# An Attention-based Neural Network Architecture for Recovering Wi-Fi Maps from Sparse Point Cloud Measurements

Gokhan Egri
gegri@g.harvard.edu

## Abstract

*Wi-Fi modeling is an essential component in ensuring Wi-Fi stability and has thus attracted significant attention over the years. In this work, we propose an attention-based neural network architecture which maps Wi-Fi RSS in indoor environments up to and including second-reflections using point-cloud samples. We find that while our method is capable of converging on viable approximations to the Wi-Fi RSS signal, our limited training data prohibits us from matching performance in unseen room structures. We conclude by noting two possible future directions for the related task of recovering room structure solely by sparse RSS measurements.*

## 1. Introduction

Wi-Fi is one of the most ubiquitous phenomena of modern life. This makes it all the more surprising that it is not subject to the same stability of performance we have come to expect from other fundamental utilities such as electric and water. An important precursory step to ensuring this stability lies in accurate and accessible modeling of Wi-Fi.

Earlier approaches to Wi-Fi modeling rely on the idea of simulation, namely ray-tracing based approaches where the received signal strength in an environment is accurately estimated upon being provided the room structure [3]. While by far the most accurate of the proposed methods, except of course arduous manual measurement, this method fails in accessibility as it not only expects domain-knowledge from end-users but also requires the environment to be mapped prior to modeling.

More accessible approaches to this problem include propagation models where engineered approximations allow closed-form heuristic solutions based on a limited number of measurements [6, 2]. These solutions however appeal to the opposite end of the spectrum and provide sub-par performance in exchange for ease of use.

Recently, there have been machine learning-based approaches to Wi-Fi modeling using Convolutional Neural Networks [4, 8]. While this approach makes progress towards an accessible and accurate Wi-Fi modeling system, it still requires a 2D model of the room.

In this work, we propose a neural network-based method which recovers an indoor Wi-Fi RSS map up to second reflections using sparse point cloud data which is easily collectable by smart phone cameras.

## 2. Related Work

**Scene Representation Networks.** Scene Representation Networks (SRN) [7] represent 3D scenes in terms of a function $\phi$ which maps from three-dimensional coordinates $(x, y, z)$ to scene properties such as opacity or color. SRNs implement this function in terms of a multi-layer perceptron which results in an implicit representation of the scene in the weights of the neural network. This formulation shows benefits over conventional 3D representations such as point clouds, voxels or meshes as it is continuous over $R^3$, can be sampled at arbitrary resolutions and take up substantially less storage space than alternatives.

**Multi-Head Attention.** Vaswani *et al*. [10] uses a variant of Bahdanau *et al*.'s *dot-product attention* [1], called *scaled dot-product attention*, formulated as

$$Att(Q, K, V) = softmax(\frac{Q^T K}{\sqrt{d}})V \tag{1}$$

where the attention layer outputs a weighted combination of the values of the input dictionary $(key, value)$ pairs based on the proximity of the $query$ to the corresponding $key$.

In the Transformer, attention is used in both the Encoder Block and the Decoder Block. In the former, attention is used to obtain learned context representations from the input sentence and in the latter, attention is used to select an aggregate of the context vectors generated by the Encoder that are relevant for the current step in the Decoder.

Vaswani *et al*. utilizes a further extension to the attention mechanism called *multi-head attention* which trains multiple attention networks on the same input such that some of the networks learn the latent, long-range contexts and some of them learn the immediate short-range contexts of the input which are then aggregated into a final context vector by the means of averaging or concatenation.

**Fourier Features.** Due to their spectral bias towards low-frequency content, neural networks have difficulty learning high frequency functions [5]. Fourier features [9] address this difficulty by incorporating synthetic high-frequency content into the input in the form of sinusoidal perturbations and are therefore significant auxiliary components to SRNs.

## 3. Methodology

**Problem Statement.** Given the line-of-sight ($H_0$), first reflection ($H_1$) and second reflection ($H_2$) received signal strength measurements at $M$ points, we want to find a function $\phi$ that maps $p = (x, y)$ from $R^2$ to received signal strengths $(H_0(p), H_1(p), H_2(p))$.

