few missing and extra walls in the predicted floorplan, for examnpole, the last two columns. We can improve on these results by performing test-time optimization using the `PhyModel` to refine the predicted floorplan as discuessed in Section 4.5. Figure 4.9 visualizes this refinement process, where an initial floorplan prediction $\hat{\phi}_{\text{start}}$ is improved by minimizing the RSSI error via backpropagation using `PhyModel`. The optimized floorplan $\hat{\phi}_{\text{end}}$ exhibits stronger alignment with the ground truth. For example, the small horizontal wall in the first column of $\hat{\phi}_{\text{start}}$ is adjusted to match the actual layout by making it vertical. The model is also able to resolve gaps (third column) and add missing partitions (fifth and sixth columns). Correspondingly, the predicted RSSI maps Ψ_{end} more closely resemble the ground truth Ψ (compare third and fifth rows with the first row).

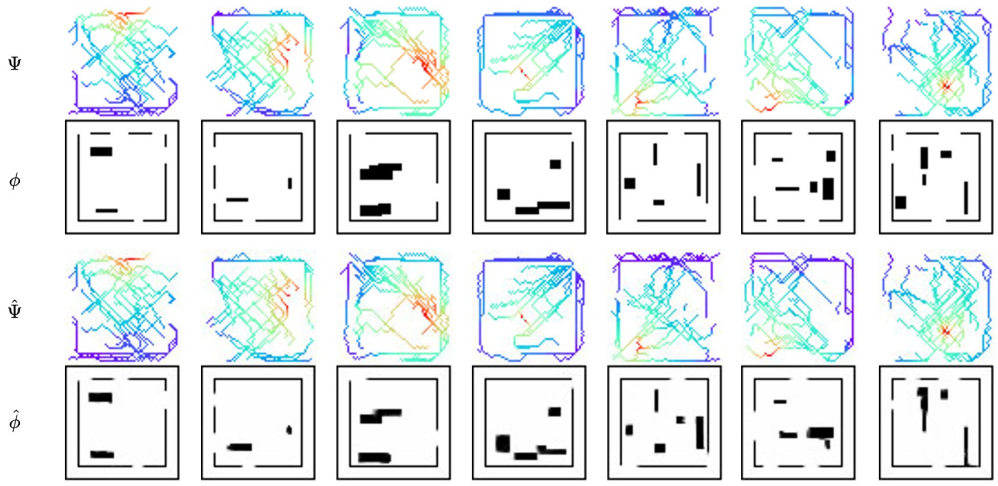

Figure 4.8: `PhyMap` results on synthetic floorplans when trained with `PhyModel`, optimizing Equation (4.3). Top row is the input RSSI measurements, second row is the ground truth floorplan, third row is the predicted RSSI values, and the last row is the predicted floorplan.

Table 4.2: Comparison of IoU and F1 Score for the models with and without `PhyModel`.

Method	**IoU**	**F1 Score**
No `PhyModel` 4.1	0.852	0.893
`PhyMap` **4.3**	0.886	0.912

4.8 Discussion

While the previous section demonstrates the promise of `PhyMap` in predicting floorplans, we observe that the inclusion of `PhyModel` does not consistently improve results across all examples. In particular, `PhyModel` often fails to capture the full complexity of the Room Impulse Response (RIR) and physical propagation effects such as multipath reflections, thereby limiting its effectiveness in guiding accurate floorplan inference.

Even though `PhyModel` can predict RSSI values at receiver locations with reasonable accuracy, we find that some alternative floorplans—distinct from the ground truth—can yield lower loss values. To illustrate this, Figure 4.10 presents contrastive examples where `PhyModel` assigns lower error to incorrect floorplans. Each set of images consists of two columns: the first column shows the floorplan with the lowest loss as predicted by `PhyModel`, while the second column displays the next-best floorplan in terms of loss. In the

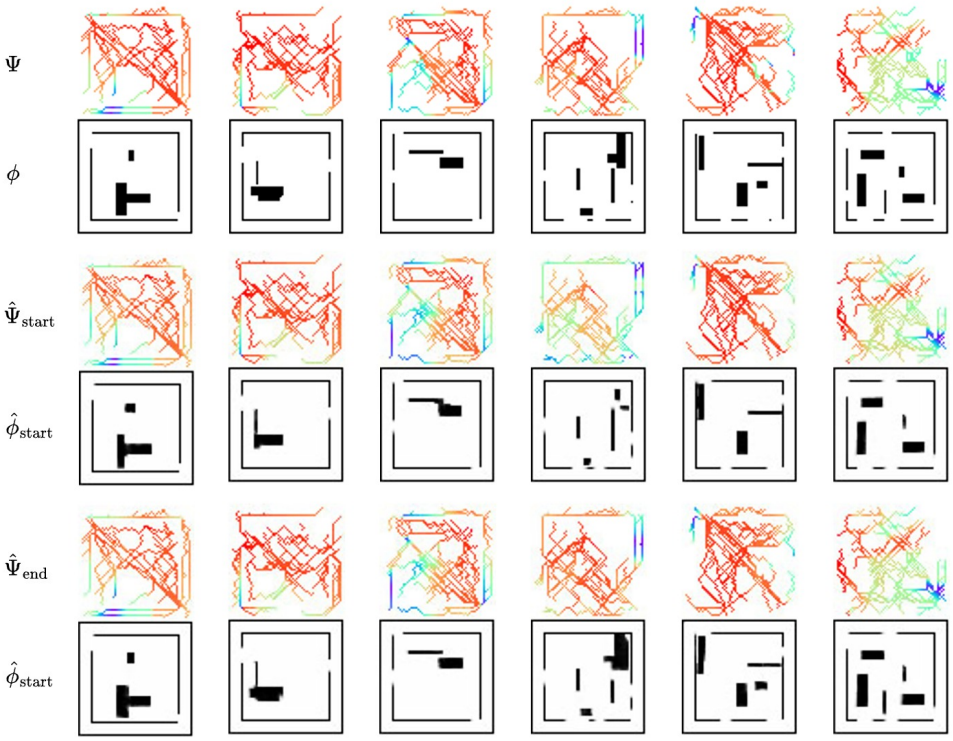

Figure 4.9: `PhyMap` Test time optimization results on synthetic floorplans optimizing for floorplan using `PhyModel`. Top two row shows the input RSSI measurements and the ground truth flooplans, while the third and fourth shows the predicted RSSI values and the predicted floorplan at the start of the optimization. The last two rows show the predicted RSSI values and the predicted floorplan at the end of the optimization.

first two sets, the floorplan with the lowest loss is identical or very close to the ground truth layout, making them successful examples. In such cases, `PhyModel` can effectively guide `PhyMap` to converge to the correct floorplan during optimization. However, the remaining three sets on the right half illustrate contrastive examples. Here, the floorplan with the lowest loss (first column) deviates significantly from the ground truth (second column). These failure cases show that optimizing with `PhyModel` can guide `PhyMap` toward suboptimal or incorrect floorplans.

To improve `PhyModel`'s predictive capability, we explored two architectural modifications: a two-component decomposition of the signal power into line-of-sight and reflected signals, and a Dense Prediction Transformer (DPT)[58] model. However, these changes did not yield substantial improvements in the performance of `PhyMap`. Details and results of these experiments are provided

in Appendix A.1.

Overall, these observations highlight a key limitation: many models can accurately fit the observed RSSI data, yet they often fail to ensure that the ground truth floorplan corresponds to the global minimum of the loss function. This suggests that current data-driven approaches, including `PhyModel`, may not be sufficient to resolve ambiguities between multiple plausible floorplans. Furthermore, we find that the performance of `PhyModel` is heavily dependent on the training data, which raises concerns about its ability to generalize to diverse layouts or real-world environments, such as those in the HouseExpo dataset. These limitations lead us to consider a more fundamental question:

Can we design a model that effectively understands wave propagation phenomena—such as line-of-sight transmission and reflections from walls—in a manner that accurately infers the underlying environment?

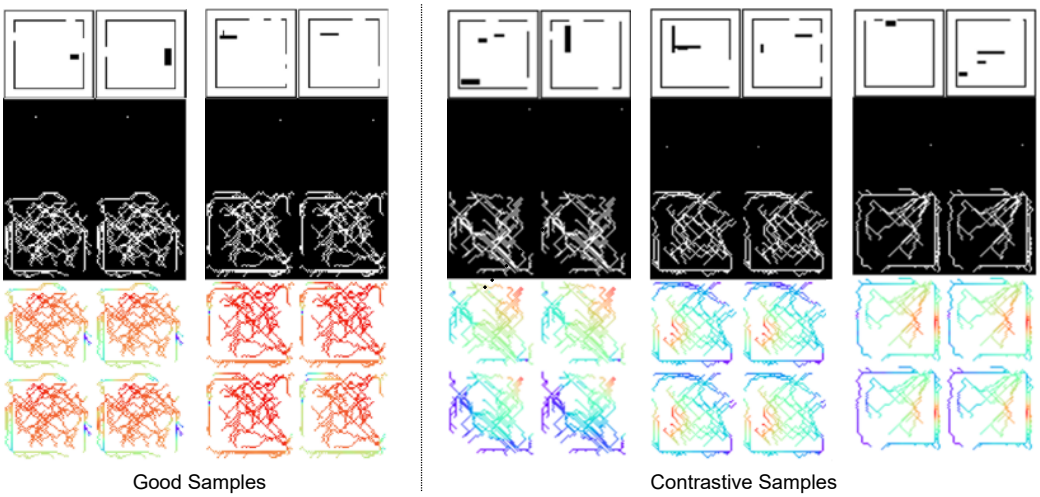

Figure 4.10: Contrastive examples of floorplan predictions using `PhyModel` in `PhyMap`.

In the following chapter, we investigate this question through a new approach based on Neural Radiance Fields (NeRF), which integrates modeling signal propagation and scene geometry directly into the learning framework to truly solve the inverse problem of floorplan inference.

CHAPTER 5

NERFMAP

5.1 Setup and Overview

The signal power ψ, received by a receiver Rx, can be modeled as:

$$\psi = \psi_{LoS} + \psi_{ref_1} + \cdots + \psi_{ref_n}$$

where ψ_{LoS} is the power from the direct line-of-sight (LoS) path, and ψ_{Ref_k} is the aggregate power from all k^{th} order reflections (i.e., all signal paths that underwent exactly k reflections before arriving at the Rx). We move the Rx to N known locations and measure ψ at each of them. We assume M fixed transmitters whose locations are also known. Our goal is to accept $M \times N$ measurements as input and output the 2D floor-plan ϕ, where ϕ is a binary matrix of size $L \times L$, where L denotes the maximum length of the floorplan.

We train `NeRFMap` on the measured data using our proposed *multipath power function* as part of the optimization objective. This function models the LoS and the first-order reflections. Higher order reflections are complex to model for real floorplans; moreover, they contribute, on average, $< 6\%$ of the total power (see Figure 5.2). Hence, we disregard higher order reflections. The network model we use is a remarkably simple MLP designed to predict the density $\delta \in [0, 1]$ and orientation $\omega \in [-\pi, \pi]$ of a specified voxel in the indoor scene (see Figure 5.1). The orientation aids in modeling reflections. The multipath power function – parameterized by voxel attributes $\langle \delta, \omega \rangle$ and the $\langle Tx, Rx \rangle$ locations – models an approximation of the received power ψ at that Rx location. Minimizing L_2 loss of this power across all Rx locations trains the MLP. Plotting out all the voxel densities in 2D gives us the estimated floorplan ϕ.

