GRIPS Sendai 2018

NEC Group

Final Report

Indoor Localization with WLAN Technology

Authors
Itaru Abe
Shizuki Goto
Ryoichiro Hayasaka
Hannah Horner
Eric Lybrand (Project Manager)

Supervisors
Dr. Kuniyasu Saitoh
Mr. Kousuke Nogami

August 10, 2018

GRIPS NEC Final Report: WLAN Indoor Localization

Itaru Abe, Shizuki Goto, Ryoichiro Hayasaka, Hannah Horner, Eric Lybrand

I. Introduction

NEC corporation is a multinational company based in Tokyo, Japan, which provides Internet and Communication Technology (ICT) services and products. Their customers include individuals, business enterprises, and government agencies. A few examples of their products and services include electrical machinery, Wi-Fi routers, and telecommunications services. NEC uses artificial intelligence to collect and process massive amount of data from ICT systems. With this data NEC analyzes various trends to address macroscopic problems that arise, for example, in manufacturing, transportation, health care, and security.

Problem Summary & Statement

A widespread problem in telecommunications includes locating where a particular transmission is being generated. With the advent of smartphones, the applications of a solution to this problem are manifold. For example, locating a customer within a store would allow a marketer to create location-based advertisements. Alternatively, locating victims of a natural disaster inside an obstructed or partially destroyed building would aid in emergency rescue operations. Perhaps the most well-known positioning system is Global Positioning System (GPS). GPS is a radio-navigation technology that localizes any GPS enabled device within the line of sight of at least four GPS satellites [9]. For this reason GPS excels in positioning in outdoor environments with few obstructions but demonstrates a poor performance in indoor settings. Therefore, a new method of localization is required for the indoor situation usage. Many other localization protocols using radio waves have been proposed that use, for example, wireless local area networks (WLAN) [4], [11], [15], Bluetooth technology [11], or radio frequency identification (RFID) [11].

Among these, WLAN-based localization systems have several advantages. Firstly, WLAN infrastructure is widespread. Many mobile devices now have Wi-Fi capability, and Wi-Fi coverage is becoming more commonplace. Further, WLAN infrastructure is cost-efficient to install and maintain. Despite these advantages, WLAN-based localization suffers from some major disadvantages. Most WLAN localization systems rely on received signal strength (RSS) readings as a proxy for distance from a given wireless access point (AP). These readings are often very noisy and are quite sensitive to factors such as device orientation, building composition, and the placement of APs. Further complicating matters is that radio waves are prone to multipath propagation in indoor settings, a phenomenon where a radio wave reflects off of walls and other surfaces and “echoes” back to the receiver. Aside from multipath propagation, interference and diffraction also exit. Interference is due to the interaction of waves that are correlated or coherent with each other. Diffraction is the phenomenon referred to the

bending of waves around the corners of an obstacle. These effects of the radio waves make it difficult to discern the true distance a transmission travels to a particular AP. Nevertheless, RSS values are easily accessible from various devices, as any device adhering to the IEEE 802.11 protocol makes RSS readings available to the user [11].

Many approaches have been formulated to account for multipath propagation, attenuation, and other effects from indoor settings. Two notable approaches include fingerprinting [15] and path loss models [4], [8]. Fingerprinting requires an offline learning phase of measuring RSS values at particular reference points in a building. Fingerprinting approaches demonstrate some robustness to multipath propagation but are prone to degrade in quality as room layout and other environmental factors change. Further, the process of gathering fingerprints is costly. Alternatively, path loss models are governed by how radio-waves propagate. Most path loss models assume rather simplistic conditions, namely that the radio wave is being transmitted through some uniform medium, or do not take into account distance [15]. Many pass-loss models also include parameters which are finely tuned based on a specific indoor environment. While easy to formulate, these over-simplistic models often cause the performance of such methods to fall short of attaining a high localization accuracy.

Motivating Research Question Is there mathematical model for path loss that does not require fingerprinting and captures the complexities of radio-wave propagation in an indoor environment that leads to a better localization performance than other path loss models?

Our Approach

Our initial approach aimed to better capture the complexity of an indoor environment by adding some properly motivated and deterministic parameters to the path loss model. In other words, we aimed to estimate the signal quality, e.g. RSS reading, *without fingerprinting*. Having a successful model that avoids fingerprinting would amount to a huge cost savings, as fingerprinting methods are the preferred choice for indoor localization techniques. As an added bonus, this model can also help us determine where to optimally place APs in a given indoor environment.

In the first half of the project, we succeeded to set up devices and start collecting RSS data in our office. A prototype of RSS simulation was developed to estimate the RSS distribution with simple inputs. After the midterm presentation, we discussed with our sponsor company and decided several directions for the latter half of the project.

During the second half of our approach we calibrated our RSS simulation and analysis, while avoiding fingerprinting as much as possible so that the implementation of localization can be available at construction sites or factories. Our methods are adaptable to different indoor settings and can be adjusted to aid in optimizing AP placement and localization. We combine our simulation results and signal analysis to create a dynamic path loss model that improves upon existing path loss models for indoor localization. Finally, we conclude with using our simulations to optimally place

wireless access points in a given indoor environment. For optimal AP placement, we use k-means clustering to determine the placement of multiple APs that maximizes wireless coverage.

Deliverables We have prepared the following products for the corporate sponsor.

1) A midterm presentation.
2) Lightweight software that estimates RSS for a given indoor environment with minimal input from the user.
3) A trainable, dynamic model for indoor path loss which can be used for indoor localization.
4) Lightweight software that uses our RSS simulating software and a simple floor plan to generate candidate placements for AP units to optimize coverage.
5) A final project's day presentation.
6) A final written report that details our work and illustrates the performance of our simulations and indoor localization techniques.

Our Team

Based on the variety of talent in our team, we had the following breakdown of roles over the course of this project.

Team Member	Area of Expertise	Role
Itaru Abe	Materials Science	Experimental Design
Shizuki Goto	Environmental Science	Simulation Team & Experimental Design
Ryoichiro Hayasaka	Mechanical Engineering	Simulation Team & Indoor Localization
Hannah Horner	Mathematics: Optimization	Simulation & AP Placement Model Design
Eric Lybrand	Mathematics: Signal Processing	Project Manager & Indoor Localization

Software Resources

Below, we provide a list of the software that was used in our endeavors.

Software Name	Purpose
Wireshark	Network Diagnostics, Data Collection, Fingerprinting
Matlab	Data Filtering and Analysis
Python	Algorithm Design and Implementation
RStudio	Data Visualization

A. Report Outline

The remainder of this report will be structured as follows. In Section II we give some background on technical terms that will be needed later on in our analysis. Section III details how we conducted our experiments and what data we have collected. Section IV pertains to our indoor simulation software and comparing it to recorded RSS readings. Section V discusses our method for processing RSS data for indoor localization and designing a dynamic path loss model. Section VI concludes with using our RSS simulations for optimal placement of access points.

II. Background

A. Wavelets and Jump Detection

Wavelets are a popular tool in the signal processing community for a wide variety of tasks such as compression and edge detection [3]. Wavelets offer a competitive advantage over classical frequency analysis in that they are localized both in frequency and in time. In other words, wavelets can analyze the variance of a curve over various time scales and over various frequencies. In our experiments, we decided to employ Haar wavelets to detect jumps in the RSS data. At scale $k \in \mathbb{N}$, we have the collection of wavelets of the form

$$\psi_{k,b}(t) := \frac{1}{\sqrt{2^k}} \psi\left(\frac{t - b2^k}{2^k}\right),$$

$$\psi(t) = \begin{cases} 0 & t < 0 \\ 1 & 0 \le t < 1/2 \\ -1 & 1/2 \le t < 1 \\ 0 & 1 \le t \end{cases}$$

Wavelet coefficients are then computed by taking inner products of a signal f with $\psi_{k,b}$. In the discrete case, the values of f are stored in a vector and the shifts of $\psi_{k,b}$ are taken by shifting indices rather than taking a continuous translation of a curve. This means that, for the discrete case, the number of data points in the support of a given wavelet at scale k is 2^k. The amount of time elapsed in this temporal window depends on the rate at which the RSS is sampled. Specifically, the j^{th} coefficient at scale k corresponds to the indices in the range $[1 + (j-1)2^k, j2^k)$.

Wavelet coefficients which are large in absolute value are indicative of sharp discontinuities. Specifically, large positive wavelet coefficients indicate a sharp drop in function value, whereas large negative wavelet coefficients indicate a sharp increase in function value.

B. The k-Means Algorithm

The k-means algorithm is an iterative method that splits a set of data into k clusters, and each cluster has a center point representing the average location of the points in that cluster. After initial centers are chosen, each data point is assigned to the cluster with the closest center. Once all data have been classified, the centers are updated to be the centroid of the data assigned to that cluster. This iterative process

of classifying data and updating centers is repeated until convergence is achieved or maximum iteration count is reached. This results in the data being clustered so that the sum of the squared distances from each data point to the nearest center is minimized. For an example of the k-means iteration see Figure 1.

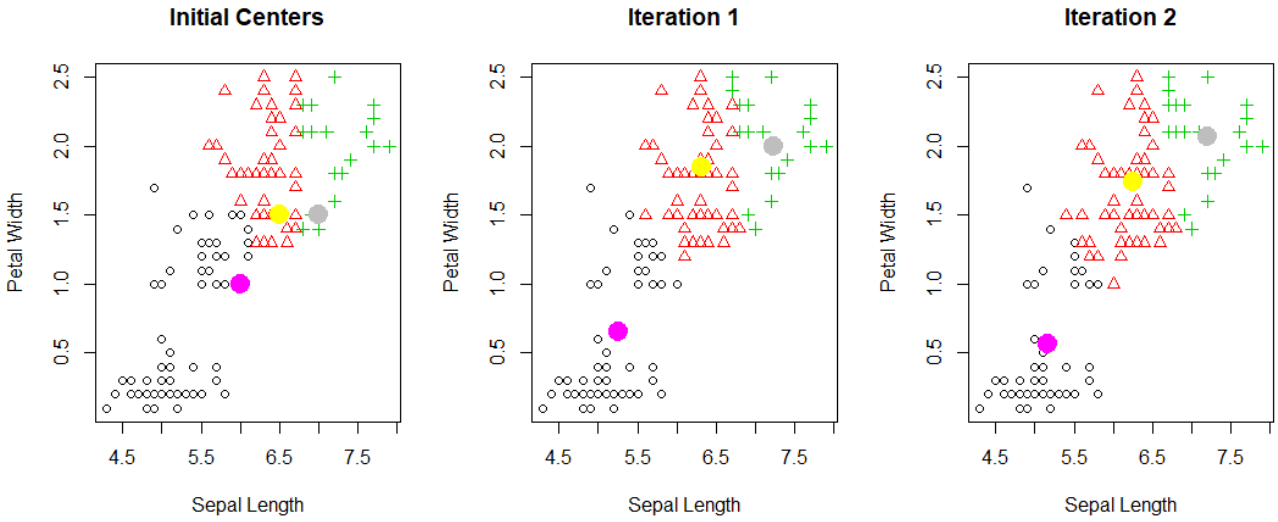


Fig. 1: The first few iterations of k-means applied to an example set of data. The data is from the well-known Iris dataset that has various measurements of iris flowers [6]. The x-axis and y-axis are two variables chosen from the dataset. The large solid-colored circles are the current cluster centers, and the red, green, and black data points have their color and shape based on which cluster they currently belong to. The pink cluster center represents the cluster made by the black circle data; red triangle data points belong to the cluster with the yellow center, and green pluses belong with the grey center.

III. Experimental Design

To monitor the behaviour of received signal strength throughout the office, we conducted 39 experiments with varying routes taken as well as with varying conditions of line of sight. We use these experiments to train our dynamic path loss model as well as to judge the quality of our received signal strength simulations. Our wireless access point is a Buffalo WXR-1750DHP2 router. To monitor and record RSS, we used two Raspberry Pi 3s with Kali Linux installed. One Raspberry Pi was set to monitor mode to capture all network traffic while the other Raspberry Pi was set to send and receive packets to and from the access point. Packets were captured with Wireshark. Captured data were filtered by transmitter and destination address so that the data were limited from router(AP) to Raspberry Pi3(RP). We set the Raspberry Pis to send and receive packets at the 2.427 GHz channel and at various rates using the `ping` command.