**Input.** The network takes as input $M$ 2D points $p_{[M\times 2]}$ corresponding to points where the ground truth $H_0(p)$, $H_1(p)$ and $H_2(p)$ values are known. The network also takes as input the router position $W_{[1\times 2]}$ and the $N$ sampled point cloud points $PC_{[N\times 2]}$.

**Output.** For each query point in $p$, the attention network outputs a $d$-dimensional embedding which has information regarding the relative position of $p_i$ with respect to the point cloud samples $PC$. The MLP network then maps this embedding vector to a single value corresponding to $H_j$ where $j \in \{0, 1, 2\}$. This is repeated for all three networks for $H_0$, $H_1$ and $H_2$.

### 3.1. 2D Room Camera Simulator

For synthetic data generation, we have first implemented a 2D room camera simulator in MATLAB with the following features:

- **Wall Interpolation from 2D Room Plan Images.** The simulator has the ability to take as input a black-on-white room structure diagram (see Figure 1) and then use Hough transform to locate and interpolate the line segments corresponding to the walls which allows the user to sample point clouds and other features later in the pipeline.
- **Point Cloud Sampling on Interpolated Walls.** Having interpolated the line segments corresponding to the wall segments in the input diagram, the simulator is then able to sample the wall segments for point clouds at a user-specified density.
- **Router Sampling.** The simulator is able to query the user for a user-specified number of router positions.
- **Depth Map Acquisition.** The simulator queries the user for multiple desired camera locations and orientations as well as custom camera intrinsics and generates 1D RGBD images from the placed cameras by means of 2D ray casting (see Figure 1).

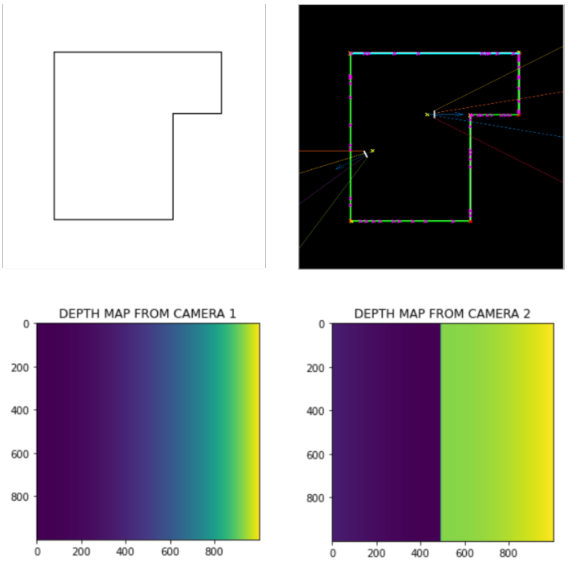


Figure 1. 2D Camera Simulator outputs. *(top left)* Input room diagram. *(top right)* Simulator output with cameras and point cloud samples. *(bottom)* 1D depth maps from Cameras 1 and 2

For measuring the received signal strength based on room and router combinations, we modify the publicly available Indoor Radio Propagation MATLAB model[1] to output the line-of-sight ($H_0$), first reflection ($H_1$) and second reflection ($H_2$) received signal strength maps which are then sampled at the query points output by the 2D Room Camera Simulator from before.

### 3.2. Wi-Fi Attention Network

We implement an Attention-based Neural Network architecture as seen in Figure 3. For each training instance, each network block (note from Figure 3 that the proposed architecture is replicated for each $H_i$ estimate) takes as input a query point $p$, a router position $W$, $M$ wall point-cloud samples $PC$ and outputs a single received signal strength estimate $\hat{H}_i$.

The first block in each network instance implements a multi-head attention mechanism which takes the query point $Q_{[1\times 2]}$ concatenated with the router position $W_{[1\times 2]}$ as the Query $(Q)$ and the $M$ point cloud array $PC_{[M\times 2]}$ as the Key $(K)$ and Value $(V)$. As in Vaswani *et al.*, these inputs are first linearly projected onto embeddings of dimension $d$ using learned projection matrices $W_{\{Q,K,V\}[d\times 2]}$ which are used to calculate a scaled-dot product attention. The block finally outputs a single vector $V_{Q[1\times d]}$ for each query point $Q$.