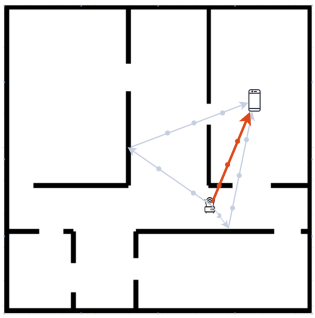

Figure 5.1: A typical line of sight path is shown in red, and first-order reflections are shown in pink. Correspoding voxels are showns as dots.

5.1.1 Approximating Channel with First-Order Reflections

`NeRFMap` models the total received power at the Rx as the combination of the LoS power `NeRFMap_LoS`, and the contributions from all the first-order reflections. To validate the contribution of the achievable power from `NeRFMap` when compared to the total received power ψ, we evaluate the relative contributions of these signals to the total power using the NVIDIA Sionna simulator [55]. To this end, we compute the ratios of the LoS signal ψ_{LoS}, LoS with the first-order reflections $\psi_{LoS} + \psi_{ref_1}$, and LoS with the first two orders of reflections $\psi_{LoS} + \psi_{ref_1} + \psi_{ref_2}$. These are compared to the total received power ψ, which is approximated as the sum of the LoS power and the power from the first ten reflections.

Figure 5.2 shows path power contribution ratio from different paths in histogram. While the ψ_{LoS} power alone only accounts for approximately 70% of the total received power and is more spread out, $\psi_{LoS} + \psi_{ref_1}$ accounts to 95% of the total power, with a reduced spread. Moreover, secondary reflections ψ_{ref_2} only contribute to less than 3% of the total power. Hence, `NeRFMap` models the first-order reflections along with the line-of-sight.

5.2 The LoS Model

Friss' equation [59] from electromagnetics models the free-space received power as $P_r = \frac{K}{d^2}$ where d is the distance of signal propagation, and K is a product of transmit-power, wavelength, and antenna-related constants [59]. We model this free-space (LoS) behavior in the NeRF framework through

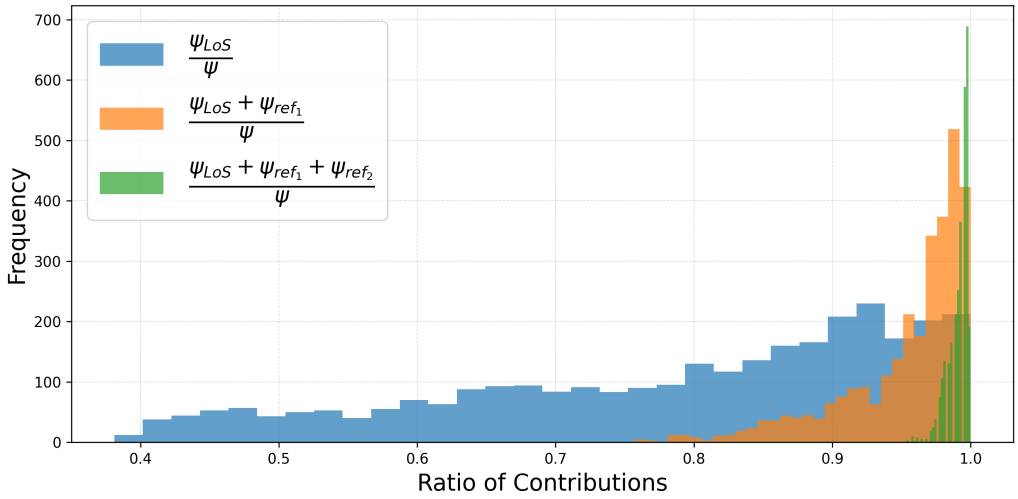

Figure 5.2: Histograms illustrating the contribution ratios from line-of-sight (LoS), LoS combined with first-order reflections, and LoS combined with first- and second-order reflections. The orange graph highlights the significant contribution of first-order reflections to the total power, supporting `NeRFMap`'s approach of modeling only the first reflection alongside the LoS power.

the following equation.

$$\psi_{LoS} = K \frac{\prod_{\{i:v_i \in LoS\}} (1 - \delta_i)}{d^2} \tag{5.1}$$

where K can be empirically measured, and d is the known distance between the $\langle Tx, Rx \rangle$. The numerator includes the product of voxel densities over all voxels along the LoS ray from Tx to Rx (with an abuse of notion, we write this as $v_i \in LoS$). This models occlusions. When the LoS path is completely free of any occlusions (i.e., $\delta_i = 0, \forall i$ where $\{i : v_i \in LoS\}$), we expect the received power to only be attenuated by the pathloss factor d^2 (in the denominator). Equation (5.1) has a slight difference to classical NeRF's volumetric scene function. In our case, voxels along the ray do not contribute to the received power (whereas in NeRF, each voxel's color is aggregated to model the final pixel color at the image). In other words, we have modeled a single transmitter in Equation (5.1).

5.3 The Reflection Model

To model reflections, we consider a given voxel v_j. Whether v_j reflects a ray from the Tx towards Rx depends on (1) the density δ_j of the voxel, (2) the orientation ω_j, (3) the $\langle Tx, Rx \rangle$ locations, and (4) whether the path from Tx to v_j, and from v_j to Rx are both occlusion-free. Parameterized by these, Equation 5.2 models the reflected power at the Rx, denoted as $\psi_{ref}(v_j)$, that arrives after reflection on v_j.

Let us explain this equation briefly. The leading δ_j ensures that voxel v_j is not a reflector when $\delta_j = 0$. The $f(\theta, \beta)$ term models the wave-surface interactions, i.e., how signals get attenuated as a function of the incident angle θ and how signals scatter as a function of the offset angle β between the reflected ray and the direction of the Rx from v_j.

$$\psi_{ref}(v_j) = \delta_j f(\theta, \beta) \frac{\prod\limits_{k \in \{Rx:v_j\}} (1 - \delta_k) \prod\limits_{l \in \{v_j:Tx\}} (1 - \delta_l)}{\left(d_{Tx:v_j} + d_{v_j:Rx}\right)^2} \tag{5.2}$$

The next two product terms ensure that for the Rx to receive this reflection, the voxels along the 2 segments (Tx to v_j, and from v_j to Rx) must be non-opaque; if any δ_k or δ_l equals 1, that reflection path is blocked, producing no power contribution via this voxel v_j to the receiver Rx. Finally, the denominator denotes the total squared distance from Tx to v_j, and from v_j to Rx, modeling the pathloss factor.

To complete the above model, the natural question is: *which voxels are contributing to the received power?* Geometrically, any opaque voxel can be a plausible reflection point between any $\langle Tx, Rx \rangle$ pair. This is because when we consider the triangle between Tx, v_j, and Rx, the voxel orientation ω_j can be assigned a direction that bisects the angle at v_j. For this ω_j, the reflected ray will perfectly arrive at the Rx. Thus, without the knowledge of orientation and density, the total first-order reflection power at the Rx should be modeled as the sum of reflections on all voxels. This makes the optimization problem excessively under-determined.

To cope with this, we assume that surfaces in the environment are orientated discretely in one of K_ω angles. When $K_\omega = 4$, the walls can either be vertical, horizontal, tilted at 45° or tilted at 135°. Under this assumption, the voxels that can produce plausible reflections are far fewer – we call this

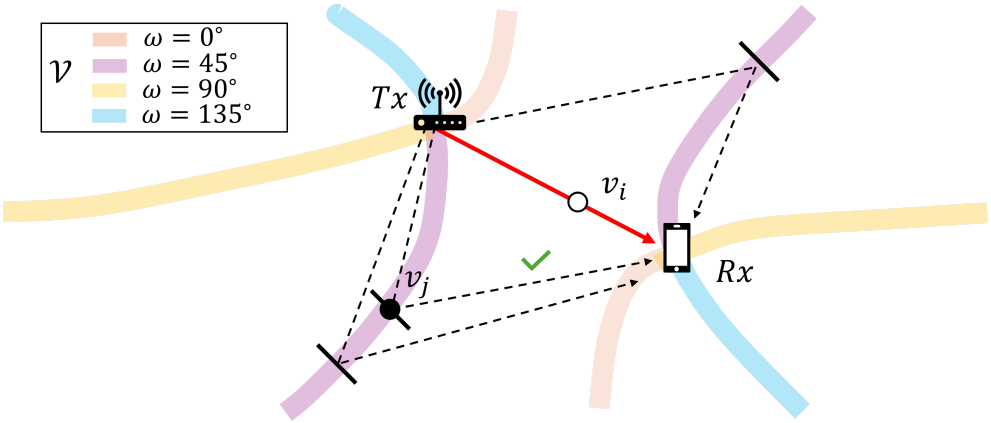

Figure 5.3: Colored stripes define the manifold from which reflections are plausible between $\langle Tx, Rx \rangle$. Voxels located on the manifold form the plausible set $\mathcal{V}$. Dashed lines show plausible reflections.

the "plausible set" $\mathcal{V}$. Figure 5.3 visualizes $\mathcal{V}$ and marks a few examples of plausible reflections from voxels with $\omega_j = 45°$. Equation (5.3) sums up the power from all reflections that occur on the plausible set:

$$\psi_{ref_1} = \sum_{j:v_j \in \mathcal{V}} \psi_{ref}(v_j) \tag{5.3}$$

Thus, the final multipath power function becomes

$$\tilde{\psi} = \psi_{LoS} + \psi_{ref_1} \tag{5.4}$$

5.4 Gradient Issues during Training

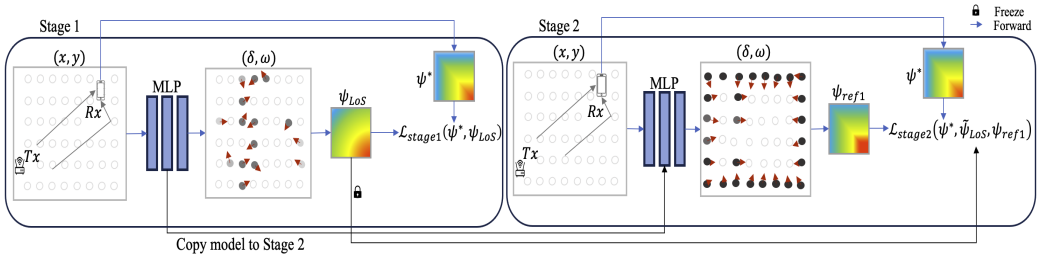

Figure 5.4: `NeRFMap`'s two-stage training approach: In Stage 1, the LoS model is trained using known Rx locations and signal power. This provides a warm-start to the reflection model in Stage 2 which refines the learned voxel densities and orientation. During inference, `NeRFMap` outputs voxel densities at set of queried Rx locations, representing the inferred floorplan.