TABLE I: Table of experiment details. Data includes experiment file names, sampling rates, starting reference points, the time interval that the experiment was recorded during, and the figure that illustrates the path taken.

Name	Sample Rate	Time Interval	Path Figure
20180710_S1_f_[1-3]_v2	2 Hz	[25s, 190s]	
20180710_S2_f_[1-3]_v2	2 Hz	[20s, 275s]	Fig.2
20180710_S3_f_[1-3]_v2	2 Hz	[20s, 110s]	
20180723_S1_[LOS,NLOS]-[1-3]	2 Hz	[30s, 130s]	
20180723_S2_[Close,Open]-[1-3]	2 Hz	[20s, 90s]	Fig.3
20180723_S3_[Close,Open]_[Close,Open]-[1-2]	2 Hz	[30s, 130s]	
20180724_S1_SR[01,02,10]-[1-3]	[10, 5, 1] Hz	[25s, 200s]	

First, we created a uniformly spaced grid of reference points across the office spaced 2 meters apart showed in Figures 2-3 as orange diamonds. Reference points were labelled with coordinates (x, y), where the origin $(0,0)$ denotes the top left reference point, $(0,11)$ denotes the bottom left reference point, and so forth. The office was divided into three disjoint sections whose labels are S1, S2, and S3. From there we established routes to take a selection of measurements for each section. Each experiment involved walking along a particular path of reference points and recording the times at which each reference point was visited. In each experiment, each reference point was visited for 10 seconds, followed by a 5 second transition to the next reference point on the route. Figures 2-3 and Table I detail the precise conditions of each experiment. Each experiments clock was started at the point designated RP in the maps. Blue arrows indicate the path before and after distance information was recorded. The red arrows indicate the path where distance information was recorded.

Each experiment was repeated 2 or 3 times. In experiments with the [LOS, NLOS] flag, the data were captured based on whether or not the Raspberry Pi had line of sight during the experiment or not. In the routes with the [Close, Open] flag, the data were captured while keeping the doors in Figure 3 open or closed.

IV. RSS Simulation

Our experiments detail how RSS varies continuously throughout our office environment. We aim to model this behavior with a simple, lightweight simulation program. Some proprietary programs, such as *Wise* [7], can simulate RSS in indoor environments. These programs need a lot of input information about the environment such as 3D structure of the room, materials of the furniture and so forth. Consequently, they are computationally expensive. Moreover, inputting an extensive amount of information every time is not realistic to apply to different environments. Since an application of this simulation is the optimization of AP placement, in which many combinations of APs' locations have to be considered, we decided to make a lightweight program. For simplicity, the input information of our simulation is only AP and wall locations in 2D. All walls are assumed to be the same material (concrete or metal) of the same thickness. In addition, only one reflection is considered although reflections

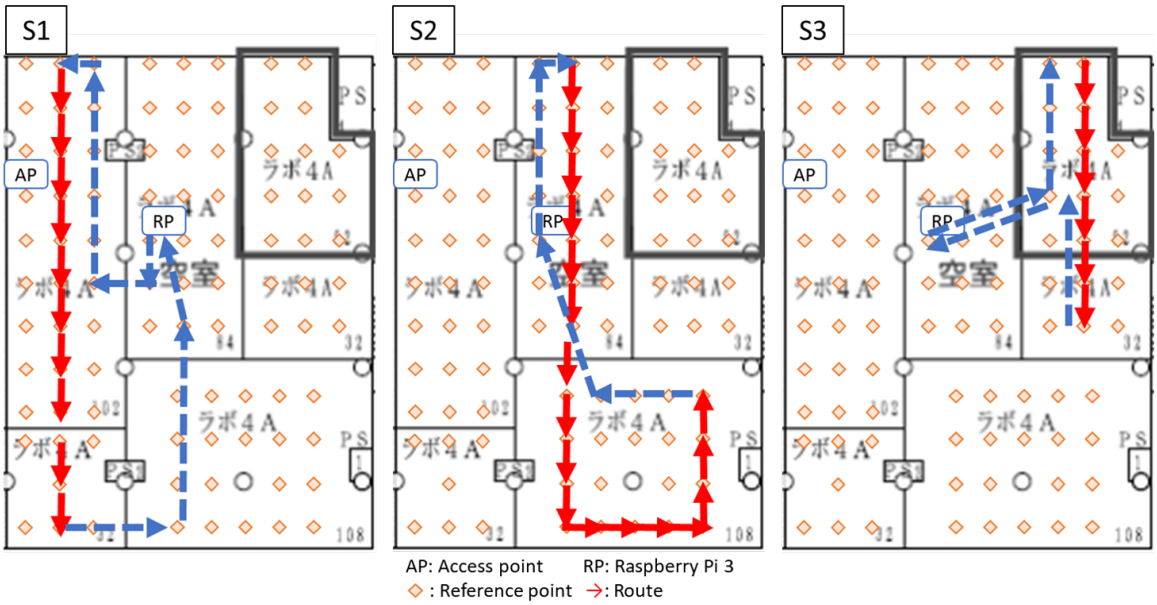


Fig. 2: S1-S3 for the experiments starting with 20180710. Blue dotted arrows show the continuous recording route and red solid arrows show the periodic recording route. S1 measurements were recorded from RP $(1,0)$ to $(1,11)$. S2 measurements were recorded from RP $(4,0)$ to $(8,8)$. S3 measurements were recorded from RP $(7,0)$ to $(7,6)$.

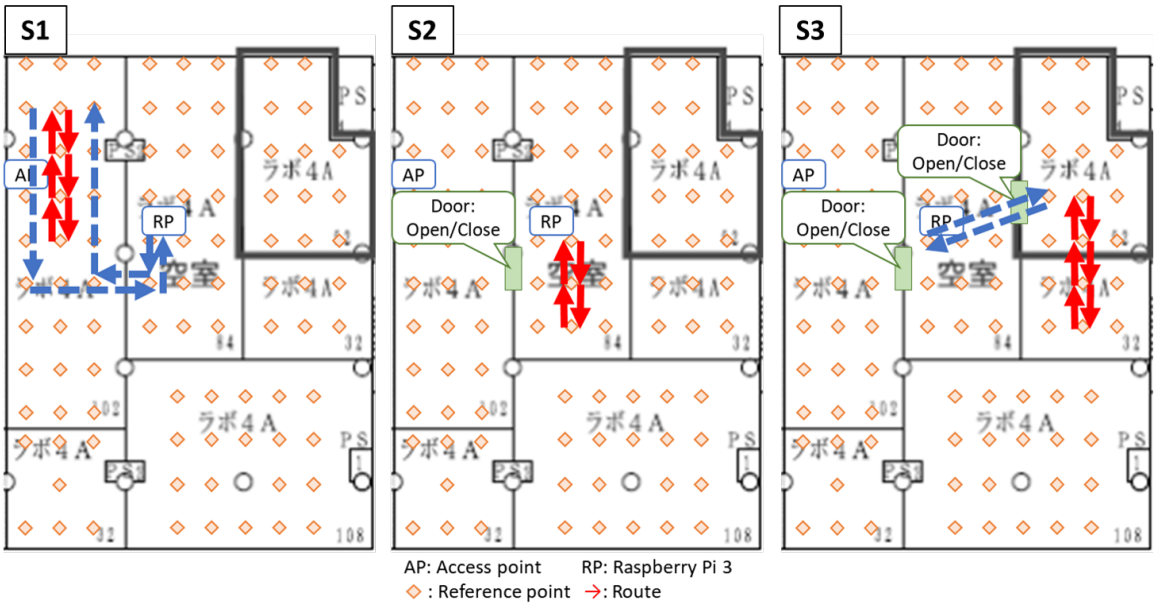


Fig. 3: S1-S3 for the experiments starting with 20180723 or 20180724. Blue dotted arrows show the continuous recording route and red solid arrows show the periodic recording route. S1 measurements started and ended at RP $(1,1)$. S2 measurements started and ended at RP $(4,4)$. S3 measurements started and ended at RP$(7,3)$.

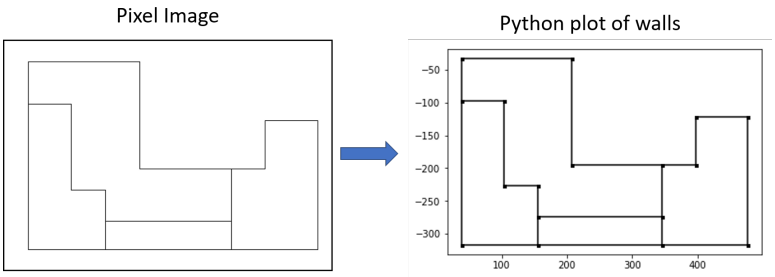


Fig. 4: An example of the walls found for the demo floor plan we created.

in practice. We have confirmed this assumption is valid in Figure 8. Interference is also ignored for simplicity, and diffraction can be neglected because of the small wavelength of Wi-Fi.

A. Generating Walls

To allow the user to easily create/replicate more complex floor plans that can be used in our simulations, we wrote a quick program that takes in pixel images and outputs a list of the coordinates of the endpoints for all of the walls in the image. Walls are assumed to be any straight line made of black pixels (with thickness of one pixel). As an example Figure 4 shows the walls generated for a simple building plan that we created to use in our analysis. When we made a pixel image to represent the 4th Floor of our office building AIMR, we assumed that each pixel has the side length of 4cm. Note that our program can only detect horizontal and vertical walls, so walls that are at a different angle must have the endpoints inputted by hand. This was the case with AIMR's 4th floor, where a few of the walls in the upper right region were input manually.

B. Implementation

The algorithm of our program is as shown in Algorithm1. In this algorithm, the direct path and paths with one-reflection are found from an AP to a grid point. Along each path, power decay is calculated by equations in Table II. Distance decay is calculated by the Friis loss equation as well as some articles [7], [10]. For reflection and pass-through loss, initial power P_n is updated to current power P_{n+1} based on an incident angle θ_i every time a path encounters a wall. Parameters used in the Table II equations are described in Table III.

C. Initial Model

Following the algorithm mentioned above, we conducted a RSS simulation in our workplace whose dimensions are 20 m × 15 m. Figure 5 shows the RSS heat map by our initial model, where an AP is represented as a red point and walls are expressed as black lines. In the simulation, we left the doors open (described as break-off parts of lines) to get strong RSS values. Nevertheless, signal power decayed significantly,

Algorithm 1

for each Access Point AP **do**
  for each grid point p **do**
    for each wall w **do**
      Check for a path from AP to p that reflects off of w
      Calculate the power of the ray along the reflection path (if exists)
    end for
    Calculate power of direct path from AP to p
    Aggregate the power of all paths into one RSS value
  end for
  Build heat map from simulated RSS values
end for

TABLE II: The equations used to model the power of each ray path found by our algorithm.

Friis equation (distance decay)	$P_r = P_t \left(\frac{c}{4\pi f}\right)^2 \left(\frac{1}{d}\right)^2 G_{ra} G_{ta}$
Reflection loss	$P_{n+1} = R_{ref(\theta_i)} P_n$ $R_{ref(\theta_i)} = \left(\frac{\tan(\theta_i - \theta_t)}{\tan(\theta_i + \theta_t)}\right)^2$ $\frac{\sin\theta_i}{\sin\theta_t} = \frac{n_{wall}}{n_{air}}$
Pass-through loss	$P_{n+1} = R_{pas} P_n = a(1 - R_{ref(\theta_i)}) P_n$
Convert Power to RSS	$P_{all} = P_r \prod^{N_{ref}} R_{ref} \prod^{N_{pas}} R_{pas}$ $RSS \text{ (dBm)} = 10\log_{10}\left(\frac{P_{all}}{1 \text{ mW}}\right)$

decreasing to as low as -280 dBm which was impossibly small for RSS in this room. Therefore, we decided to improve our model as mentioned next.