[1] https://www.mathworks.com/matlabcentral/fileexchange/64695-3d-ray-tracing-for-indoor-radio-propagation

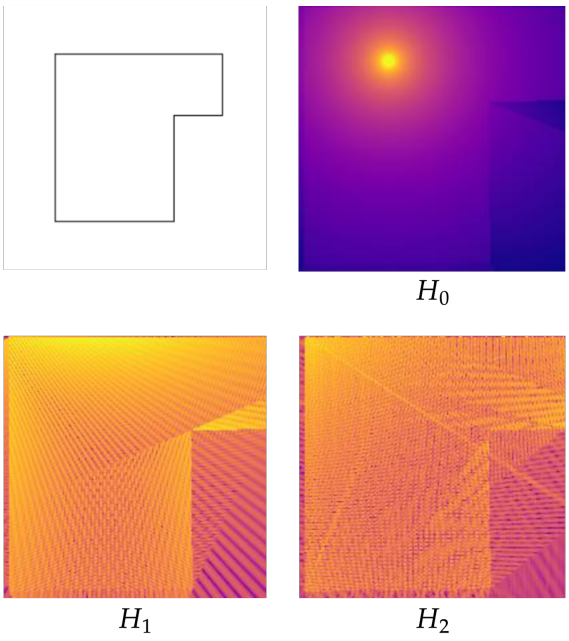


Figure 2. Wi-Fi Simulation Results up to and including second-reflections.

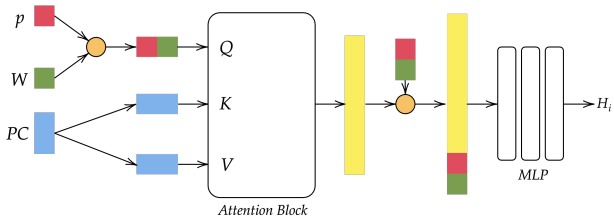


Figure 3. Wi-Fi Attention Network diagram. Note that this is replicated for $H_0$, $H_1$ and $H_2$.

The second block takes as input the vector $V_{Q[1\times d]}$ (we also experiment with concatenating the vector $V_{Q[1\times 2]}$ with the original Query $(Q)$ vector) and uses an 8-layer MLP which outputs a scalar $\hat{H}_i$.

We have also experimented with the following additional features:

- **Centering data at the router position:** As prior domain knowledge shows that the received signal strength is related to the relative position of each query point with respect to the router position, we can center all query points $p$ and point cloud points $PC$ at the corresponding router position $W$. This also enables us to remove the vestigial router position $W$ from the input.

- **Inputing the log-distance:** Based on the free-space path loss model as

$$FSPL(dB) = 20\log_{10}(d) + 20\log_{10}(f) - 147.55 \tag{2}$$

we also calculate the log-distances between query points $p$ and the router $W$ as $d_{log} = \log_{10}(||p - W||)$ and append the result to the input vector as $u^* = [p|W|d_{log}]$.

- **Skip connections:** In order to ensure that the MLP-part of the architecture has access to the original $p$ and $W$ values, we concatenate these values to the output of the attention block and input the resulting vector into the MLP.

- **Fourier Features:** In order to ensure that the network is able to learn the high-frequency variations between points in the case of walls and reflections, we implement two types of Fourier Features.

  *Linear Fourier Features.* These work as in [9] and take the $u = [p|W]$ input to the attention network and map it to

$$\begin{aligned}\phi(u) = [&a_1\cos(2\pi b_1^T u), a_1\sin(2\pi b_1^T u),\\ &\cdots a_m\cos(2\pi b_m^T u), a_m\sin(2\pi b_m^T u)]\end{aligned} \tag{3}$$

  *Radial Fourier Features.* As prior domain knowledge states that radio propagation is in essence a radial procedure, we can change the common linear Fourier Features from before by mapping not the linear input vector $u = [p|W]$ but rather the polar coordinate representation of $u$ as $\circ u = [r, \theta] = [||p - W||, \arctan\frac{p_y - W_y}{p_x - W_x}]$ to

$$\begin{aligned}\phi(\circ u) = [&a_1\cos(2\pi b_1^T(\circ u)), a_1\sin(2\pi b_1^T(\circ u)),\\ &\cdots a_m\cos(2\pi b_m^T(\circ u)), a_m\sin(2\pi b_m^T(\circ u))]\end{aligned} \tag{4}$$

## 4. Experiments & Discussion

For experimentation, we use the four room designs seen in Figure 4. For each room, we sample 15 different router positions $W$. For each router position, we sample 50 different query points $p$ inside and outside the room. For each router and query point combination, we sample 96 point cloud samples $PC$.