Training against an L_2 loss, $\mathcal{L} = ||\tilde{\psi} - \psi^*||^2$ [1], did not generate legible floorplans. We found that the ψ_{LoS} dominated the loss term, drowning the reflection model's influence on learning. At a high level, the gradient of the LoS model (Equation (5.1)) w.r.t. δ_i has fewer terms in the numerator's product, and a smaller d^2 in the denominator. The reflection model's gradient w.r.t. δ_i has many more terms since the reflection path is much longer; the denominator is also larger. Since $(1 - \delta_j) \leq 1$, their products force the gradient to decrease geometrically with more terms, causing the reflection gradient to be much smaller compared to LoS.

We formalize this explanation below by considering the LoS and reflection losses individually[2].

$$\mathcal{L}_{LoS} = \left\| \tilde{\psi}_{LoS} - \psi^*_{LoS} \right\|_2^2, \quad \mathcal{L}_{ref} = \left\| \tilde{\psi}_{ref} - \psi^*_{ref} \right\|_2^2$$

Consider the gradient of $\mathcal{L}_{LoS}$ w.r.t the density of v_i.

$$\nabla_{\delta_i} \mathcal{L}_{LoS} = 2(\tilde{\psi}_{LoS} - \psi^*_{LoS}) \sum_{\{n : i \in LoS^{(n)}\}} \nabla_{\delta_i} \psi^{(n)}_{LoS}$$

$$\text{where} \quad \nabla_{\delta_i} \psi^{(n)}_{LoS} = -\frac{K}{d^{(n)2}} \prod_{\substack{j \in LoS^{(n)} \\ j \neq i}} (1 - \delta_j) \tag{5.5}$$

where $LoS^{(n)}$ is the n^{th} LoS path passing through voxel v_i.

The gradient of $\mathcal{L}_{ref}$ has a nearly identical expression with the only difference being many more product terms and that $\nabla_{\delta_i} \psi^{(n)}_{ref}$ depends on where v_i is present in the n-th set of reflection path voxels (denoted by $Ref^{(n)}$). For example,

$$\nabla_{\delta_i} \psi^{(n),Tx}_{ref} = -C \prod_{\substack{j \in Ref^{(n)}_{Tx:v} \\ j \neq i}} (1 - \delta_j) \prod_{k \in Ref^{(n)}_{v:Rx}} (1 - \delta_k) \tag{5.6}$$

$$C = \frac{\delta^{(n)} f^{(n)}(\theta, \beta)}{(d^{(n)}_{Tx:v} + d^{(n)}_{v:Rx})^2}$$

here $\nabla_{\delta_i} \psi^{(n),Tx}_{ref}$ denotes the gradient when v_i is between Tx to the reflection

[1] Here, ψ^* denotes the ground truth signal power

[2] For the ease of explanation, we use ψ^*_{LoS} and ψ^*_{ref} to denote the ground truth LoS and reflection powers, respectively. We do not need to know these terms in practice.

point v.

Finally, since the modeled power approximates the measured power, the residual error will remain non-zero even if the floorplan is accurately learnt. As a result, the optimization is biased towards voxels of higher gradients, i.e., voxels on the LoS path, suppressing the importance of reflections. To address this, we train `NeRFMap` in 2 stages.

5.5 Multi-stage Training

Stage 1: We first use the LoS model against the measured ground truth power ψ^*. This converges quickly because the network easily learns the transparent voxels ($\delta = 0$) that are located along LoS paths. For LoS paths that are occluded, the network *incorrectly* learns excessive opaque voxels between the $\langle Tx, Rx \rangle$, but this does not affect the LoS error since the path is anyway occluded. Hence, the outcome is a crude floorplan but a near-perfect LoS power estimate $\tilde{\psi}_{LoS}$. We utilize this $\tilde{\psi}_{LoS}$ in Stage 2 (discussed soon).

As Stage 1 training progresses, some opaque voxels emerge, offering crude contours of some walls. We estimate a voxel's spatial gradient, $\nabla\delta_i$, and use it to supervise the orientation ω_i of that voxel. The intuition is that a voxel's orientation – needed to model reflections in Stage 2 – is essentially determined by the local surface around that voxel. The gradient $\nabla\delta_i$ offers an opportunity for weak supervision. Thus, the loss for Stage 1 is:

$$\mathcal{L}_{stage1} = \|\psi^* - \psi_{LoS}\|_2^2 + \mathcal{L}^+ \tag{5.7}$$

$$\text{where} \quad \mathcal{L}^+ = \lambda_1 \sum_{\forall j} \|\nabla\delta_j - \omega_j\|_2^2 + \lambda_2 \mathcal{L}_{reg} \tag{5.8}$$

with $\lambda_1, \lambda_2 > 0$ being tunable hyperparameters. The regularization term $\mathcal{L}_{reg}$ will be discussed soon. Finally, the near-perfect estimate of LoS power, denoted $\tilde{\psi}_{LoS}$, is also carried over to Stage 2 to ensure the reflection model is penalized when it veers away from this LoS estimate.

Stage 2 focuses on training the reflection model using the following loss function.

$$\mathcal{L}_{stage2} = \left\|\tilde{\psi}_{LoS} - \psi_{LoS}\right\|_2^2 + \left\|\psi^* - \tilde{\psi}_{LoS} - \psi_{ref_1}\right\|_2^2 + \mathcal{L}^+ \tag{5.9}$$

The first term in the RHS ensures that the Stage 1's LoS estimate is honored in Stage 2. The second term subtracts Stage 1's LoS power from the measured power, $(\psi^* - \tilde{\psi}_{LoS})$; this models the total power *only due to reflections*. Our (first-order) reflection model ψ_{ref_1} is trained to match this aggregate power (L_2 loss). The supervision on orientation and the regularization terms are the same as in Stage 1.

5.6 Regularization

Floorplans demonstrate significant local similarity in orientation; hence we penalize differences in orientation among neighbors, using a regularization (Equation (5.10)) similar to Total Variation [60]. This can be achieved without additional computational cost to the neural network by directly utilizing voxel orientations obtained from each ray.

$$\mathcal{L}_{\text{reg}} = \frac{1}{n_v(n_r - 1)} \sum_{n=1}^{n_v} \sum_{i=1}^{n_r - 1} \|\omega_{n,i+1} - \omega_{n,i}\|_2^2 \tag{5.10}$$

Here, n_v is the number of voxels queried from the plausible set $\mathcal{V}$ and n_r is the number of voxels along each ray.

5.7 Relaxing Tx assumptions

We relax the assumption that Tx locations are known. Given the set of receiver locations $\{\mathbf{Rx}_i\}$ and the signal powers $\{\psi_i^*\}$, the goal is to estimate the transmitter location $\mathbf{Tx} = (\mathbf{Tx_x}, \mathbf{Tx_y})$. To achieve this, we apply a maximum likelihood estimate (MLE). Briefly, among all the measured signal powers $\{\psi_i^*\}$ from a given Tx we identify the P strongest signal powers and their corresponding received locations. The rationale behind selecting the strongest powers is that they are significantly influenced by the LoS component, allowing us to model them effectively only using the Friss' equation [59]. We assume independence among the measurements since the received LoS power across locations, for a given a Tx location, are independent. So,

the likelihood equation for all these P measurements can be written as:

$$p(\psi_1^*, \psi_2^*, \ldots, \psi_P^* | \mathbf{Tx}) = \prod_{i=1}^{P} p(\psi_i^* | \mathbf{Tx})$$

We approximate that the ψ_i^* is normally distributed with a mean modeled by the line-of-sight power $\frac{K}{d_i^2}$ and variance σ^2 where $d_i = ||\mathbf{Tx} - \mathbf{Rx}_i||$ is the distance between $\mathbf{Tx}$ and $\mathbf{Rx}_i$. The likelihood function for each observation ψ_i^* is thus given by:

$$p(\psi_i^* | \mathbf{Tx}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(\psi_i^* - \frac{K}{d_i^2})^2}{2\sigma^2}\right)$$

Maximizing log-likelihood L of $\{\psi_i^*\}\ \forall\ i \in \{1, \ldots, P\}$

$$\log L(\mathbf{Tx}) = \sum_{i=1}^{P} \log\left(\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(\psi_i^* - \frac{K}{d_i^2})^2}{2\sigma^2}\right)\right)$$

$$\log L(\mathbf{Tx}) = -\frac{P}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{P}\left(\psi_i^* - \frac{K}{d_i^2}\right)^2$$

Minimizing the second term gives the optimal $\mathbf{Tx}^*$ as:

$$\mathbf{Tx}^* = \underset{\mathbf{Tx}}{\operatorname{argmin}} \sum_{i=1}^{P}\left(\psi_i^* - \frac{K}{\|\mathbf{Tx} - \mathbf{Rx}_i\|_2^2}\right)^2$$

We use Scipy's 'minimize' with the BFGS method to numerically solve for $\mathbf{Tx}^*$.

5.8 Implementation Details

We assume the floorplan is an unknown shape inside a 512×512 grid. For any ray or ray segment, we uniformly sample $n_r = 64$ voxels on it. We choose $\lambda_1 = \lambda_2 = 0.01$ for LoS training followed by $\lambda_1 = \lambda_2 = 0.1$ for training the `NeRFMap` model. We find that discretized opacity values to $\{0, 1\}$ improve our LoS model. We use the straight-through estimator to avoid the unavailability of the gradient at the discretization step. To help optimization and to encourage sparsity of the number of reflections, we use only the top-

k contributions (with $k = 10$) while training the reflection model. We use the ADAM optimizer [51] with 1.0^{-4} learning rate. We train our models on NVIDIA A100 GPUs.

5.8.1 Additional Details on Model training

The signal power measured at the receiver is typically represented in a logarithmic scale. RSSI values generally range from -50 dB to -120 dB, where a higher value (e.g., -50 dB) corresponds to a stronger signal. Figure 5.5 illustrates a typical input to `NeRFMap` where measurements have been collected from approximately 2000 Rxs positioned in the floorplan, with data gathered from five Txs.

5.8.2 Linear-Scale RSSI Loss:

For the training of `NeRFMap`, we optimize on the linear-scale RSSI values. Linear loss ensures that the receivers that capture stronger signals are given more importance during training. We partition our dataset into an 80-20 split, using 80% of the data for model training, including baselines.

For `NeRFMap`'s network, we employ a simple 8-layer MLP with a hidden dimension of 256 units. For each voxel v_j, the outputs from the final layer are passed through a sigmoid activation to obtain the opacity δ, and through a Gumbel softmax [49] layer to sample the output normal ω from one of the possible K_ω orientations. This sampled orientation is then used in the subsequent stages of training, such as for calculating the direction of the reflected signal M. For the learnable baselines, such as `MLP` and `NeRF2`, we adopt the same architecture as used in `NeRFMap`.