D. Model Improvement

1) Modification of distance decay: Reexamining our initial algorithm, we soon realized our distance decay calculation was slightly off. For paths that included a reflection off of a wall, we had found the length of the two segments of the path (AP to wall and

TABLE III: List of Parameters

Description	Parameter	Value	Units	Reference
Received power	P_r		mW	calculated
Transmission power	P_t	10	mW	
Speed of light	c	3×10^8	m/s	
Distance (unwrapped)	d			calculated
	f	2.4	GHz	
Receiver Gain	G_{ra}	1		est
Transmitter Gain	G_{ta}	1		est
Incident angle	θ_i			calculated
Refractive angle	θ_t			calculated
Refractive index of concrete	n_{wall}	1.6		est
Refractive index of air	n_{air}	1		est
Transmission rate	a	0.316		est[16]
Reflection rate of metal	R_{ref}	0.7		est[1]
Count of reflections	N_{ref}			calculated
Count of pass-through	N_{pas}			calculated

wall to user) and applied the Friis equation to each segment separately. Because the Friis equation uses distance squared, and because the formula involves multiplication by constants, applying the formula to segmented parts of a path is not equivalent to applying it once to the entire path. We corrected this error so that the Friis equation is only applied once and the two segments' lengths are combined to find total path length. After the improvement, RSS in our workplace was simulated again. Figure 6 shows the simulated RSS by the improved model. In this simulation, we left the doors closed in order to compare simulated RSS with experimental RSS. Fortunately, the signal power in this simulation does not decay so significantly as it did in the initial model. RSS values range over -80 dBm, which is reasonable compared to the recorded RSS. Since this simulation is based on some estimated values, the effects of these values are examined. We found that the possible difference in the refractive index of concrete and its transmission rate does not affect RSS significantly. As for the transmitter gain G_{ta} and the receiver gain G_{tr}, we set both of them to 1, which is a common value for an omni-directional antenna [14].

2) Two reflections: In this section, we calculate the effect of paths with two reflections. After the model improvement, we made an advanced program that considers two reflections. Yellow dotted lines in Figure 7 express two-reflection paths in an example room, where red and blue points represent a AP and a grid point, respectively. The paths are found by searching each combination of two walls. At first, in Figure 7, we calculated RSS considering two-reflection paths. The calculated RSS was -45.819 dBm, which is only 0.157 dBm bigger than RSS without two-reflections paths. Since walls are few in this case, we expected the two-reflection effect might be bigger in a larger room with more walls. To investigate the reflection effect, we compared RSS heat maps with no, one and two reflections in a large floor, whose dimensions are 40 m × 30 m, to see reflection effects as shown in Figure8. Among these figures, (a) indicates the concrete-wall case and (b) the metal-wall case. In the concrete-wall case, there is an apparent difference between no reflection (a.1) and one reflection (a.2), while

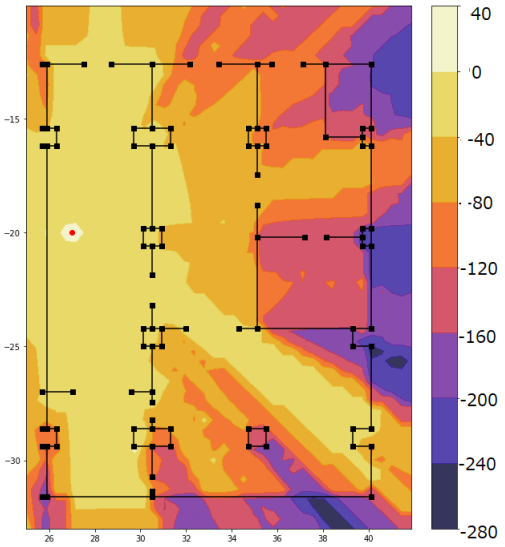


Fig. 5: RSS heatmap generated by the initial simulation model considering paths with up to one reflection.

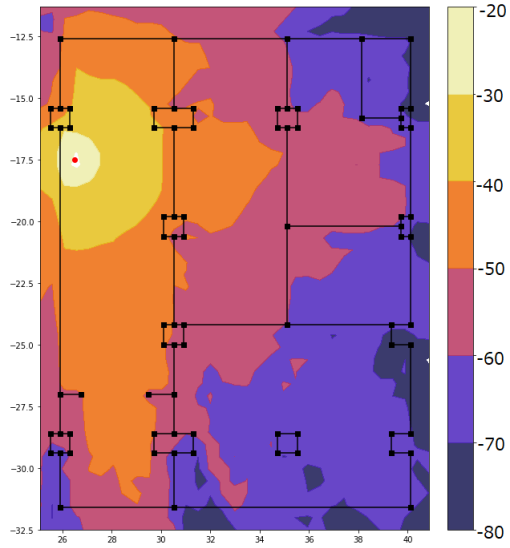


Fig. 6: RSS heatmap generated by the improved simulation model considering paths with up to one reflection.

no apparent difference between one reflection (a.2) and two reflections (a.3). In fact, difference in RSS between the one-reflection model and the two-reflection model is 0.015 dBm on average, and the biggest difference is 0.32 dBm of all the grid points. For the metal-wall case, there is a more significant difference between no reflection (b.1) and one reflection (b.2) because of the high reflection rate of metal, whereas there is no significant difference between one reflection (b.2) and two reflections (b.3). The average difference between the one-reflection model and the two-reflection model is 0.0016 dBm and the maximum difference is 0.11 dBm. Therefore we ascertain that the one reflection model can simulate RSS sufficiently. Moreover the running time of the one-reflection model is much quicker than the two-reflection model, which enables the calculation for the optimization of AP placement in a reasonable computation time.

E. Quantitative Performance Analysis

Simulated RSS values are analyzed quantitatively by comparing with experimental data that is mentioned in Section III. In Figure 9, simulated RSS values are compared with experimental values collected at several points along red paths in the room. Experimental plots include the average (red squares), maximum and minimum (error bars) of three experiments. Some points do not have error bars because of the missing data. Others have wide error bars probably because slight changes in device orientation, or the direction that the receiver antenna is pointed, allow obstructions, such as computers and whiteboards, interrupt signals. Especially, we have verified that the third point in S1 was affected by a computer monitor, which resulted in a wide error bar. The standard deviation of the error between simulated and experimental RSS is

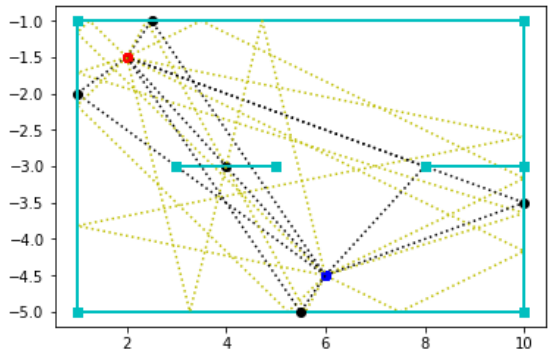


Fig. 7: An example of the two-reflection path detection. The red and blue squares denote the access point and transmitter, respectively. Dotted black lines indicate single reflection paths. Dotted yellow lines indicate double reflection paths.

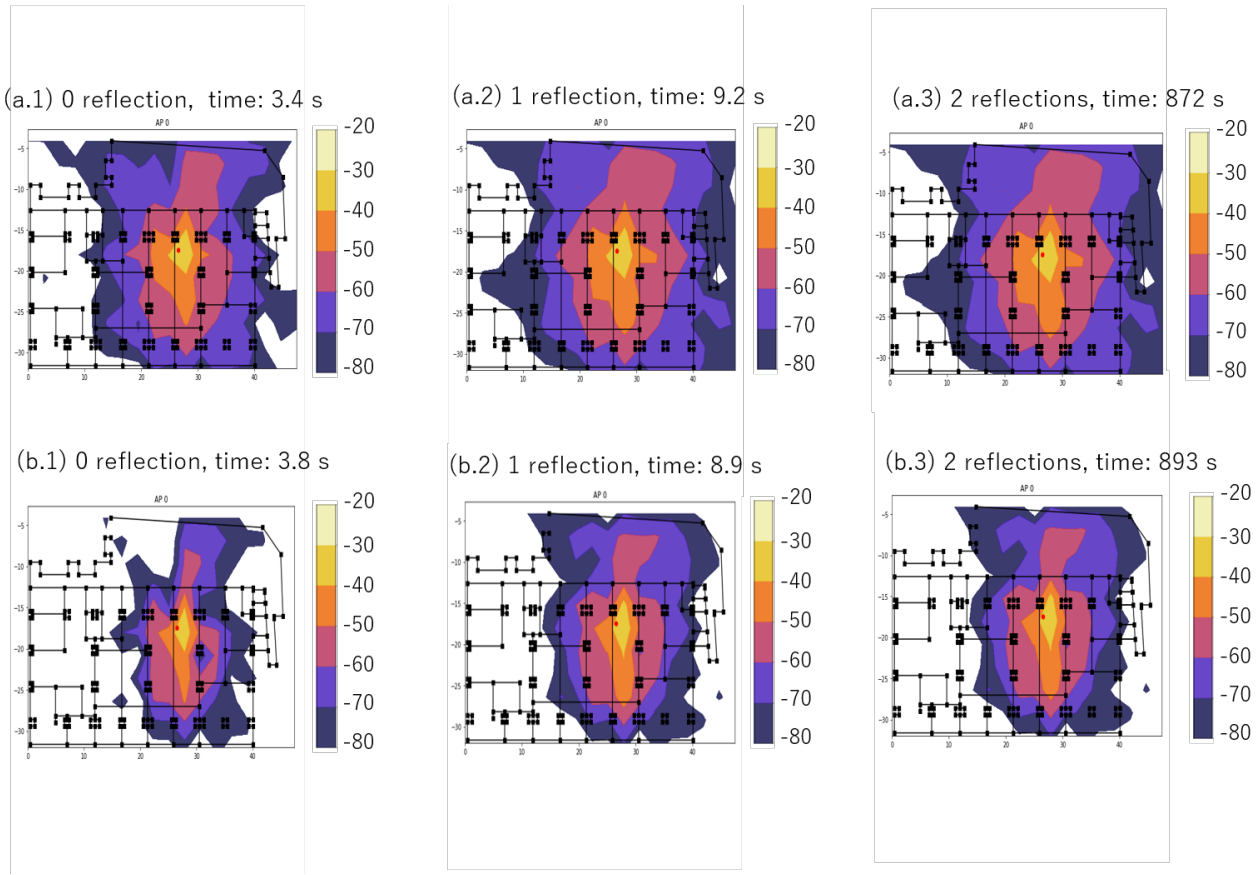


Fig. 8: Simulated RSS heatmaps across the fourth floor of the AIMR office building with zero, one, and two-reflection models. Figures (a) 1, 2, 3 consider the case when walls are made of concrete. Figures (b) 1, 2, 3 consider the case when the walls are metal.

7.2 dBm, which is on the level of variation that can be observed by various changes in the environment. Thus, we concluded that this simulation provides quantitatively accurate RSS values.

Additionally, simulated RSS is plotted as a function of distance from an AP in Figure 10. Figure 10 shows simulated RSS decreases along with the Friis equation in the LOS situation and decreases linearly in the NLOS situation. Experimental data show a better agreement with the NLOS simulation data than with the LOS data. This is because most experiments were conducted in the NLOS situation. In the NLOS condition, power passing through and reflecting walls are diminished by constant ratios, which causes the exponential decrease of the power, consequently, the linear decrease of RSS. A regression line in Figure 10 is derived from NLOS simulation data, which is used for localization in Section V.