Our preliminary experimentation shows that the performance is heavily affected by the presence and type of Fourier Features used. As such we present our results on three experiments as using (1) no Fourier Feature, (2) linear Fourier Features and (3) radial Fourier Features.

For all experiments, we fix the embedding dimension $d = 64$ and all activation functions to Rectified Linear Unit (ReLU) and employ the log-distance and skip connection implementations explained in the previous sections.

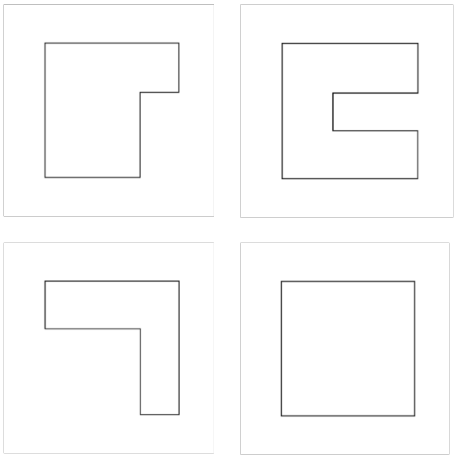

Figure 4. Room structures for experimentation.

### 4.1. Experiment Results

We present quantitative and qualitative results in Table 4.1 and Figure 5 respectively.

| $\mathcal{L}_{MSE}$ | No FF | Lin. FF | | | Rad. FF | | |
|---|---|---|---|---|---|---|---|
| | | 10 | 20 | 64 | 10 | 20 | 64 |
| Training | 23.96 | 18.98 | **15.56** | 20.98 | 19.99 | 17.27 | 21.29 |
| Validation | 23.81 | 19.66 | **16.74** | 22.42 | 19.86 | 18.35 | 24.51 |

Table 1. Quantitative results from experiments. While nearly all cases outperform the baseline, we see that Linear FF works better than Radial FF.

We make the following observations regarding these results:

*Line-of-sight RSS is recovered relatively quickly.* Looking at the first column of Figure 5, we observe that in most cases, our network is able to recover the line-of-sight RSS relatively easily (notice that the area near the router is nearly identical in all cases to the ground truth). This is not surprising as including the log-distance between the router and query points in our input essentially relegates the $H_0$ estimation to a linear mapping (see Equation 2). However, we observe that the attenuations due to the obstructions around the corner are not well-captured by the No FF network as this obstruction presents a high-frequency change in the RSS. Notice that both Linear and Radial FF networks are able to capture this variation.

*Increasing the dimensionality of the Fourier Features causes over-fitting.* Looking at Table 4.1, we observe that while increasing the dimension of the Fourier Features from 10 to 20 noticably increases performance, increasing the dimensionality further from 20 to 64 causes overfitting and results in artifacts (see the last row of Figure 5).