5.8.3 Supervising Voxel Orientations

`NeRFMap` leverages the spatial gradient of a voxel's opacity, $\nabla\delta$, to supervise its orientation during the multi-stage training process. To compute this gradient, we evaluate the opacities of neighboring voxels along each of the K_ω directions and apply finite difference methods. We found that this approach yielded superior results compared to using the gradient available via

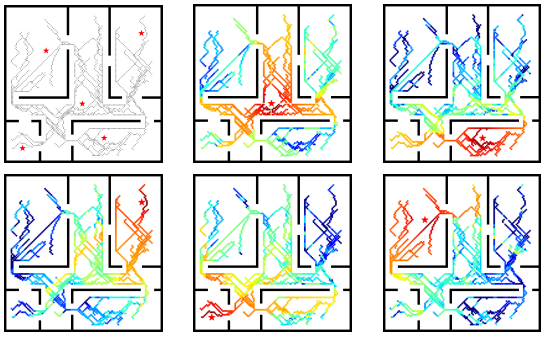

Figure 5.5: Observed signal power at the Rxs . The top left figure shows the positions of the Rxs and Txs, followed by the power at the receivers from each of the five transmitters. The colormap ranges from red showing stronger signals to blue for weaker signals.

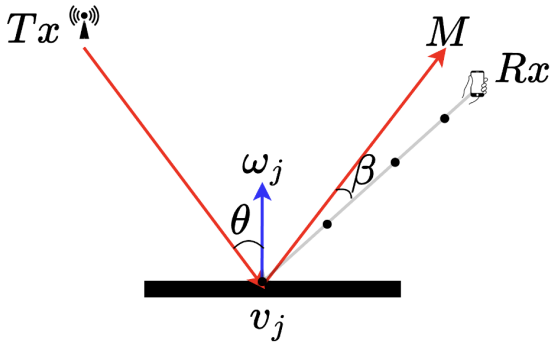

Figure 5.6: An incoming ray from a transmitter Tx reflecting around voxel v_j and arriving at receiver Rx. The incoming ray makes an incident angle θ with the normal ω_j to the reflecting surface. The ray after reflection passes a receiver Rx at a certain distance making an angle β.

autograd.

In general, the power from reflections depends not only on the total distance traveled but also on the angle at which the reflection occurs at the voxel v_j, and whether the reflected ray reaches the receiver (Rx). We model this behavior through θ and β, respectively, which parameterize a nonlinear function f. The incidence angle θ measures the angle between the Tx and the orientation ω_j, and β denotes the angle Rx makes with the reflected ray M (see Figure 5.6). Note that if $v_j \in \mathcal{V}$, and if ω_j is correct, $\beta = 0$. The reflected ray M can be computed as shown in Equation (5.11).

$$M = (v_j - Tx) - 2\left[(v_j - Tx).\omega_j\right]\omega_j \tag{5.11}$$

Here ω_j is a unit vector.

CHAPTER 6

RESULTS

We now present our experimental setting followed by the results and analysis.
Floor plan and Trajectory. Floorplans are drawn from the Zillow Indoor Dataset [20] composed of one-story homes. We also generate floorplans from realistic apartment layouts. In each floor plan, we use the A^* algorithm [50] to generate a walking trajectory that traverses all rooms. This mimics a user walking with a phone and collecting WiFi measurements. The rooms are devoid of furniture; however, we will later add a few toy shapes to mimic objects in the path of trajectories.
Wireless Simulation Dataset. We utilize the NVIDIA Sionna RT [55, 19], a differentiable ray tracer for radio propagation modeling, as the platform to compute the ground truth signal power (also known as *received signal strength index* (RSSI)). We randomly place M Txs, one in each room, denoted as T_m. To simulate omnidirectional transmissions at 2.4GHz from each Tx location, we shoot 10^7 rays into the given floorplan. For receiver locations, we sample the user trajectory at a fixed time interval to obtain N Rx locations, denoted as R_n. Only rays that reach the Rx location within 5 bounces (i.e., up to 5-th order reflections) contribute to the RSSI; others attenuate excessively and are negligible. Sionna accounts for specular reflections and refraction when these rays interact with walls in the specified floor plan; we use the default materials for the walls. As with most WiFi simulators, Sionna does not model signal penetration through walls – this means that a Tx and Rx located on opposite sides of a wall will not receive any RSSI. Overall, we gather $M \times N$ RSSI measurements $(T_m, R_n, \psi_{m,n})$ that serve as input to `NeRFMap`.

6.1 Baselines

Results are compared against the following baselines:

Table 6.1: Performance Results for Wall_IoU, F1 Score, and RPE

	2000 receiver locations			1000 receiver locations		
Method	Wall_IoU ↑	F1 Score ↑	RPE ↓	Wall_IoU ↑	F1 Score ↑	RPE ↓
`MLP`	-	-	1.03	-	-	0.65
`Heatmap Segmentation`	0.12 ± 0.03	0.21 ± 0.05	1.32	0.09 ± 0.02	0.16 ± 0.04	1.46
`NeRF2`	0.14 ± 0.02	0.24 ± 0.03	4.36	0.12 ± 0.02	0.21 ± 0.04	4.2
`NeRFMap_LoS`	0.27 ± 0.07	0.42 ± 0.10	9.12	0.25 ± 0.04	0.39 ± 0.06	10.86
`NeRFMap`	0.38 ± 0.06	0.55 ± 0.06	3.56	0.32 ± 0.06	0.48 ± 0.05	4.32

1. `NeRF2` [15]: Models WiFi reflections via virtual transmitters to predict the channel impulse response (CIR).

2. `Heatmap Segmentation` [61]: Interpolates CIR across the whole floorplan and applies an image segmentation algorithm (on the interpolated RSSI heatmap) to isolate each room. Essentially, the algorithm identifies the contours of sharp RSSI change since such contours are likely to correspond to walls. The raw trajectory signal power values are first interpolated to obtain a heatmap that provides a smoother representation of the input measurements. Of course, interpolating for regions without any data can lead to incorrect results, especially in larger unseen areas. A rule-based classifier is then applied for segmentation, using two criteria: (1) RSSI values above a threshold to identify potential room areas, and (2) smoothness of the RSSI signal, assessed through the second-order derivative, to ensure continuity within rooms. The initial segmentation is refined using morphological operations (via dilation and erosion) with a 3x3 kernel to smooth rough edges and eliminate small components. Overlapping regions are resolved by comparing gradient magnitudes, followed by additional morphological processing and connected component analysis to obtain the final, refined segmentation. Implementation details are included in the Appendix.

3. **MLP**: Trains a MLP network to directly estimate the RSSI based on Tx and Rx locations.

4. **NeRFMap_LoS**: Reports `NeRFMap`'s result considering only LoS path (ablation study).

Metrics. We evaluate using 3 metrics:

(A) Wall Intersection over Union (Wall_IoU): This metric measures the degree to which the predicted walls and the true walls superimpose over

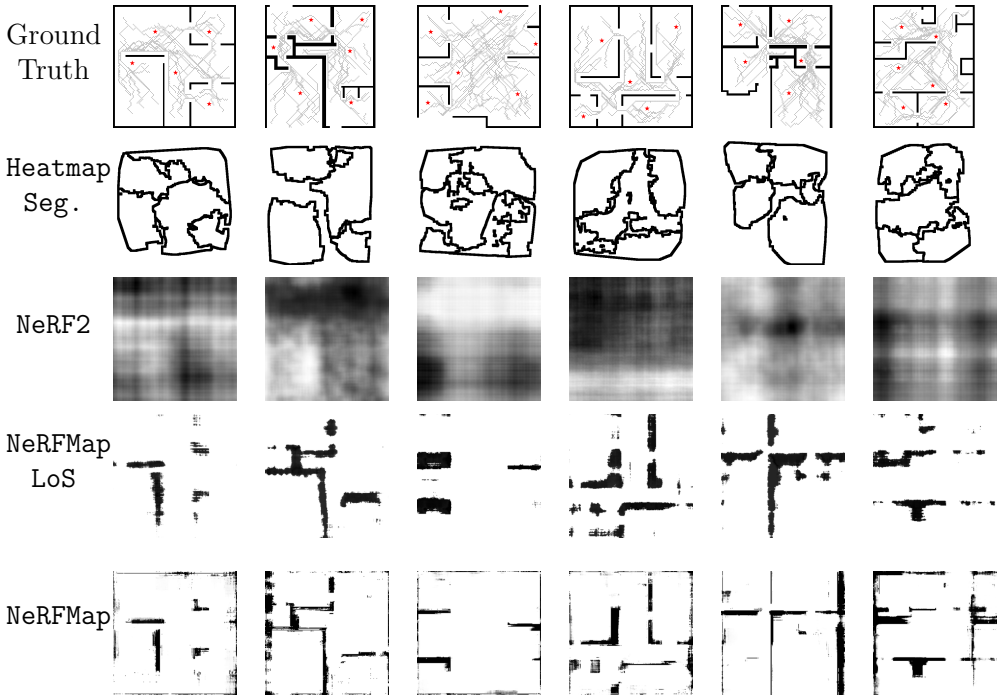

Figure 6.1: Qualitative comparison of Ground Truth floorplans against those inferred by baselines `Heatmap Segmentation` and `NeRF2`. The bottom two rows show floorplans by our proposed models `NeRFMap_LoS` and `NeRFMap` with clearly demarcated walls and boundaries.

each other in the 2D floorplan. The following equation defines the metric

$$\texttt{Wall\_IoU} = \frac{WP \cap WP^*}{WP \cup WP^*}$$

where WP denotes the set of predicted wall pixels and WP^* denotes the true wall pixels. This is a harsh metric given wall pixels are a small fraction of the total floorplan; if a predicted wall is even offset by one pixel from the true wall, the `Wall_IoU` drops significantly. IoU [56] has often been defined in terms of room pixels (instead of wall pixels); this is an overestimate in our opinion, since predicting even an empty floorplan results in an impressively high IoU.

(B) F1 score [57]: Defined as $F1 = \frac{2 \times P \times R}{P+R}$, where P is the *precision* and R is the *recall* of the bitmap. P and R are defined based on wall pixels, similar as above.

(C) RSSI Prediction Error (RPE): We split all Rx locations into a training and test set. RPE reports the average median RSSI error over all the test locations across floorplans.

6.2 Overall Summarized Results

Table 6.1 reports comparative results between NeRFMap and baselines, averaged over 20 different experiments, using all 3 metrics. With N=2000 Rx locations, the user is expected to walk for around 8 minutes, assuming her phone is sampling at 4Hz. We repeat the experiment for fewer measurements of N=1000, reducing the user's burden to 4 minutes of walking; this is sparse measurements given most floorplans are more than $250,000$ pixels. Mean and standard deviation are reported in the table. NeRFMap outperforms all models in terms of Wall_IoU and F1 Score. Compared to NeRFMap_LoS, NeRFMap demonstrates visible improvements, highlighting the advantage of modeling reflections. The absolute Wall_IoU values are understandably low because *the metric penalizes small errors.*

NeRF2 is unable to predict the floor plan (opaque voxels) well and is only able to achieve better RPE than NeRFMap_LoS. NeRFMap outperforms both NeRFMap_LoS and NeRF2. Interestingly, MLP incurs a lower RPE than NeRF2 suggesting that RSSI is amenable to interpolation, and NeRF2's implicit representation may not be an advantage for this interpolation task.

6.3 Qualitative Results

We report qualitative results, including visual floorplans, predicted RSSI heatmap, and ray-tracing visualization.