V. Indoor Localization

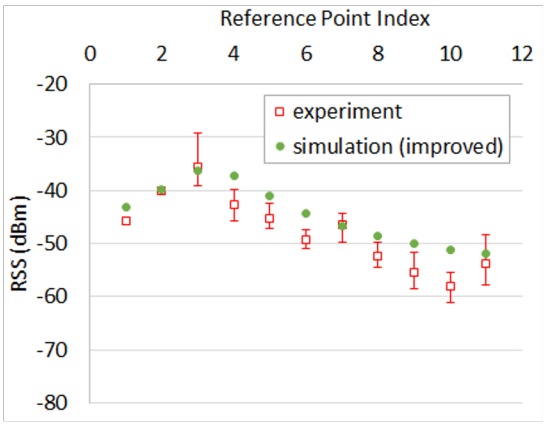
There are many techniques for indoor localization that use various data such as fingerprints, channel state information, received signal strength (RSS), angle of arrival, and time of arrival. For a survey of the literature, please refer to [4], [11]. RSS is the only data listed above that is made available among almost all wireless-capable devices. Fingerprints are used to form a database of premeasured RSS readings at prerecorded locations and are used to localize by comparing actual measurements to these values. Unfortunately, fingerprints are costly to collect and their quality degrades as the indoor environment changes.

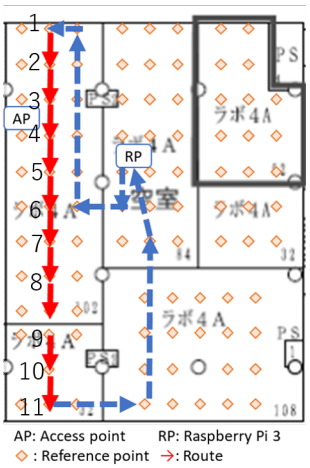
To avoid the expense of collecting fingerprints, we consider using path loss methods for localization. The most common path loss model is the log-distance path loss model which expresses RSS as a function of distance via

$$RSS = T_x - 10n\log_{10} d, \tag{1}$$

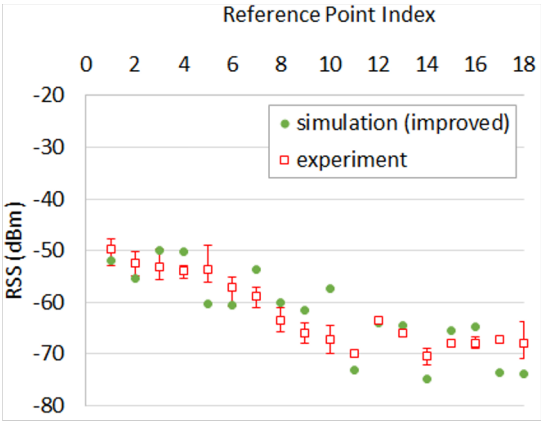
where T_x is the transmission power (dBm) of the access point and n is the path loss exponent [8], [4]. Typically, n is chosen by optimizing distance estimation using a few fingerprints of a particular location and is then fixed henceforth. For example, in the Friis free space model used to calculate distance decay in our simulation, $n = 2$. While static path loss models function well in homogeneous environments with few line of sight obstructions, they perform poorly in indoor settings where small obstructions such as furniture or people blocking the line of sight can cause sharp fluctuations in received signal strength. Our experiments corroborate these claims. Figure 11 illustrates the severity of these jumps in the case where the line of sight is obstructed by a computer monitor. These jumps are not captured by our simulations since our simulations only consider line of sight obstructions caused by walls.

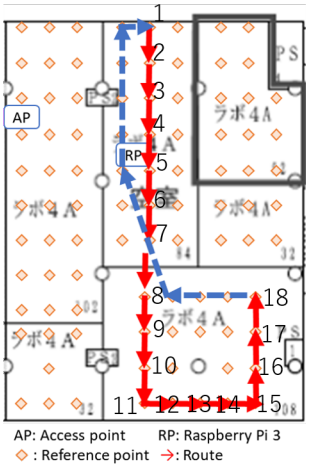
To remedy these issues we propose using an indoor localization algorithm which dynamically adjusts the path loss exponent based on macroscopic trends detected in the RSS values. To the best of our knowledge, this work is the first to consider such a model. In order to detect when the path loss exponent should be adjusted, we focus on detecting sharp jumps in the curve generated by RSS as a function of time. Our experiments indicated that sharp jumps are due to changes in the severity of line of



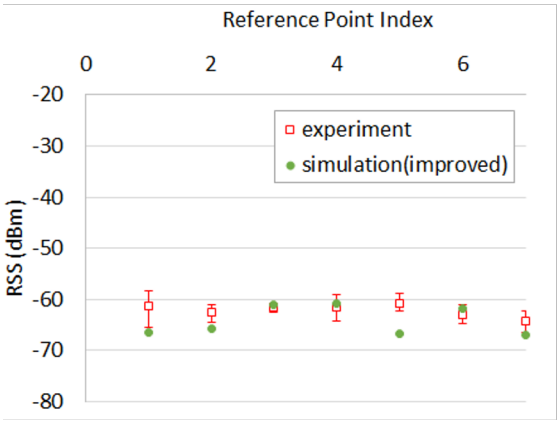


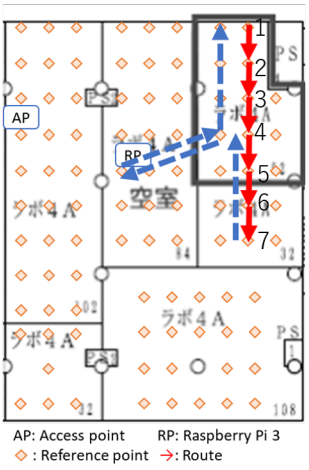
(S1)





(S2)





(S3)

Fig. 9: Comparison of simulated RSS to recorded RSS at reference points in S1, S2 and S3. Orange diamonds indicate reference points. The path where measurements were recorded is indicated with red arrows. Blue arrows indicate the path taken before and after the experiment's time frame. Experimental plots include the average, maximum and minimum of three experimental RSS.

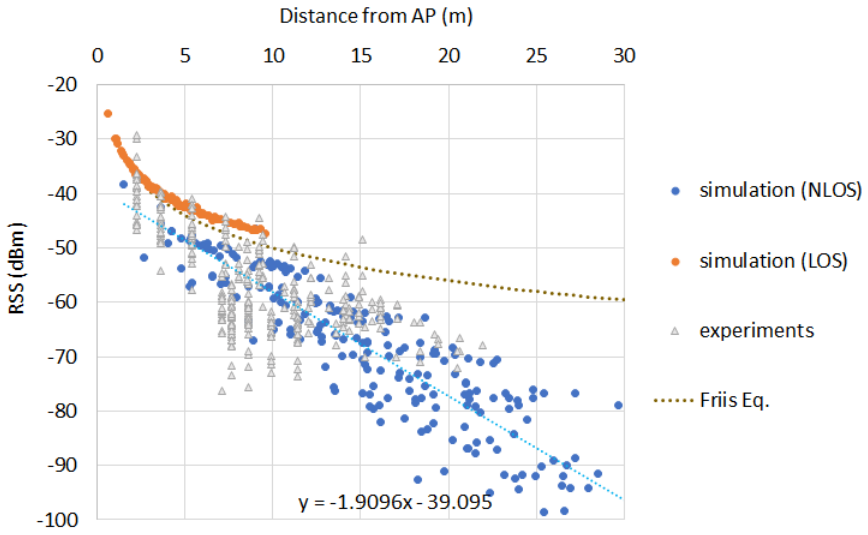


Fig. 10: Comparison between experimental and simulated RSS as a function of distance from a access point

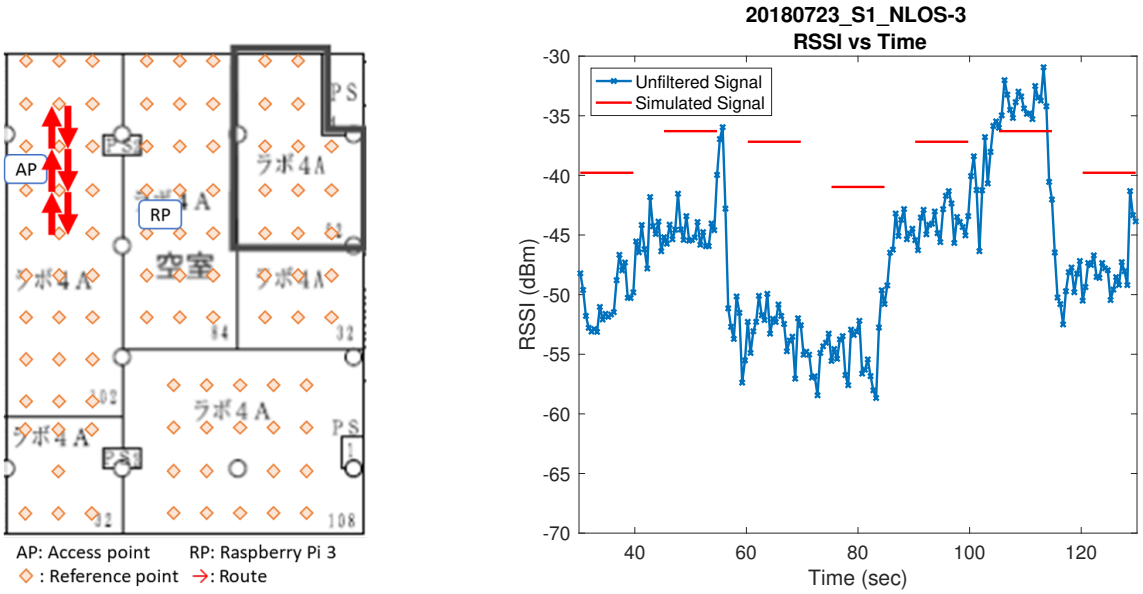


Fig. 11: RSS from experiment 20180723_S1_NLOS-3. Large jumps are due to obstruction of line of sight by a computer monitor. Predicted RSS from our simulations at these reference points is plotted in red. Standard deviation of RSS is 6.39 dBm.

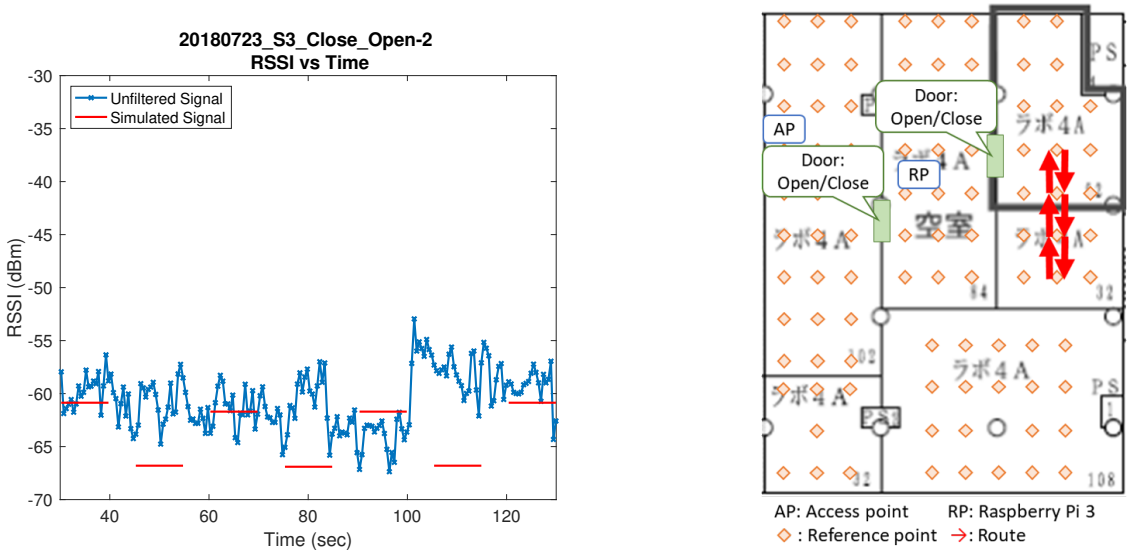


Fig. 12: RSS from experiment 20180723_S3_Close_Open-2. Predicted RSS from our simulations at these reference points is plotted in red. Standard deviation of RSS is 2.68 dBm.

sight or non-line of sight conditions. Consequentially, it seems appropriate to adjust the path loss exponent appropriately.