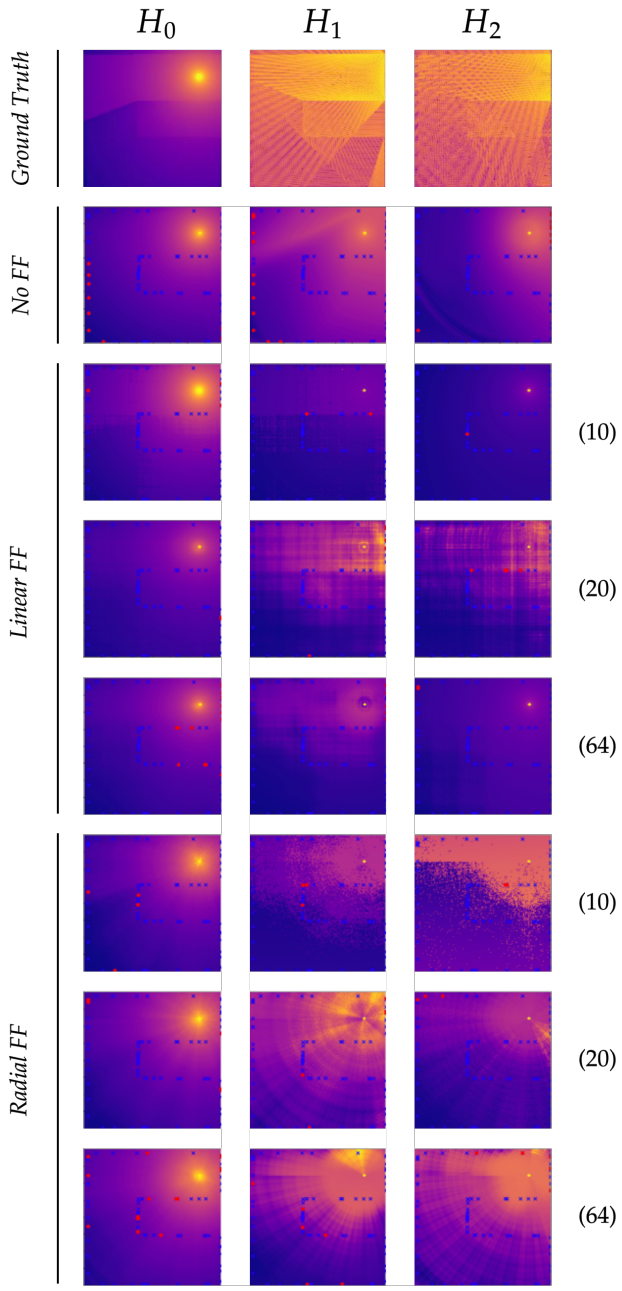


Figure 5. Qualitative results from experiments. Numbers to the right indicate the dimension of the Fourier Features used. No FF is not able to reconstruct the high-frequency details and Radial FF create artifacts.

*Linear Fourier Features outperform Radial Fourier Features.* Table 5 shows that Linear FF outperforms Radial FF in all experiment dimensions. This is likely caused by the fact that Radial FF tries to map two different class of values (the norm is in meters whereas the angle is in radians) in the same way, which causes artifacts.

## 5. Conclusion

Based on our experiments, we conclude that while our network is able to recover accurate heatmaps in the case of new router positions and point cloud samples in rooms similar to the training data, it fails to match performance upon changing the room structure which indicates that training on more room samples would likely resolve this problem.

## References


[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2016. 1

[2] R. S. de Souza and R. D. Lins. A new propagation model for 2.4 ghz wireless lan. In *2008 14th Asia-Pacific Conference on Communications*, pages 1–5, 2008. 1

[3] Zhong Ji, Bin-Hong Li, Hao-Xing Wang, Hsing-Yi Chen, and T.k. Sarkar. Efficient ray-tracing methods for propagation prediction for indoor wireless communications. *IEEE Antennas and Propagation Magazine*, 43(2):41–49, 2001. 1

[4] Ron Levie, Çağkan Yapar, Gitta Kutyniok, and Giuseppe Caire. Radiounet: Fast radio map estimation with convolutional neural networks, 2020. 1

[5] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred A. Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks, 2019. 2

[6] Hemant Kumar Rath, Sumanth Timmadasari, Bighnaraj Panigrahi, and Anantha Simha. Realistic indoor path loss modeling for regular wifi operations in india, 2017. 1

[7] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations, 2020. 1

[8] S P Sotiroudis, S K Goudos, and K Siakavara. Deep learning for radio propagation: Using image-driven regression to estimate path loss in urban areas. *ICT Express*, 6(3):160–165, 2020. 1

[9] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains, 2020. 2, 3

[10] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008. Curran Associates, Inc., 2017. 1


## A. Future Directions

Here we note two alternative solutions as future directions that can be pursued. The methods described momentarily are tangential rather than identical to the problem described in the main paper and aim to recover room structure solely from sparse RSS measurements.

### A.1. Iterative Refinement using Neural Rendering

The first method posed relies on the principle of neural rendering where the room structure is implicitly captured in the weights of an MLP which takes as input $2D$ points and outputs room characteristics (signed distance, normal vector, scattering coefficient, etc.). Using the room characteristics, we then find the specular $N$-bounce paths where $N \in \{1, 2\}$ by starting with a transmission angle $\theta$ and optimizing for a fixed number of iterations.