■ **Visual floorplans.** Figure 6.1 presents visualization from all baselines and a comparison with our LoS-only model (as ablation). All the floorplans use $N = 2000$ receiver locations. We make the following observations. (1) Heatmap Segmentation leverages the difference of RSSI on opposite sides of a wall, however, reflections pollute this pattern, especially at larger distances between Tx and Rx. Further, signals leak through open doors, injecting errors in the room boundaries. (2) NeRF2 performs poorly since its MLP learns one among many possible assignments of virtual transmitters to fit the RSSI training data. The virtual transmitters hardly correlate to the walls of the environment. (3) NeRFMap_LoS can infer the position of inner walls. However, these walls are thick and slanted because while NeRFMap_LoS can identify occlusions between a $\langle Tx, Rx \rangle$ pair, it cannot tell the shape

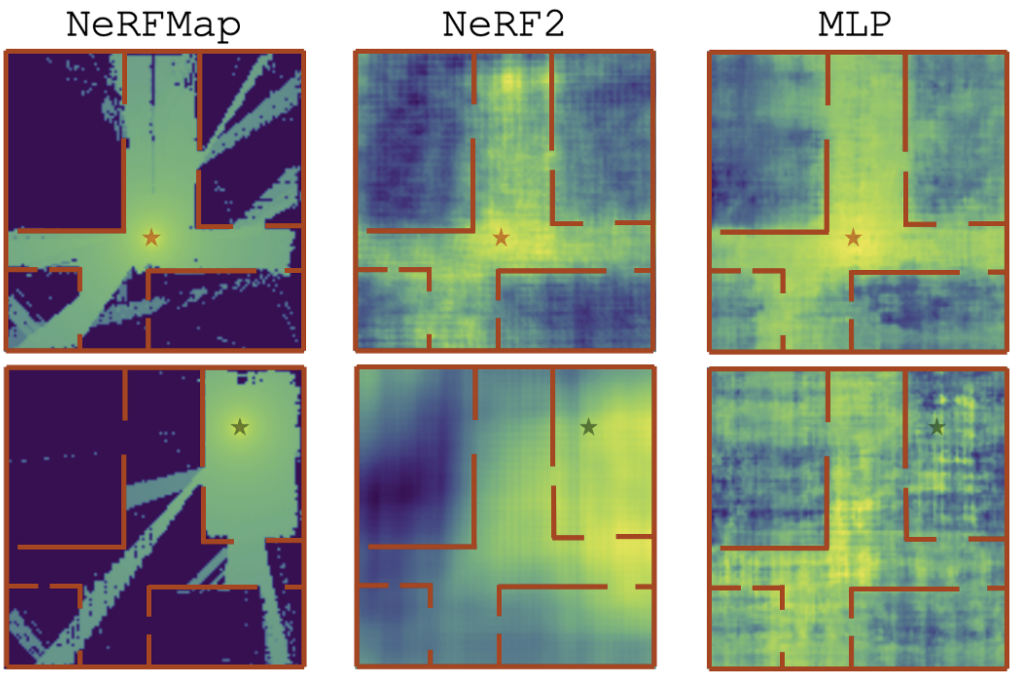

Figure 6.2: Heatmaps highlighting NeRFMap's ability to learn signal propagation. (Top row) Inferred RSSI heatmaps with Tx (red star) as used in training. (Bottom row) A new Tx (green star) degrades NeRF2 and MLP while NeRFMap shows accurate predictions.

and pattern of these occlusions. Crucially, NeRFMap_LoS also cannot infer the boundary walls since no receivers are located outside the house. (4) NeRFMap outperforms the baselines, sharpens the inner walls compared to NeRFMap_LoS, and constructs the boundary walls well.

Shortcomings: Recall that some parts of the floorplan are in the "blind spots" of our dataset since no reflection arrives from those parts to any of our sparse Rx locations (e.g., see bottom left corner of the 1^{st} floorplan; no signals reflect off this region to arrive at any of the Rx locations). Hence, NeRFMap is unable to construct the bottom of the left wall in this floorplan. Finally, note that areas outside the floorplan (e.g., the regions on the right of 6^{th} floorplan) cannot be estimated correctly since no measurements are available from those regions (hence, those voxels do not influence the gradients).

■ **RSSI prediction.** Figure 6.2 visualizes and compares predicted RSSI. The top row shows predictions at new Rx locations with the Tx held at the trained location; the bottom row shows predictions when both Tx and Rx are moved to new locations. Two key observations emerge: (1) NeRFMap is limited by Sionna's inability to simulate through-wall signal penetration;

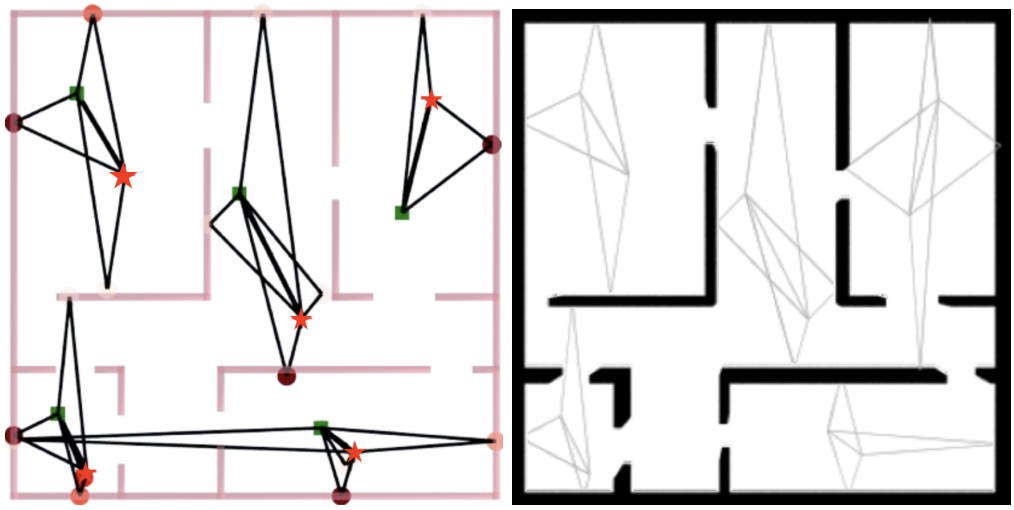

Figure 6.3: (a) Tracing reflections on the learnt floorplan. (b) Ground truth reflections from Sionna. Star indicates Tx

`NeRF2` has access to an expensive license for a through-wall simulation and shows better predictions inside the rooms. However, in areas that `NeRFMap` can "see" (e.g., corridors in the top row), the awareness of reflecting surfaces leads to significantly better predictions. (2) When the Tx location differs from that used in training, `NeRFMap`'s improvement over `NeRF2` is significant. This is the core advantage of first solving the inverse floor plan inference problem and then leveraging it for RSSI prediction.

■ **Learning reflected rays.** For a given $\langle Tx, Rx \rangle$ pair, we examine the points in the plausible set $\mathcal{V}$ that contribute to the reflections. Figure 6.3 compares the ray-tracing results from the NVIDIA Sionna simulator (we pick only first-order reflections). `NeRFMap` captures many of the correct reflections. Of course, some are incorrect – a false positive occurs in the bottom right room since some wall segment is missing in our estimate; false negatives also occur in the top right room where again some parts of the wall are missing.

6.4 Relaxing Assumptions & Sensitivity Study

■ **Transmitter's location.** `NeRFMap` assumes known Tx locations but we relax this assumption. We estimate the Tx location using maximum likelihood estimation from observed RSSI power, ψ^*. We describe the method in Section 5.7; on average, the estimated Tx location error is 2.08 pixels in floorplans of sizes $\approx 512 \times 512$ pixels.

Figure 6.4. visualizes the ground truth and the estimated Tx locations across 6 floorplans. The estimated Tx positions closely match the ground truth, and we report the Tx location error to be 2.08 pixels. Figure 6.5. demonstrates the performance of `NeRFMap_LoS` and `NeRFMap` using the estimated Tx locations for the 6 floorplans in Figure 6.4. Performance is comparable to that achieved with ground truth Tx locations, highlighting robustness.

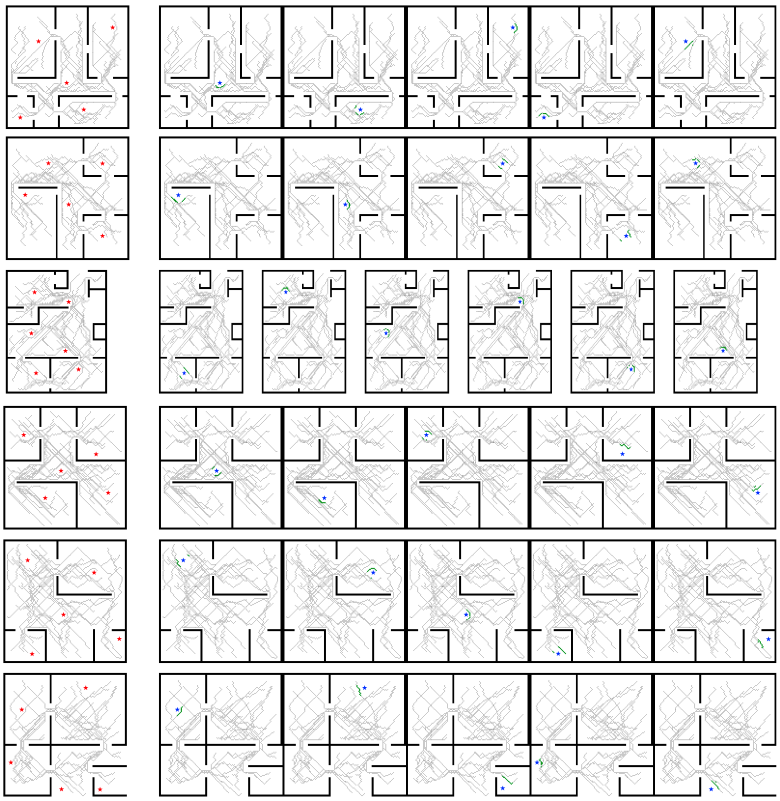

Figure 6.4: Comparison of Ground truth Tx locations indicated in red in the first column with the estimated Tx locations shown in blue from starting from column two. The Rx positions used for the estimation are marked in green.

■ **Receiver location error.** Indoor positioning systems are able to localize users but still incur some error. Table 6.2 shows `NeRFMap`'s sensitivity to this error. We inject Gaussian noise $\mathcal{N}(0, \sigma^2 I)$ to the Rx locations; $\sigma = 1$ implies a physical error of 1m. `Wall_IoU` accuracy obviously drops with error but 0.5 meter of error is tolerable without destroying the floorplan structure. Advancements in WiFi positioning systems are demonstrating robust sub-meter error.