A. Filtering RSS Data

RSS is a notoriously noisy variable, as it depends on various environmental factors which change over time such as furniture placement, the number of people obstructing the field of view of the access point, and so forth. The biggest challenge for detecting jumps is to distinguish between jumps due to noise and jumps due to more significant events such as changing rooms.

We tried various ways of denoising the RSS measurements. We found traditional filtering techniques such as applying a low-pass or a band-pass filter required too many parameters to finely tune and demonstrated mixed performance on a variety of experiments. For example, some of our paths were composed of very low frequencies such as those taken on July 10th, 2018. On the other hand, experiments taken on July 23rd, 2018 are composed of higher frequencies due to the cyclic nature of the paths traversed. This range in the spectrum of the RSS data necessitated the use of a more universal filtering scheme. The most simple and effective method we tried uses wavelet denoising. The work of [5] provides a method of filtering signals by shrinking empirical wavelet coefficients under the assumption that the true signal is corrupted with Gaussian noise. Rather than choosing pass-bands, stop-bands, and filter types we only need to choose a wavelet scale to denoise at. Before denoising, we resampled our signals at the median empirical sampling rate. We found that despite setting a uniform rate during our experiments, environmental factors caused packet loss and time delays that resulted in non-uniform sampling. After resampling, we

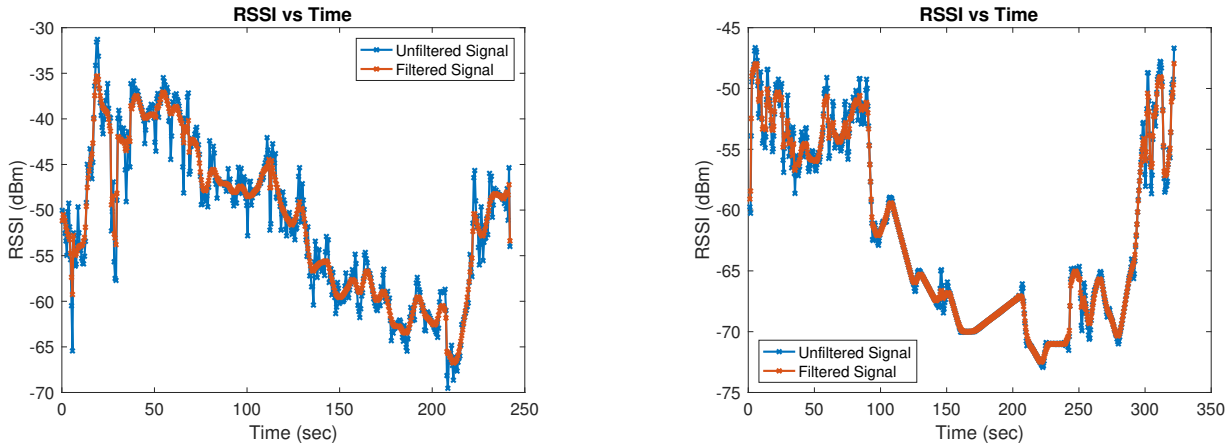


Fig. 13: Comparison of the filtered and unfiltered RSS data from experiments 20180710_S1_f_1_v2 and 20180710_S2_f_1_v2 on the left and right, respectively.

center our signal by subtracting the empirical mean and then divide by the empirical standard deviation. Once these transformations have been applied, we choose the wavelet scale adaptively based on the sampling rate so that each wavelet was localized in a 5 second time window. We observed empirically that most jumps observed over this time period. Figure 13 illustrates how the filtered curves differ from the raw data.

B. Jump Detection

As mentioned earlier, it is important to distinguish between jumps in the RSS curve due to noisy fluctuations and those due to changes in line of sight. To model the noise, we used the difference between the unfiltered RSS curve and the filtered RSS curve. We shall refer to this difference as the empirical noise. We use the wavelet coefficients of the empirical noise as a threshold for classifying a significant jump in the filtered curve. We tried various thresholding procedures but the most successful procedure was one that classifies a wavelet coefficient at scale k of the filtered RSS curve as significant if it is larger than the $1-\alpha$ percentile of the empirical noise at scale k where $0<\alpha<1$. In our experiments we observed $\alpha = 0.25$ worked well. We believe with sufficiently many experiments, α can be chosen with cross validation. An illustration of the difference between the unfiltered and filtered wavelet coefficients is shown in Figure 14.

For a given jump over the time interval $[t_0, t_1]$, we adjust the path loss exponent in (1) proportional to the difference of the RSS measurements at t_1 and t_0. Letting ΔRSS denote this difference and letting $n = n(t)$, our dynamic model updates the path loss exponent according to the rule

$$n(t_1) = n(t_0) - \beta(\Delta RSS), \tag{2}$$

where β is some predetermined constant. For $t \in (t_0, t_1)$, we update the path loss exponent according to the proportion of the jump that has been traversed at measurement t. Elsewhere and away from jumps, the path loss exponent is constant. Apart from

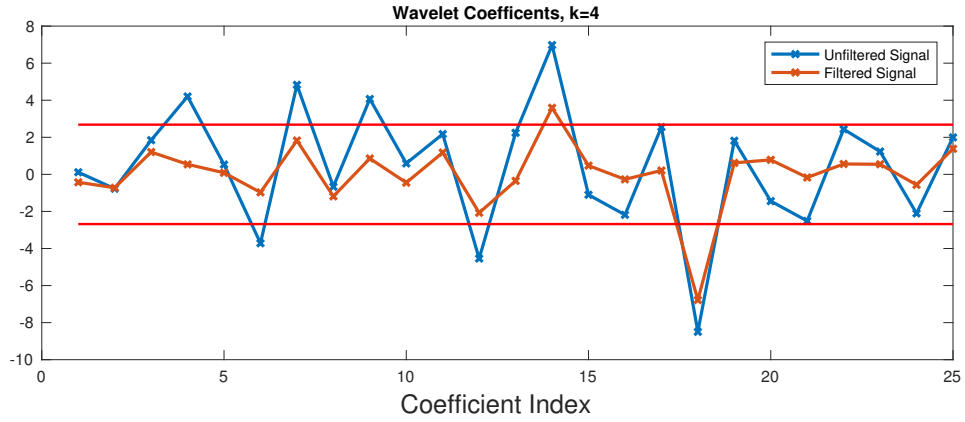


Fig. 14: Wavelet coefficients at scale $k = 4$ for the unfiltered and filtered RSS data generated from experiment 20180723_S3_Close_Open-1. Plotted in red is the threshold for classifying significant wavelet coefficients. Wavelet coefficients lying above the top red line or below the bottom red line are classified as jumps.

the parameter β, an initial path loss exponent must be chosen. Figure 16 illustrates how the path loss exponent changes with time for one of our experiments.

Judging the accuracy of our jump detection algorithm proved to be somewhat challenging. Experiments performed on July 23rd, 2018 were designed to carefully keep track of when the Raspberry Pi lost line of sight to the wireless access point. Nevertheless, due to a dynamic office environment with people walking around, furniture moving, and so on, it was difficult to record every possible line of sight obstruction. For the obstructions that we did record we found that our jump detection algorithm detected these jumps on most of our experiments.

Our experiments show that jump detection is sensitive to more than just the noise from measuring RSS. Jump detection is also affected by the global variance, or dispersion, of the RSS data. Comparing Figures 11 and 12 illustrate how dispersion varies as a function of line of sight. We believe rooms which have areas with line of sight will experience more dispersion as they are more likely to encounter line of sight obstructions from furniture or people in the room. However, for rooms with no areas of line of sight the dispersion is smaller because the wireless signal is already attenuated by passing through walls or furniture regardless of the position in the room. Perhaps unsurprisingly, when we normalize the RSS data by dividing by the standard deviation, jumps of large magnitude will remain pronounced while jumps of smaller magnitude will be diminished. Another factor affecting jump detection is the sampling rate. While most of our experiments were sampled at a rate of 2 Hz, some experiments experienced severe packet loss due to strong non-line of sight conditions. In these cases, jump detection behaved somewhat erratically. Looking at all of our experiments, we found that data with small fluctuations in rate and sampled at a rate of 1 Hz or greater generally admitted accurate jump detection. We found that experiments whose rate deviated below 1 Hz failed to detect some jumps, as the threshold for significant wavelet coefficients was significantly larger. This is because

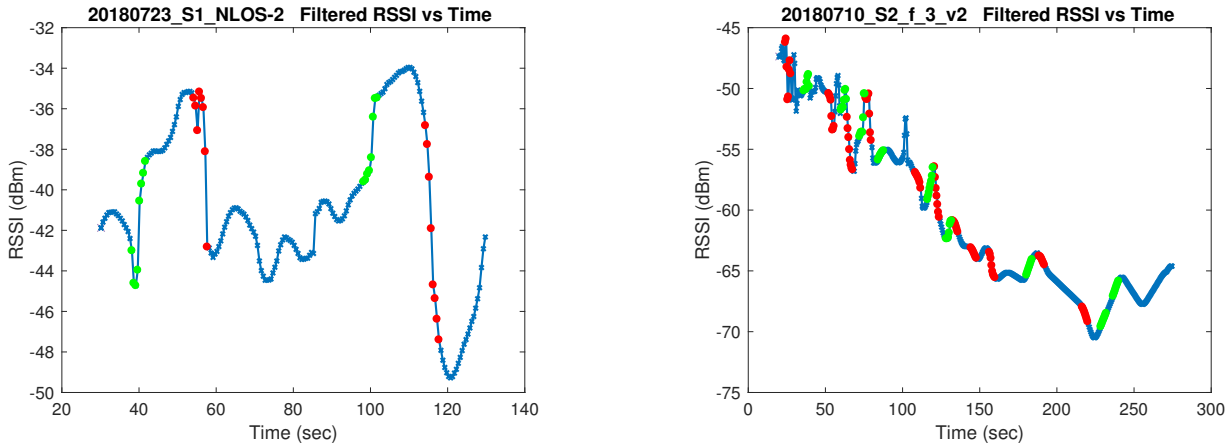


Fig. 15: Detected jumps for experiments 20180723_S1_NLOS-3 and 2018710_S2_f_3_v2 on the left and right, respectively. Standard deviation in consecutive time differences is 0.21 seconds and 2.44 seconds prior to resampling, respectively. Green dots represent detected sharp increases in RSS while red dots represent detected sharp decreases in RSS.

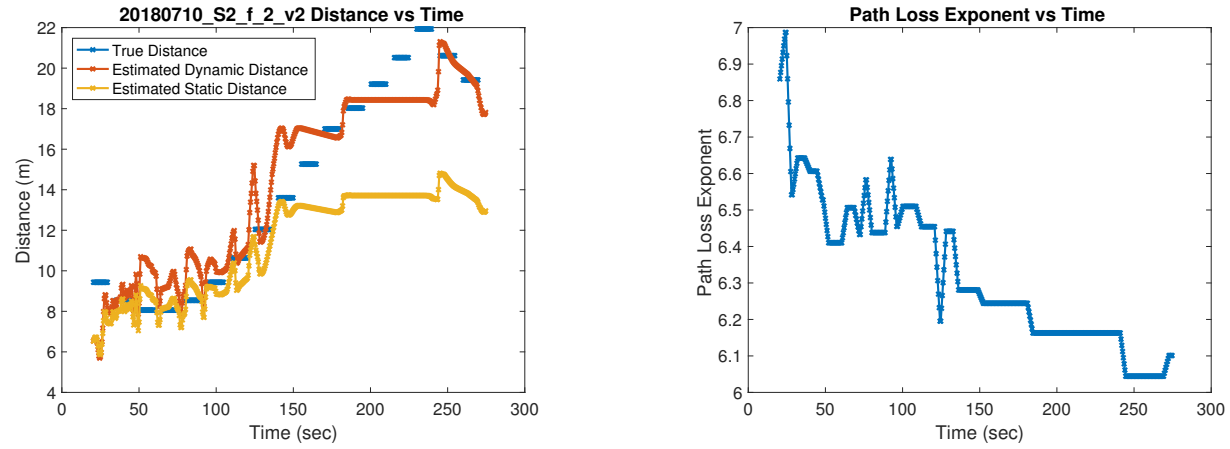


Fig. 16: Left: Estimated distances from the access point for the dynamic and static path loss models for experiment 2018710_S2_f_2_v2. The true distance from the access point is plotted in blue. Right: the path loss exponent as a function of time.

the empirical noise wavelet coefficients will be larger at higher frequencies, or smaller wavelet scales. Figure 15 illustrates the performance of jump detection on two of our experiments.