The primary algorithm is given in Figure 7 which alternates between updating the specular paths and updating the room structure.

As this is an ill-posed problem, we can predict that for this method to work we need a good initial guess for the room structure which we can estimate from $H_0$ using a deterministic method such as convex hulls.

### A.2. Recursive Rendering Using Sub-routers

The second method posed relies on the idea that once we know the incoming direction and energy at a point on the walls, we can treat that point as a sub-router and use it to render other sub-routers.

Then, using a recursive estimation scheme, we can start first by estimating the incoming RSS and direction at points on the wall based on the original (actual) router. Then, for the next iteration, we can calculate the contribution of each sub-router on the other sub-routers and update our sub-router RSS by that amount which corresponds to first reflections. Doing this for $N$-iterations gives us a reconstruction of the RSS up to $N$-bounce reflection.

The main difficulty with this approach is that the method by itself is very similar to ray-tracing, however adding the task of recovering the room structure to the problem makes this somewhat intractable as now, the sub-router positions change between iterations and must be somehow interpolated from previous iterations. This could likely be approached by an update mechanism which ensures that the room structure is updated in a way that does not drastically alter the specular path (a wall segment moving orthogonally to its normal should affect the specular reflection path less than say it moving parallel to the normal).

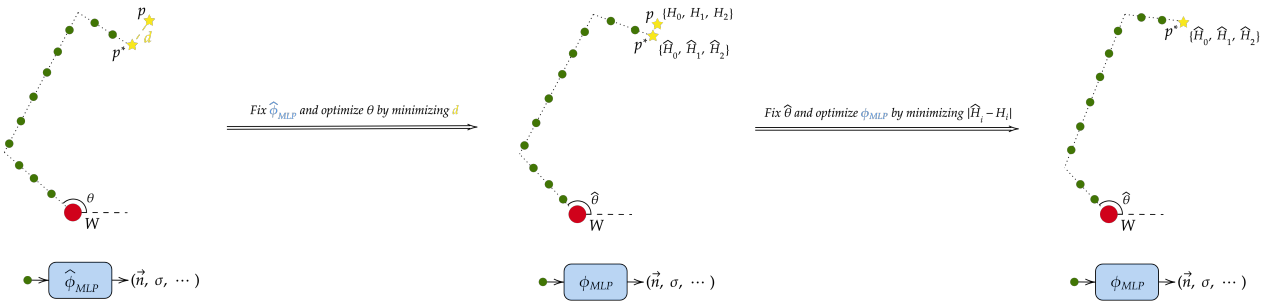


Figure 6. Iterative Refinement using Neural Rendering. In each iteration, we alternate between optimizing room structure and finding specular paths.

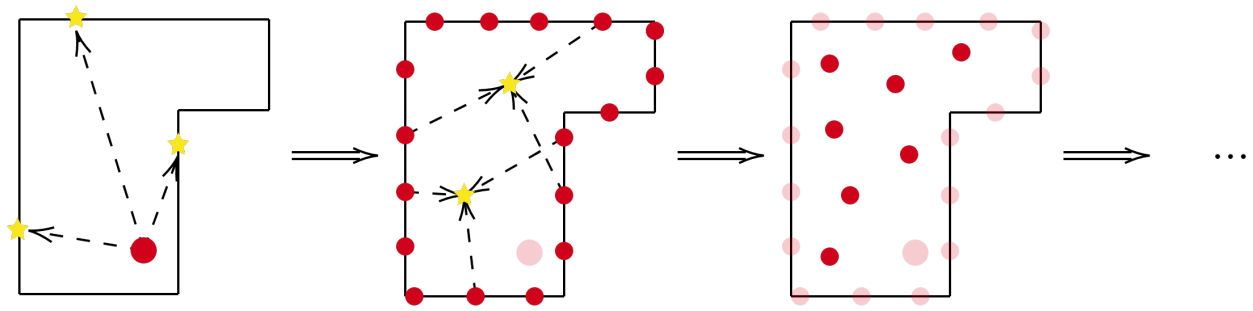

Figure 7. Recursive Rendering Using Sub-routers. In each iteration, we update the incoming energy at each sub-router based on the sub-routers in the previous iteration.