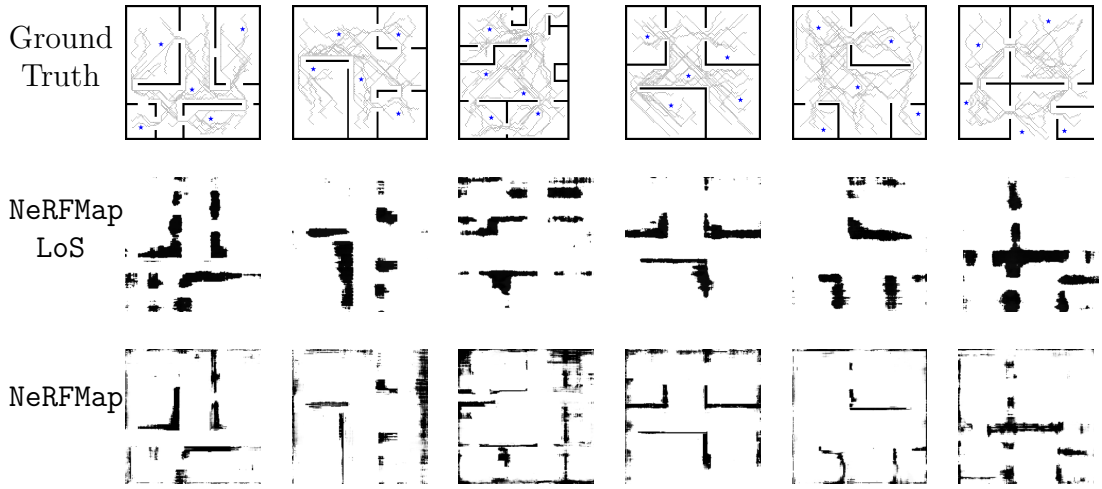

Figure 6.5: Qualitative comaprison of `NeRFMap_LoS` and `NeRFMap` when Tx locations are unknown, and are estimated. The top row shows the ground truth floorplan, Rx locations along with the estimated Txs in blue. The second and third row displays the performance of `NeRFMap_LoS` and `NeRFMap` respectively. Despite the Tx locations being unknown, our methods accurately estimate them, leading to performance comparable to the case where Tx locations are known.

Table 6.2: Estimated Wall_IoU at various levels of injected noise σ

Error σ(m)	0	0.5	1	2	3
`Wall_IoU`	0.38	0.35	0.33	0.29	0.26

■ **Effect of Furniture.** Figure 6.6 visualizes inferred floorplans when toy objects are scattered in open spaces (Rx locations remain N=2000). `NeRFMap` is able to identify some of the object blobs but sharpening the small objects is challenging, due to more higher-order reflections from furniture. Follow-up work is needed, either in modeling 2^{nd} order reflections or by imposing stronger regularizations.

Follow-ups. (1) The Sionna simulator does not model through-wall signal penetration; hence, we have placed a Tx in each room. In reality (or in expensive ray tracing simulators [62]), WiFi signals will penetrate walls. This is an advantage since signals from a single Tx will be measurable across the whole home. However, voxel opacities will no longer be bimodal (0 or 1), hence `NeRFMap` will need to assign $\delta \in [0, 1]$ to match the measured RSSI. This will require modifications to our models. (2) The ability to model 2^{nd} or-

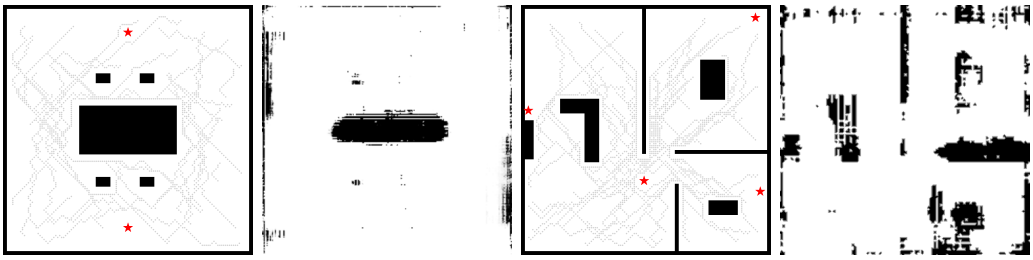

Figure 6.6: `NeRFMap`'s floorplan inference with furniture present in a typical conference room (left) and apartment (right) layout.

der reflections will boost `NeRFMap`'s accuracy, allowing it to sharpen the scene and decode smaller objects. Even for short range applications, such as non-intrusive medical imaging, such 2^{nd} and 3^{rd} order reflections would be helpful. This remains an open problem for the community. (3) Extending `NeRFMap` to 3D floorplans is also of interest, and since it is undesirable to increase the number of measurements, effective 3D priors, or 2D-to-3D post-processing, may be necessary. Such post-processing tools exist [63], but we have not applied them since our goal is to improve NeRF's inherent capability. (4) Fusing additional signal modalities, such as audio (from music loudspeakers), or overhearing from many WiFi enabled devices at home, could lead to increased measurement density. (5) Finally, `NeRFMap` floorplans can offer valuable spatial context to Neural RIR synthesizers like [15, 16, 25, 18, 26]. Synthesized RIR could in-turn aid `NeRFMap`'s floorplan inference, forming the basis for an alternating optimization strategy. We leave these ideas to follow-up research.

CHAPTER 7

CONCLUSION

In this thesis, we present two complementary approaches for inferring indoor floorplans using wireless signal measurements. First, we introduced `PhyMap`, a data-driven method that leverages a learned physical model to map RF signal strengths to plausible floorplan representations. At the heart of `PhyMap` is a physics-based model `PhyModel` that implicitly learns a continuous signal field over the environment, capturing key propagation effects such as line-of-sight attenuation and reflections. This physical model serves as a differentiable guide to refine the floorplan during training and facilitates test-time optimization, enabling improved inference through gradient-based refinement.

Next, we propose `NeRFMap`, a Neural Radiance Fields (NeRF)-inspired approach that explicitly models signal propagation to tackle the inverse problem of floorplan inference. By adapting the classical NeRF framework to RF signals, `NeRFMap` captures multipath propagation, including line-of-sight and all first-order reflections—accounts for most signal energy, by parameterizing each voxel with opacity and orientation. The key idea involves teaching the NeRF that reflections originate from a manifold, and then using a two-stage optimization to mitigate gradient issues arising from line-of-sight dominance. Our results demonstrate that `NeRFMap` successfully infers accurate floorplans, outperforming baselines that predict the signal power without solving the inverse problem.

Both `PhyMap` and `NeRFMap` exhibit additional capabilities beyond floorplan inference, including transmitter localization, signal field prediction, and basic ray tracing. Looking ahead, we envision a unified framework that integrates both paradigms—combining the data generalization strengths of `PhyMap` with the signal modeling capability and interpretability of `NeRFMap`. We believe such a hybrid model can pave the way for robust and generalizable RF-based scene understanding in diverse real-world environments.

REFERENCES

[1] C.-Y. Lee, V. Badrinarayanan, T. Malisiewicz, and A. Rabinovich, "Roomnet: End-to-end room layout estimation," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 4865–4874.

[2] C. Zou, A. Colburn, Q. Shan, and D. Hoiem, "Layoutnet: Reconstructing the 3d room layout from a single rgb image," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2051–2059.

[3] C. Yan, B. Shao, H. Zhao, R. Ning, Y. Zhang, and F. Xu, "3d room layout estimation from a single rgb image," *IEEE Transactions on Multimedia*, vol. 22, no. 11, pp. 3014–3024, 2020.

[4] H. Jia, H. Yi, H. Fujiki, H. Zhang, W. Wang, and M. Odamaki, "3d room layout recovery generalizing across manhattan and non-manhattan worlds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 5192–5201.

[5] W. Zhang, W. Zhang, and J. Gu, "Edge-semantic learning strategy for layout estimation in indoor environment," *IEEE transactions on cybernetics*, vol. 50, no. 6, pp. 2730–2739, 2019.

[6] S. Stekovic, S. Hampali, M. Rad, S. D. Sarkar, F. Fraundorfer, and V. Lepetit, "General 3d room layout from a single view by render-and-compare," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVI 16*. Springer, 2020, pp. 187–203.

[7] L. Ericson and P. Jensfelt, "Beyond the frontier: Predicting unseen walls from occupancy grids by learning from floor plans," *IEEE Robotics and Automation Letters*, 2024.

[8] J. Xiao and Y. Furukawa, "Reconstructing the world's museums," *International journal of computer vision*, vol. 110, pp. 243–258, 2014.

[9] 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," *Communications of the ACM*, vol. 65, no. 1, pp. 99–106, 2021.

[10] Z. Wang, T. Shen, J. Gao, S. Huang, J. Munkberg, J. Hasselgren, Z. Gojcic, W. Chen, and S. Fidler, "Neural fields meet explicit geometric representation for inverse rendering of urban scenes," 2023. [Online]. Available: https://arxiv.org/abs/2304.03266

[11] 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," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 5855–5864.

[12] A. Yu, V. Ye, M. Tancik, and A. Kanazawa, "pixelnerf: Neural radiance fields from one or few images," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 4578–4587.

[13] A. Basu and A. Chakraborty, "Specnerf: Neural radiance field driven wireless coverage mapping for 5g networks," in *Proceedings of the Twenty-Fifth International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing*, ser. MobiHoc '24. New York, NY, USA: Association for Computing Machinery, 2024. [Online]. Available: https://doi.org/10.1145/3641512.3690037 p. 440–445.

[14] T. Orekondy, P. Kumar, S. Kadambi, H. Ye, J. Soriaga, and A. Behboodi, "WineRT: Towards neural ray tracing for wireless channel modelling and differentiable simulations," in *The Eleventh International Conference on Learning Representations*, 2023. [Online]. Available: https://openreview.net/forum?id=tPKKXeW33YU

[15] X. Zhao, Z. An, Q. Pan, and L. Yang, "Nerf2: Neural radio-frequency radiance fields," in *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*, 2023, pp. 1–15.

[16] H. Lu, C. Vattheuer, B. Mirzasoleiman, and O. Abari, "Newrf: A deep learning framework for wireless radiation field reconstruction and channel prediction," in *Forty-first International Conference on Machine Learning*, 2024.

[17] A. Luo, Y. Du, M. Tarr, J. Tenenbaum, A. Torralba, and C. Gan, "Learning neural acoustic fields," *Advances in Neural Information Processing Systems*, vol. 35, pp. 3165–3177, 2022.

[18] S. Liang, C. Huang, Y. Tian, A. Kumar, and C. Xu, "Av-nerf: Learning neural fields for real-world audio-visual scene synthesis," *Advances in Neural Information Processing Systems*, vol. 36, pp. 37 472–37 490, 2023.

[19] J. Hoydis, F. A. Aoudia, S. Cammerer, M. Nimier-David, N. Binder, G. Marcus, and A. Keller, "Sionna rt: Differentiable ray tracing for radio propagation modeling," in *2023 IEEE Globecom Workshops (GC Wkshps)*. IEEE, 2023, pp. 317–321.

[20] S. Cruz, W. Hutchcroft, Y. Li, N. Khosravan, I. Boyadzhiev, and S. B. Kang, "Zillow indoor dataset: Annotated floor plans with 360deg panoramas and 3d room layouts," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 2133–2143.

[21] P. Prandoni and M. Vetterli, *Signal Processing for Communications*. Boca Raton, FL: CRC Press, 2008.

[22] N. Max, "Optical models for direct volume rendering," *IEEE Transactions on Visualization and Computer Graphics*, vol. 1, no. 2, pp. 99–108, 1995.

[23] D. Tse and P. Viswanath, *Fundamentals of Wireless Communication*. Cambridge University Press, 2005.