C. Jump Detection Performance

We choose our initial path loss exponent and rate of change parameter β by cross-validation with ℓ_2-loss. More formally, letting f_i and g_i denote the vectors of predicted

Region	σ_s	σ_d	n_{init}	β
S1	1.44m	1.40m	7.52	-0.06
S2	3.71m	2.19m	7.02	-0.04
S3	1.13m	0.95m	6.21	0.03

TABLE IV: Empirical risk for dynamic models trained to specific regions in our office along with the parameters for that model. Risks and parameters are given up to three significant figures.

and actual distances respectively each with length m_i, we define the risk σ to be

$$\sigma = \frac{1}{N}\sum_{i=1}^{N} L(f_i, g_i),$$
$$L(f_i, g_i) = \left(m_i^{-1} \sum_{j=1}^{m_i} (f_i(j) - g_i(j))^2 \right)^{1/2}$$

To compare the static path loss model to the dynamic path loss model, we first optimized the choice of path loss exponent for the static model to minimize σ with f_i being the predicted distances for the i^{th} experiment using the static path loss model. We then used this exponent as the initial path loss exponent for the dynamic model and chose β by cross validation. When we used all of our experiments for cross validation, the difference in risk on our validation set between the static path loss model and the dynamic path loss model was 0.7 millimeters. In other words, the average standard deviation from the true distance was 0.7 millimeters improved in the dynamic model. We believe this negligible improvement is due to the bias of our data set, as most of our experiments were taken in the room with the access point. Letting σ_s, σ_d denote the risk for the static and dynamic models respectively, we obtained the estimates as shown in Table IV for the risk of each model when training on each region individually.

We also compared our dynamic path loss model to the model that predicts distance from the regression line fitted to measurements from our experiments as plotted in 10. We found that both the static and dynamic path loss model outperformed the regression line model. Figure 17 illustrates how the cumulative distribution function (CDF) of the error in distance estimation grows for each model. As mentioned above, the static and dynamic models behave roughly the same. However, 95% of all errors made in our experiments by the static and dynamic models are less than 4.6 meters, whereas for the linear model we have 95% of all errors being less than 5.6 meters.

From our observations we are led to believe that jump detection offers modest improvements over the static path loss model on average. However, when there are transitions between regions of good wireless coverage and poor wireless coverage jump detection offers significant improvements. Indeed, S2 was the only region in our experiments that had such a transition and the dynamic model improved the risk by 1.52 meters. In larger settings such as factories or construction sites where

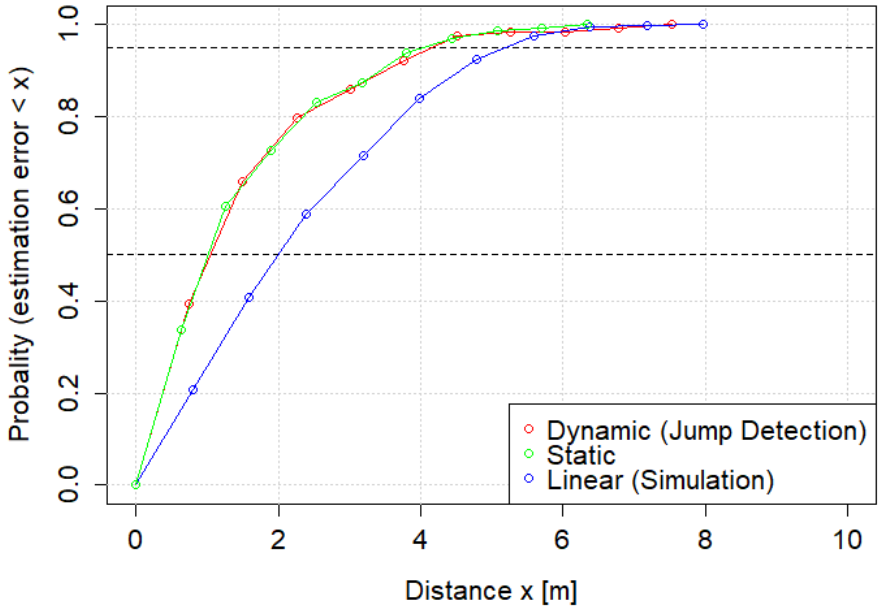


Fig. 17: CDF of the distance estimation error for the static, dynamic, and regression (linear) models. 50% of all errors made with dynamic and static path loss models are less than 1.6m, and 95% less than 4.2m. For the regression model, 50% of all errors made by fitting regression line are less than 2m. 95% less than 5.6m.

there is the potential for there to be many regions of varying signal strength dynamic path loss models could offer even better performance.

On a select few of our experiments the inability of our jump detection algorithm resulted in an increase in our localization error. Figure 18 illustrates an example of this. Unfortunately, this can cause error to propagate into future distance estimations. We are led to believe that the performance of our dynamic path loss model is strongly related to the performance of jump detection. As a result, we would encourage any efforts to improve the dynamic path loss model towards first improving jump detection. We leave this as future work.

Apart from this, we suspect there are many improvements that could be made to the dynamic model. Importantly, the dynamic model is sensitive to the jump detection itself. Failure to capture significant jumps could cause some error to accumulate in localization. While our software is able to detect jumps across various time scales, we only had time to analyze jump detection data for jumps detected at the 5 second time scale. Further, jump detection seems to be somewhat sensitive to the variance of the RSS curve. RSS data that contain sharp jumps on the order of 15 dBm to 20 dBm may cause jumps of smaller order go undetected. A dynamic scheme that analyzes distinct portions of RSS data and detects jump on each of these portions may prove to be more robust. Finally, since all of our experiments used only one access

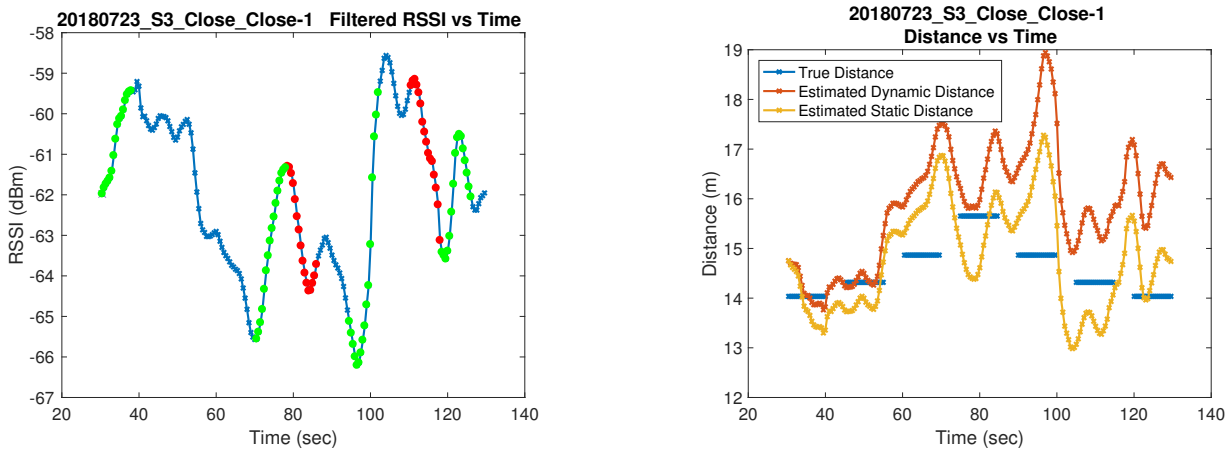


Fig. 18: Left: Filtered RSS data from experiment 20180723_S3_Close_Close-1. Observe that the jump(s) from 50-65 seconds are undetected, causing error to propagate in the distance estimation plotted on the right. Green dots represent detected sharp increases in RSS while red dots represent detected sharp decreases in RSS.

point we believe having multiple access points with coordinated jump detection could significantly improve localization performance. As an added benefit, having multiple access points allows one the freedom to design access point placement for localization purposes. Because of its shortcomings and rather modest improvements, we do not think the dynamic path loss model is effective for improving accuracy until additional improvements to jump detection are made. As a step in this direction we discuss below one of the above listed improvements, namely optimal access point placement.

VI. Optimal AP Placement

AP placement is important because it determines the efficiency of the user's equipment. Intelligent choices in AP placement allow the user to purchase fewer AP devices as well as experience better Wi-Fi coverage within their building. It is difficult to define a single optimal placement because there are several different ways to measure the optimality of a placement and it is unlikely that one placement could minimizes across all of these objectives. Some example objectives include minimizing the area of the building with low coverage, maximizing average signal strength, and minimizing the overlap in coverage area between two APs to reduce redundancy.

There are many existing papers that have created programs to suggest AP placement. The main techniques we encountered in our literature review were using a genetic algorithm, formulating an integer program, or using techniques from game theory. Some have a single objective function, while others use a combination of multiple objectives so the user can specify the priority level of different performance measurements [2]. Most of these programs involve using a 2D or 3D floor plan and RSS simulation software to do their calculations. For more detailed information refer to [2], [12], [13].

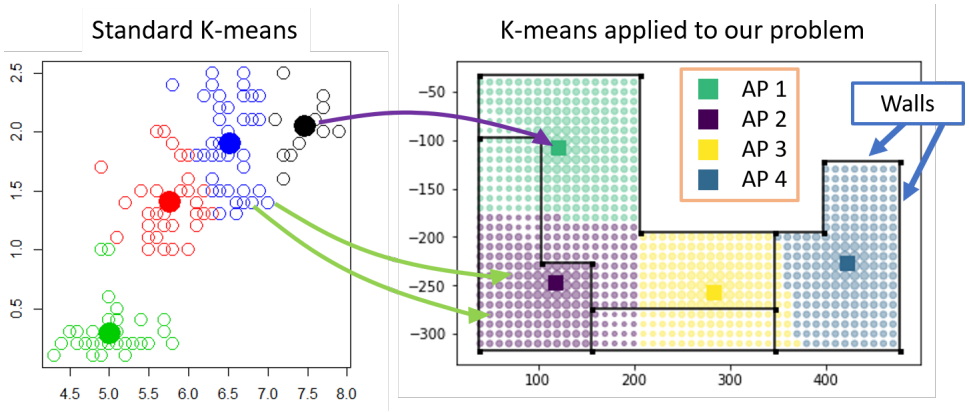


Fig. 19: A comparison of the traditional application of k-means compared to how we use it in our problem. Usually data is a collection of measurements (often multi-dimensional) as seen in the scatter plot on the left, while in our problem data only has x and y coordinates to signify the data point's location within the building.

Like the existing methods for AP placement, we use a floor plan and simulation software to guide our main program. However, for the main program we wanted to find a direction that had not been explored yet in the literature and has the potential to be a lightweight yet accurate program. In the end we decided to use a k-means algorithm to look for intelligent AP placement. For a brief overview of the k-means algorithm see Section II-B.

To apply the algorithm to our current problem, we let the cluster centers represent APs and the data points represent grid points or a specified collection of points in a building. For a visual comparison see Figure 19). We chose this algorithm because k-means finds fairly evenly-spaced clusters to minimize distance to the nearest center. In the context of our problem this means that k-means will help us minimize the distance from each point in the building to the nearest AP. RSS is highly affected by distance so we hope to get a good result by placing APs in locations corresponding to cluster centers. We also think this is a good method because it should scale well to larger buildings with many APs. NEC has showed interest in AP placement in the factory or construction site setting, and we want our method to still be a viable option for this case. While the computational cost will grow with increased building size or number of APs the algorithm's complexity remains the same, and the individual computations could be computed simultaneously using multiple processors.

A. Data Generation

In order to use the k-means algorithm, we need to make a few modifications to our floor plan processing code introduced in Section IV-A. First, from our building plan we need to extract some selection of reference points so they can be clustered. We took measurements at equally-spaced grid points from the inside of the building. This gives all areas of the building an equal priority. The user may add or delete some data points to prioritize one area over another. Because our data selection method is quite

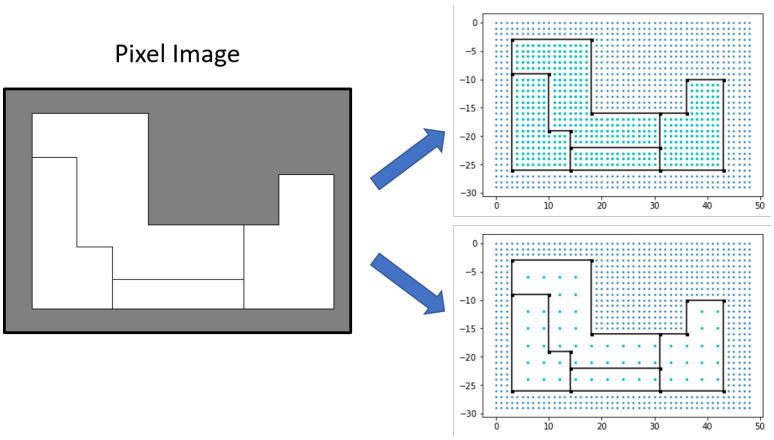


Fig. 20: Left: inputted pixel image. Right: the generated outside-the-building points(dark blue) and our k-means dataset made of interior points(light blue). In order for the individual pixel outside points to be visible, a miniature version of our demo floor plan was used. The top right image has a spacing of one, i.e. every grid point in the building that is not a wall is selected as a data point. The bottom right image has a higher spacing, where only one data point is selected per 3×3 pixel grid.

simple, we thought the user should have at least one parameter of flexibility, namely, the amount of data to collect. The size of building floor plans could vary greatly and using our RSS simulation program can get expensive if evaluated on a large dataset. This is because for one RSS k-means iteration, every data point has to have the RSS calculated for each AP. Therefore, we enable the user to specify how many pixels are in between each data point, giving them some control of computation time and how precise their results will be.

The other feature we need in our code is a method to detect if a point is outside of the building. This is important for knowing which data points to collect, but also for testing if a cluster center has been assigned to a point outside of the building. We want to avoid this case, as an AP unit meant to transmit indoor WiFi is not usually put outside. To address both of these needs, we create an array that contains the coordinates of all gridpoints that are outside of the building. This is done by adding a third color to our original pixel floor plan: all grey pixels are assumed to be infeasible for AP placement, such as points outside of the building. It is important to note that these pixels will also not be considered for data point selection so the signal quality of these areas do not influence the model. Figure 20 depicts an example of the interior data points and exterior points datasets generated by our code.

B. Distance Metrics

In each iteration of the k-means algorithm, the "distance" from each point to each cluster center is measured in order to find the closest center. Traditionally, the distance is measured as the Euclidean distance and algorithm minimizes the sum of the squares of the Euclidean distances from points to the nearest AP:

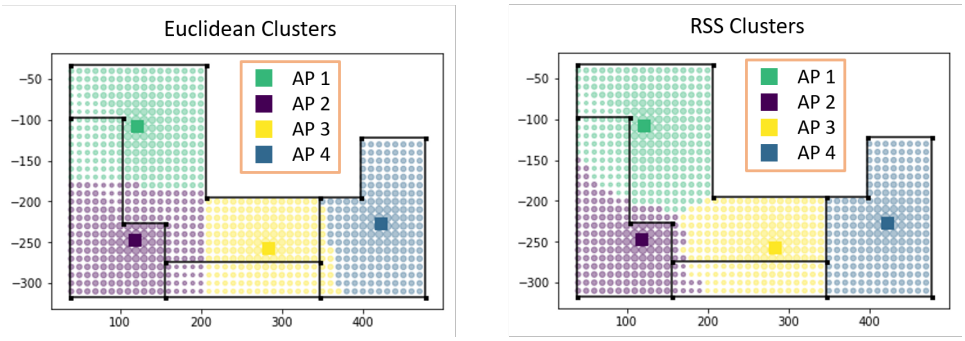


Fig. 21: Both images have four APs in the same location of the same building. On the left, the data points are colored based on which AP has the smallest Euclidean distance from that point. On the right, the color of a data point matches the AP that has the strongest signal strength at that data point. This means that given the current AP locations, the two images show how the k-means would assign clusters based on Euclidean distance and RSS respectively.

$$\min_{APplacement} \sum_{x_i \in Data} |AP_{nearest} - x_i|^2.$$

In our case we are interested in getting the best signal strength throughout the building. The signal strength-related objective function that is most compatible with the k-means objective function above would be

$$\max_{APplacement} \sum_{x_i \in Data} RSS_{AP_{best}}(x_i),$$

i.e. maximizing the sum of the best signal strength received by the data. These two measurements are undeniably related, however they should not be considered interchangeable. Figure 21 gives us an illustration of the differences between the two measurements and why Euclidean Distance may not be the best distance metric for our problem. After assigning the data to clusters as done in the figure, the next k-means iteration would move the APs to the centers of their respective clusters. We can see that our AP locations in the next iteration would be significantly different if we chose to use RSS clusters instead. Because clustering based on RSS values more accurately reflects our actual objective of optimizing signal strength and will likely give us an additional set of AP placements to consider, we apply multiple k-means methods with different metrics.

Using the two distance metrics we created three distinct methods. The first is the Euclidean Method, which simply uses the original Euclidean distance k-means algorithm. The second is the RSS method that only uses simulated RSS to determine clustering. The third method is a combination of the two. The Hybrid method starts with running the Euclidean k-means algorithm to convergence, then using the final clusters as initial input clusters it runs the RSS k-means algorithm.

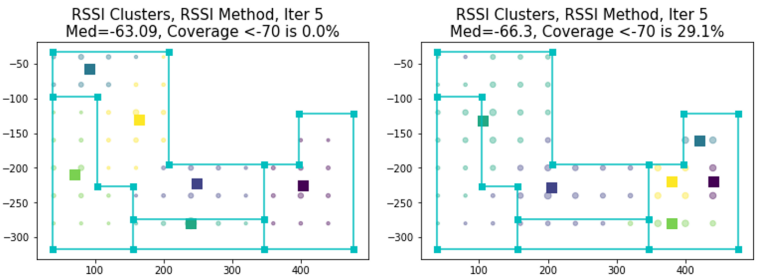


Fig. 22: Left: a lucky choice of initial centers. A center was placed low enough to stay in the bottom center room, whereas in other experiments it did not have an AP and therefore usually had low coverage. Right: an unlucky choice of centers. Four of the six APs get put in the same room and their signal is not strong enough to go through the wall and capture new points for their clusters. Unable to get any APs moved out of the room, the rest of the building is covered by only two APs. Despite having the same number of clusters, the two choices in initial clusters have resulted in dramatically different performance with 0% low coverage and 29.1% low coverage respectively.

C. Performance

Starting with our demo floor plan, we ran a number of experiments to test the behaviour and performance of our model. Because the k-means algorithm is deterministic, new initial centers are needed to get different final clusters. To solve this we generate the desired number of initial centers randomly. If at any iteration there are APs placed at infeasible locations, the center will be moved to the closest data point and will therefore be feasible.

To measure the performance of our methods we used three qualitative metrics: median RSS of the data points, average RSS of the data points and percent of the data points where the RSS is below a user-defined threshold, i.e. the percent of the building that has "low coverage." In our experiments we used -70 dBm as our threshold, as our experiments in Section III did not yield any RSS lower than -80 dBm. We include three performance measurements because the user's priorities and definition of optimal are likely to vary.

After running the program a few times it became clear that there could be very poor choices of initial clusters as well as very good initial choices. This is especially true for the methods using RSS as shown in Figure 22. Because our initial centers are random, this simply means we need to run many trials of the same experiment to ensure a good variety of starting points.

Once we had seen some case-by-case experiments we constructed a larger experiment to examine trends. We estimated that the demo building would need around three to six clusters, and specified a spacing range of 10 to 50. Here spacing indicates the number of pixel units between indoor data points. To evenly distribute between the different cluster numbers and spacing values we created 40 sets of initial centers with 10 sets of each cluster number. These were matched with the data sets of spacing levels 10, 20, 30, 40, and 50, and then inputted into each of the three methods, for a

total of 600 experiments. The graphs of our three performance metrics are shown in Figure 23. We will discuss in more detail in Section VI-D, but we know that having a sparser dataset has a higher potential for error than a dense dataset. The sparse data results are not known to yield inaccurate RSS predictions and coverage, but the more fine-mesh datasets are believed to be more reliable in terms of performance matching the real-life coverage of the entire building. For this reason we have made the results with higher spacing levels more translucent so they have less influence on conclusions.

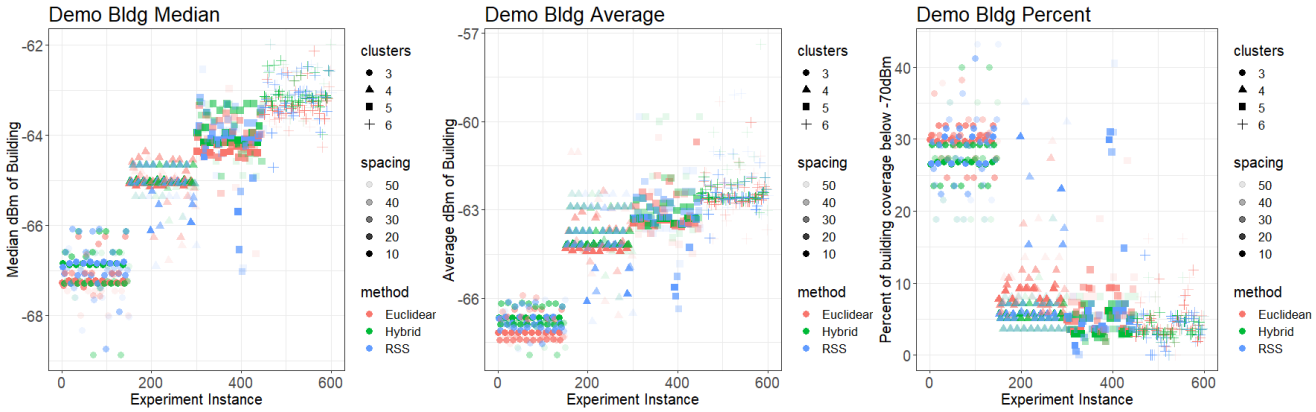


Fig. 23: Median, Average, and Percent with low coverage for the demo building experiments. High values are desirable for Median and Average, low values as close to 0% as possible are desirable for Percent. The number of clusters is increasing from left to right for each graph.

As we can see from the graphs, there are many repeated values, especially for the three and four cluster trials. This suggests that the same solution is achieved many times, which is quite possible since there are only a couple of clusters and the demo building is fairly simple. We can also see that there are several trials where the solution was significantly worse than the majority. Most of these instances were using the RSS method, so it is quite possible that a similar situation happened as the second image in Figure 22.

Looking at the trends in the graph, we find unsurprisingly that increasing the number of clusters generally improves all three performance metrics. However the improvement in Average and Percent seems to slow down and plateau at 6 clusters. This is inevitable for the percent low coverage, since the percent is quickly approaching zero and cannot go any lower. Even at only five clusters there is an RSS instance that achieves 0% low coverage.