[24] K. Su, M. Chen, and E. Shlizerman, "Inras: Implicit neural representation for audio scenes," *Advances in Neural Information Processing Systems*, vol. 35, pp. 8144–8158, 2022.

[25] A. Brunetto, S. Hornauer, and F. Moutarde, "Neraf: 3d scene infused neural radiance and acoustic fields," *arXiv preprint arXiv:2405.18213*, 2024.

[26] S. Liang, C. Huang, Y. Tian, A. Kumar, and C. Xu, "Neural acoustic context field: Rendering realistic room impulse response with neural fields," *arXiv preprint arXiv:2309.15977*, 2023.

[27] S. Majumder, C. Chen, Z. Al-Halah, and K. Grauman, "Few-shot audio-visual learning of environment acoustics," *Advances in Neural Information Processing Systems*, vol. 35, pp. 2522–2536, 2022.

[28] Z. Oufqir, A. El Abderrahmani, and K. Satori, "Arkit and arcore in serve to augmented reality," in *2020 International Conference on Intelligent Systems and Computer Vision (ISCV)*. IEEE, 2020, pp. 1–7.

[29] W. Ge, T. Hu, H. Zhao, S. Liu, and Y.-C. Chen, "Ref-neus: Ambiguity-reduced neural implicit surface learning for multi-view reconstruction with reflection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 4251–4260.

[30] Z. Yan, C. Li, and G. H. Lee, "Nerf-ds: Neural radiance fields for dynamic specular objects," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 8285–8295.

[31] S. Seo, Y. Chang, and N. Kwak, "Flipnerf: Flipped reflection rays for few-shot novel view synthesis," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 22 883–22 893.

[32] V. Rudnev, M. Elgharib, W. Smith, L. Liu, V. Golyanik, and C. Theobalt, "Nerf for outdoor scene relighting," in *European Conference on Computer Vision*. Springer, 2022, pp. 615–631.

[33] M. Toschi, R. De Matteo, R. Spezialetti, D. De Gregorio, L. Di Stefano, and S. Salti, "Relight my nerf: A dataset for novel view synthesis and relighting of real world objects," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 20 762–20 772.

[34] Y.-C. Guo, D. Kang, L. Bao, Y. He, and S.-H. Zhang, "Nerfren: Neural radiance fields with reflections," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 18 409–18 418.

[35] D. Verbin, P. Hedman, B. Mildenhall, T. Zickler, J. T. Barron, and P. P. Srinivasan, "Ref-NeRF: Structured view-dependent appearance for neural radiance fields," *CVPR*, 2022.

[36] M. Boss, R. Braun, V. Jampani, J. T. Barron, C. Liu, and H. Lensch, "Nerd: Neural reflectance decomposition from image collections," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 12 684–12 694.

[37] X. Zhang, P. P. Srinivasan, B. Deng, P. Debevec, W. T. Freeman, and J. T. Barron, "Nerfactor: Neural factorization of shape and reflectance under an unknown illumination," *ACM Transactions on Graphics (ToG)*, vol. 40, no. 6, pp. 1–18, 2021.

[38] V. Hedau, D. Hoiem, and D. Forsyth, "Recovering the spatial layout of cluttered rooms," in *2009 IEEE 12th international conference on computer vision*. IEEE, 2009, pp. 1849–1856.

[39] A. Inc., "Arkit: Augmented reality for ios," 2017, accessed: 2025-02-17. [Online]. Available: https://developer.apple.com/arkit/

[40] A. G. Schwing, T. Hazan, M. Pollefeys, and R. Urtasun, "Efficient structured prediction for 3d indoor scene understanding," in *2012 IEEE conference on computer vision and pattern recognition*. IEEE, 2012, pp. 2815–2822.

[41] H. Shin, Y. Chon, and H. Cha, "Unsupervised construction of an indoor floor plan using a smartphone," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 6, pp. 889–898, 2011.

[42] L. Del Pero, J. Bowdish, D. Fried, B. Kermgard, E. Hartley, and K. Barnard, "Bayesian geometric modeling of indoor scenes," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2012, pp. 2719–2726.

[43] B. Okorn, X. Xiong, B. Akinci, and D. Huber, "Toward automated modeling of floor plans," in *Proceedings of the symposium on 3D data processing, visualization and transmission*, vol. 2, 2010.

[44] C. Liu, J. Wu, and Y. Furukawa, "Floornet: A unified framework for floorplan reconstruction from 3d scans," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 201–217.

[45] A. Van Den Oord, O. Vinyals et al., "Neural discrete representation learning," *Advances in neural information processing systems*, vol. 30, 2017.

[46] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *International conference on machine learning*. PMLR, 2017, pp. 214–223.

[47] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[48] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[49] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," in *International Conference on Learning Representations*, 2017. [Online]. Available: https://openreview.net/forum?id=rkE3y85ee

[50] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[51] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017. [Online]. Available: https://arxiv.org/abs/1412.6980

[52] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *arXiv preprint arXiv:1308.3432*, 2013.

[53] L. Tingguang, H. Danny, L. Chenming, Z. Delong, W. Chaoqun, and M. Q.-H. Meng, "Houseexpo: A large-scale 2d indoor layout dataset for learning-based algorithms on mobile robots," *arXiv preprint arXiv:1903.09845*, 2019.

[54] S. Song, F. Yu, A. Zeng, A. X. Chang, M. Savva, and T. Funkhouser, "Semantic scene completion from a single depth image," *Proceedings of 30th IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[55] J. Hoydis, S. Cammerer, F. A. Aoudia, A. Vem, N. Binder, G. Marcus, and A. Keller, "Sionna: An open-source library for next-generation physical layer research," *arXiv preprint arXiv:2203.11854*, 2022.

[56] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese, "Generalized intersection over union: A metric and a loss for bounding box regression," 2019. [Online]. Available: https://arxiv.org/abs/1902.09630

[57] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Information Processing & Management*, vol. 45, no. 4, pp. 427–437, 2009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0306457309000259

[58] R. Ranftl, A. Bochkovskiy, and V. Koltun, "Vision transformers for dense prediction," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 12 179–12 188.

[59] C. A. Balanis, *Antenna Theory: Analysis and Design*, 3rd ed. John Wiley & Sons, 2016. [Online]. Available: https://www.wiley.com/en-us/Antenna+Theory+and+Design%2C+3rd+Edition-p-9780470576649

[60] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D: Nonlinear Phenomena*, vol. 60, no. 1, pp. 259–268, 1992. [Online]. Available: https://www.sciencedirect.com/science/article/pii/016727899290242F

[61] D. Liu, B. Soran, G. Petrie, and L. Shapiro, "A review of computer vision segmentation algorithms," *Lecture notes*, vol. 53, 2012.

[62] R. Inc., "Wireless insite® 3d wireless propagation software," 2024, accessed: 2025-01-03. [Online]. Available: https://www.remcom.com/wireless-insite-propagation-software

[63] Cedreo, "Convert 2d floor plan to 3d," 2025, accessed: 2025-03-01. [Online]. Available: https://cedreo.com/floor-plans/convert-2d-floor-plan-to-3d/

APPENDIX A

PHYMODEL

A.1 Architectural Variants of `PhyModel`

To address the limitations of `PhyModel` discussed in Section 4.8, we investigated alternative architectures with the aim of improving its ability to guide floorplan inference.

A.1.1 Two-Component Signal Decomposition

The first approach involved decomposing RSSI prediction into two stages: one to model line-of-sight (LoS) signals and the other to model reflections. We trained two independent ResNet50 models for each component and combined their outputs to estimate the final RSSI values. As shown in Figure A.1, this approach reduced the overall prediction error. However, it requires ground truth separation of LoS and reflected components, which is impractical in real-world settings.

A.1.2 Dense Prediction Transformer

We also evaluated a Dense Prediction Transformer (DPT) [58] to model RSSI fields. The model was trained with the same loss function and dataset used for `PhyModel`, but no architecture-specific adjustments were made from the original DPT. As shown in Figure A.2, the DPT model achieved an MSE of 7.31 dB in RSSI prediction. However, this model did not translate into better performance for floorplan reconstruction when integrated into `PhyMap`.

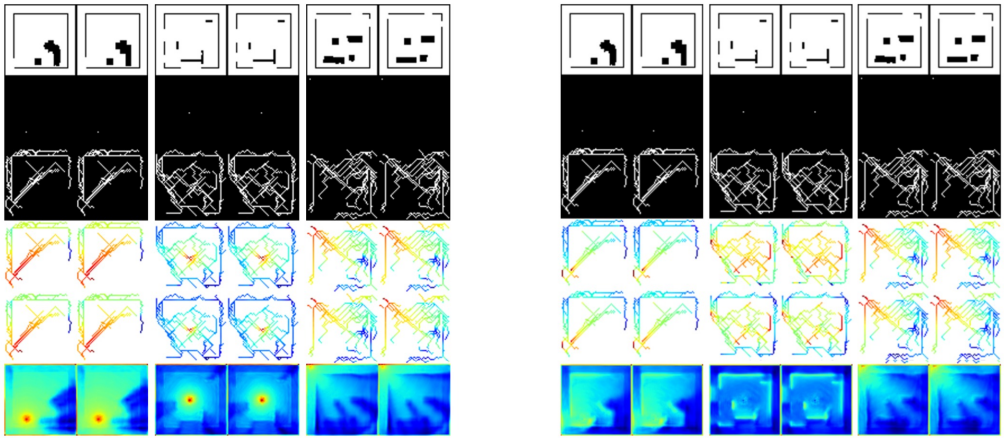

Figure A.1: RSSI prediction using a dual-ResNet model separating line-of-sight and reflected signal components. Left set denotes the line-of-sight model, while the right set denotes the reflected signal model. First column in every image pair shows performanace with VQ-VAE reconstructed floorplans, while the second column shows performance with ground truth floorplans.

A.2 Additional Attempts

We also experimented with injecting contrastive samples during training to improve robustness. However, these efforts did not lead to measurable improvements in the ability of `PhyModel` to distinguish between correct and incorrect floorplans. These findings suggest that architectural complexity alone may not be insufficient to overcome the limitations of current data-driven models in capturing signal propagation.

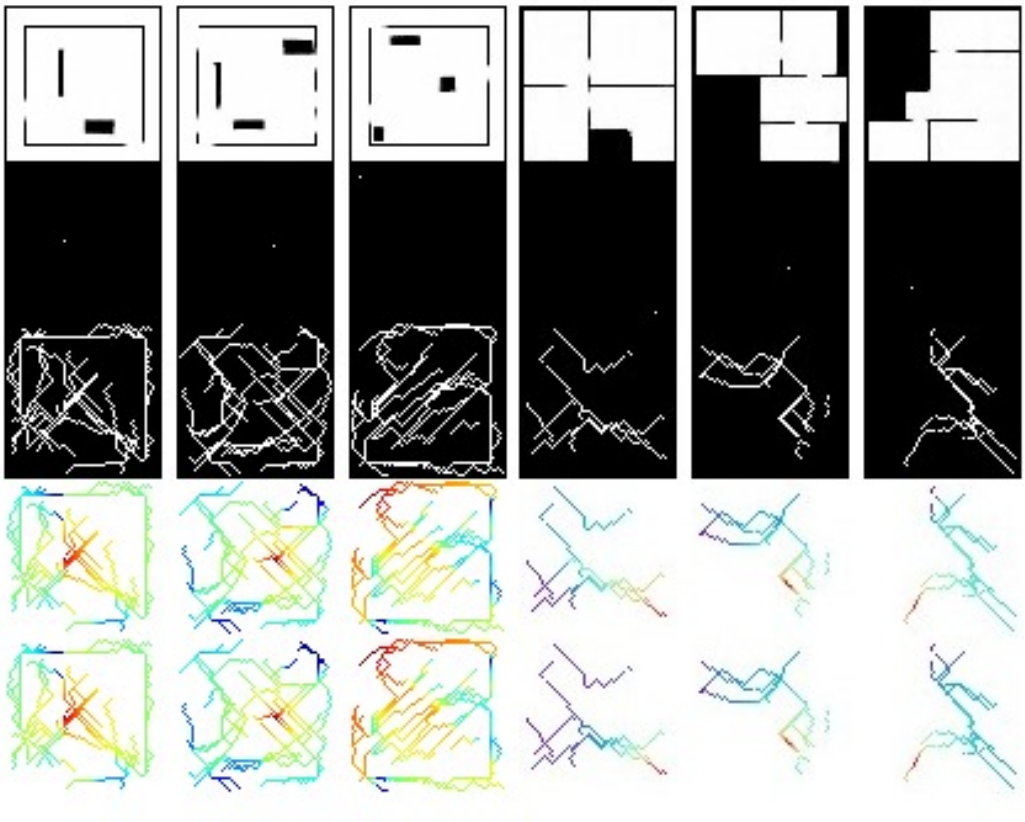

Figure A.2: RSSI predictions using a Dense Prediction Transformer. Top row is the VQ-VAE reconstructed flooplans, second row is the transmitter, third row is the receiver binary mask, and the last two rows show the ground truth and predicted RSSI results.

APPENDIX B

NERFMAP

B.1 Evaluation on Additional Floorplans

We evaluate NeRFMap on additional floorplans to demonstrate its generalization capabilities. Figure B.1 shows the performanace on realistic apartment layouts. We note that while NeRF2 is unable to predcit any reasonable floorplan, Heatmap Segmentationś shape is limited by the (convex hull of the) trajectory data (see bottom left of the second row, first column). Additionally, it fails to capture critical details, such as door openings. The fourth and fifth rows show floorplans by our proposed models NeRFMap_LoS and NeRFMap. NeRFMap_LoS captures the rough shape of the floorplan, especially the interior walls, while NeRFMap further improves these walls by adjusting their thickness and accurately correcting their shape. NeRFMap also correctly identifies the floorplan boundary, as evidenced in the last column, where the exact boundary is captured just from the reflections. To understand the signal propagation captured by NeRFMap, we place one Tx in each floor plan randomly (that is not present in the training data) and evaluate the signal power at discrete receivers. These Rxs are placed on a 2D grid at equal intervals and the predicted signal power is converted into a heatmap. The bottom row shows these inferred signal power heatmaps with the brightest point indicating the Tx location (as the Rx closest to the Tx receives the highest power). NeRFMap is not only able to predict the signals well across the floorplan, but also capture the propagation paths i.e., LoS signal and the first-order reflections. For instance, in the first column, the left portion of the center hall receives power only due to the wall reflection from the left wall.

Figure B.2 shows the performanace on ZIND floorplans. The fourth and fifth rows show floorplans by our proposed models `NeRFMap_LoS` and `NeRFMap` with clearly identified walls and boundaries. The bottom row shows inferred signal power heatmaps demonstrating `NeRFMap`'s capability to learn accurate signal propagation.

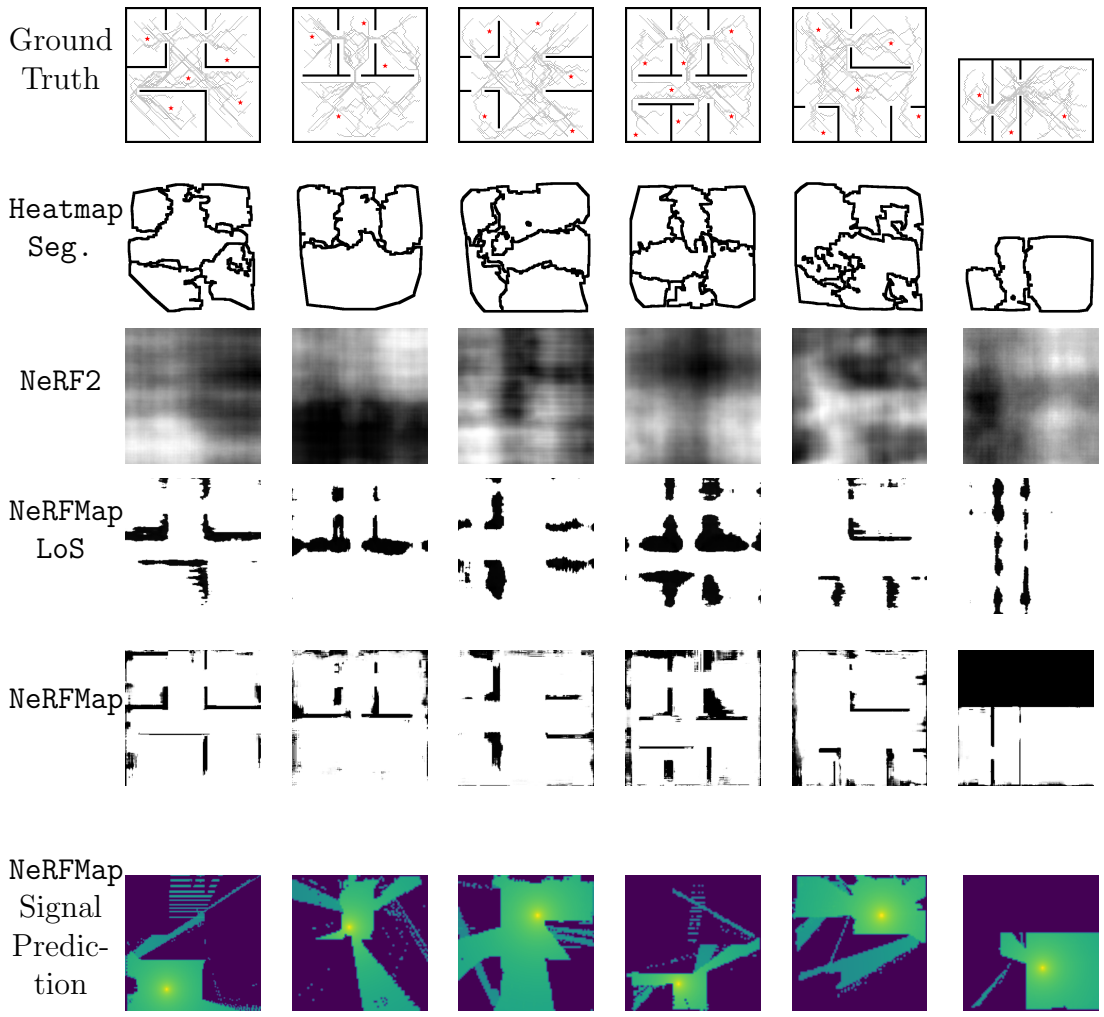

Figure B.1: Qualitative comparison of Ground Truth floorplans against those inferred by baselines `Heatmap Segmentation` and `NeRF2`

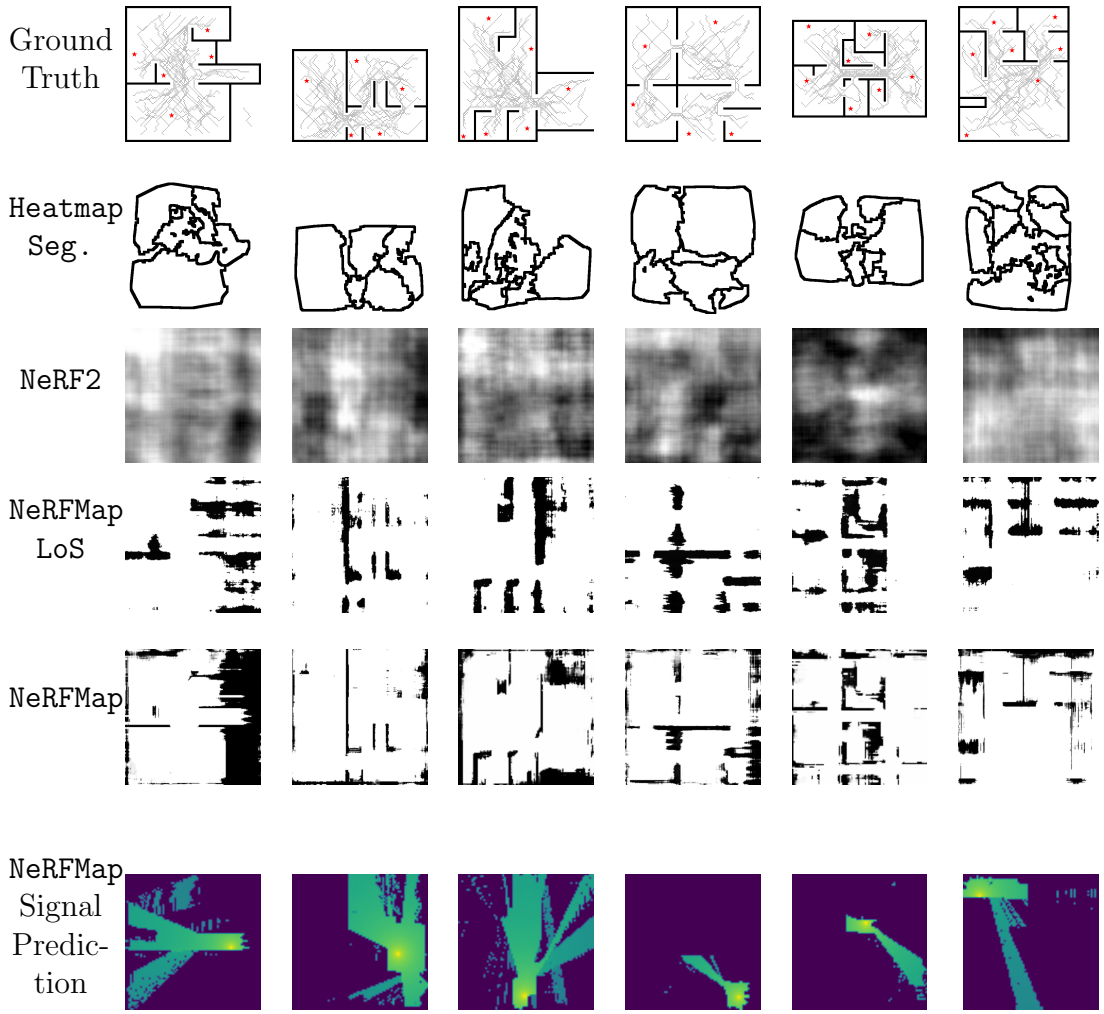

Figure B.2: Additional qualitative comparisons of Ground Truth floorplans against those inferred by baselines `Heatmap Segmentation` and `NeRF2`