As for comparing methods, we note that the bulk of the Euclidean trials fall visibly behind in performance compared to the Hybrid and RSS methods. However the advantages of the Hybrid method over the RSS method are difficult to judge. The RSS is not as consistent as it has most of the worst-case trials, but that does not necessarily put it behind the Hybrid method. For our problem the worst-case performance is not

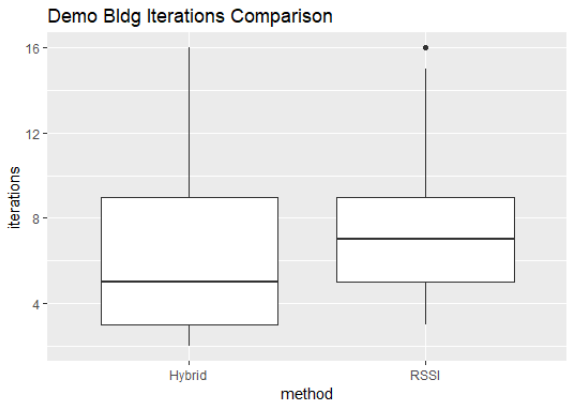


Fig. 24: Boxplots for the number of RSS iterations needed for the Hybrid and RSS methods. The iteration cutoff was 15, and 16 was reported if the algorithm had not converged by the 15th iteration. The number of iterations for the Hybrid method is considered to be the number of RSS iterations after Euclidean convergence since Euclidean iterations are inexpensive.

overly important since all we need is one good AP placement that meets the user's needs. In short, we get to keep the best placement we find, so a method is as good as its best-case placement. With this in mind, most of the best performances seem to be in favor of the RSS method. It is important to note we might pick a placement with the best percent but find that it has an unremarkable average and median. This is where it comes down to the user's priorities. If only one of the performance metrics has the most importance, then the optimal placement to suit the user's needs can be easily decided. If multiple measures are important, then it is necessary to consider all three scores as a whole. This cannot be done with our Figure 23 graphs and would require a more detailed examination of the scores of the trials.

Another factor that impacts the value of the Hybrid and RSS methods is the computation time. For our demo building the computations were fairly quick at one iteration costing well under one minute. For larger problems one iteration could take a substantial amount of time, so having a method that requires fewer iterations is desirable. In Figure 24 we see that the Hybrid method generally requires fewer iterations, likely due to the fact that its initial centers are already somewhat spread evenly from the Euclidean method. But despite this initial centers advantage, the Hybrid and RSS methods have the same top quartile cutoff of 9 iterations.

We now have a good first impression of how these methods behave, so next we want to apply our methods to a more complicated floor plan. Using the AIMR 4th floor plan used in simulations, we conducted a similar large-scale experiment like we did for the demo. An example of one of these experiments can be seen in Figure 25 The number of clusters and spacing levels were scaled up to range from 10 to 30 clusters and 40 to 100 pixels. The distribution between the groups was less even than the previous experiment because an RSS iteration for this larger problem required on the order of 20 minutes so the experiment was done over several sessions. Additionally,

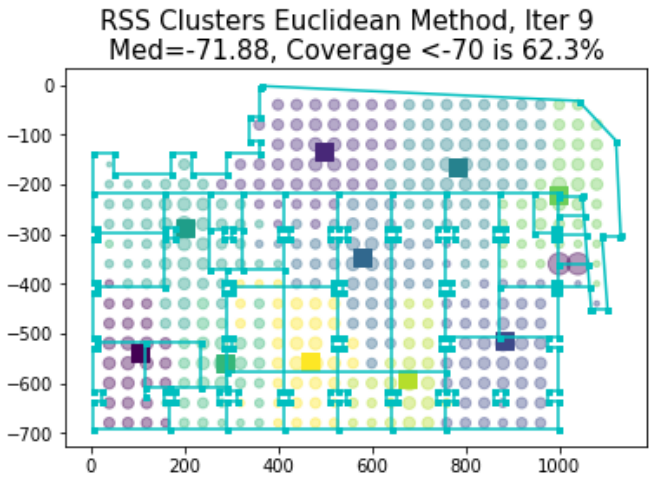


Fig. 25: An example of one iteration of k-means for the fourth floor of AIMR with 10 APs/clusters.

we noticed in our demo building experiments that there were many cases when the final performance scores of the Hybrid and RSS methods were not as good as those in previous iterations. For this reason we decided to monitor the percent low coverage for each iteration and reported the best percent score in addition to the final score. We decided to focus on percent rather than median or average since it is usually important to get decent coverage throughout the building.

Performance graphs like the ones in Figure 23 are displayed in Figure 26. To increase the fidelity of our experiments, we focused on getting trials with fairly dense data. Figure 27 shows the iteration comparison between the Hybrid and RSS methods.

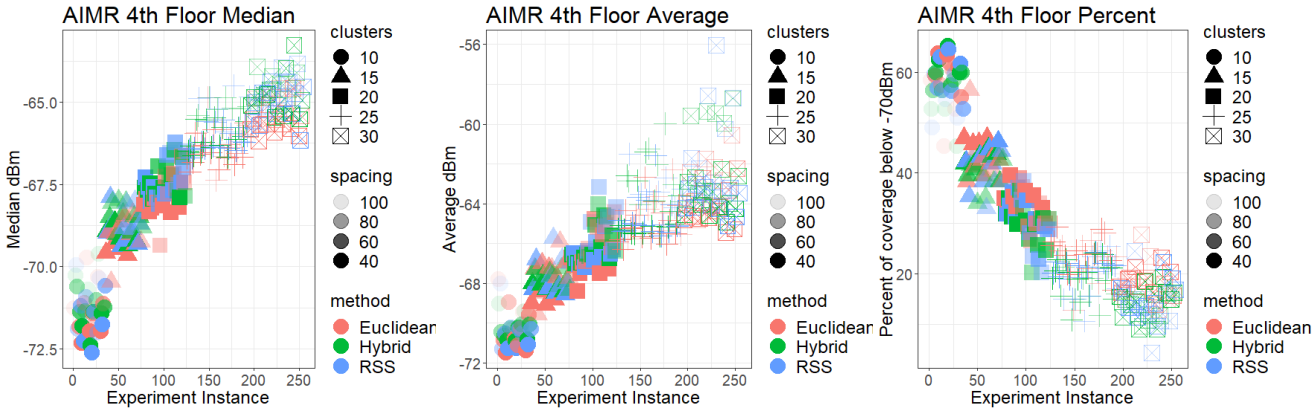


Fig. 26: Performance scores for the AIMR 4th floor.

Even though there are significant differences in the size, shape, and design of the demo and AIMR 4th floor plans, the overall trends observed for the methods still remain the same. The most notable difference is the lack of repeated values, but since

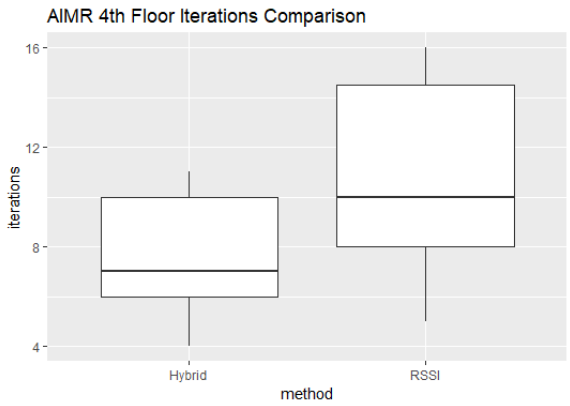


Fig. 27: Boxplots for the number of RSS iterations needed for the Hybrid and RSS methods. The iteration cutoff was 10 and 15 for the methods respectively, and 11 and 16 were reported if the algorithm had not converged by the cutoff iteration. The number of iterations for the Hybrid method is considered to be the number of RSS iterations after Euclidean convergence since Euclidean iterations are inexpensive.

our problem is more complicated it would make sense that repeatedly finding the same local minimum would be much less likely. We still observe that the performance scores improve with increasing the number of clusters but begin to plateau at the higher values. As before, we see that the Hybrid and RSS methods out-perform the Euclidean method but neither method seems significantly better than the other. Most of the best placements appear to belong to the RSS method, but the Hybrid method usually has a close second place and according to Figure 27 the Hybrid method usually does not need as many iterations.

The additional information that we did not have for the demo building is the best percent achieved rather than just the final percent, so we include the comparison in Figure 28. There are not any significant changes and most of the points that changed seem to have higher spacing levels. However, there were some small improvements in the fine-spaced points that would lead to new optimal placement choices. So while improvements were minor, they are still improvements and it is computationally very cheap to keep track of performance over the iterations.

In addition to improving the best performance scores, keeping track of the performance and centers at each iteration has other advantages as well. Storing information on every iteration will increase the choice portfolio for the user which is useful when the user has a more nuanced set of priorities as opposed to only caring about low coverage percent, for example. Having a more complete set of information can also give the user a thorough understanding of the achievable performance levels of the given number of APs. This can help them make an informed decision on the number of APs to purchase for their building.

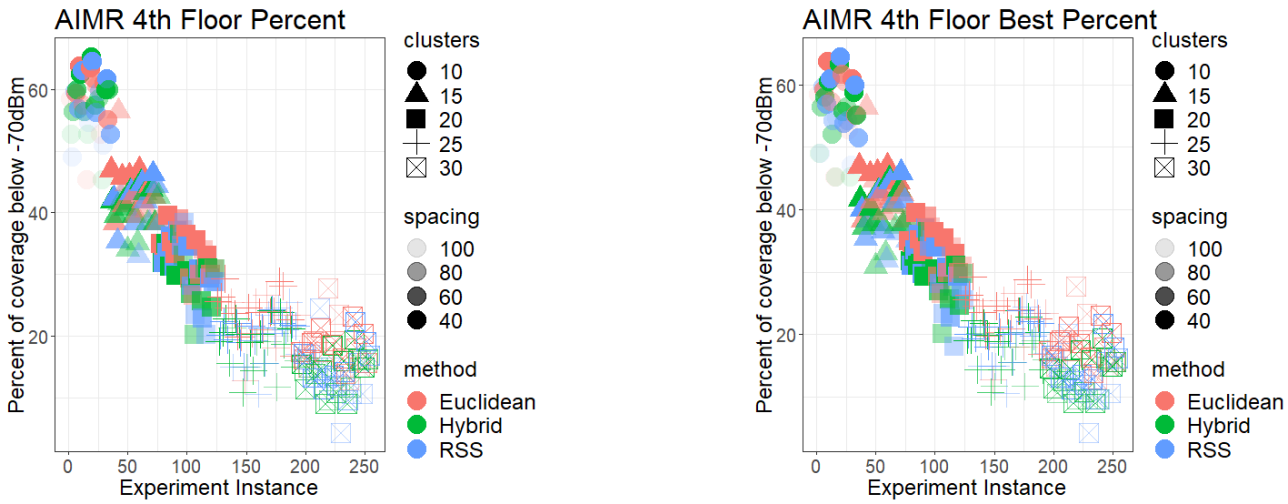


Fig. 28: A comparison of the final percent against the best percent in the AIMR 4th floor experiments.

D. Sparsity Levels of the Dataset

We mentioned before that using a sparser dataset can save a lot of computation time, but we also need to know how much we can trust the results. We ran a short experiment where we performed our k-means algorithm on sparse datasets for the AIMR 4th floor. We collected the reported performance data and AP locations, and then tested the performance of the clusters using a denser data set with spacing set to 40 pixels. We are not able to calculate the actual performance of this placement, since we were limited to one access point. Therefore, we used our dense data set as an approximation. We tested the results for 10 APs and 20 APs at spacing levels 80 and 100. Graphs of the results are displayed in Figure 29.

From the graphs we can see that most of the experiments with sparse datasets overestimated the performance of their AP placements. This is especially true for the sparser of the two spacing levels, as every single one of the 100-pixel spacing experiments overestimated performance. Meanwhile, the 80-pixel spacing experiments mostly hovered around the cutoff line and and some even underestimated performance slightly. We also see the magnitude of the overestimations are much higher for the 100-pixel spacing compared to 80-pixel.

If we consider the actual range of performance for these sparse datasets found in Figure 29 and compare with the approximate range of performance of our dense data sets in Figure 26, we see that the sparse datasets are no longer outperforming the dense datasets. Therefore we conclude that even though they are not as expensive to generate, we should avoid calculating AP placements with very sparse datasets. Given our results, we would say that for our office building 100-pixels is too sparse, 80-pixels is on the borderline of acceptable accuracy, and spacing levels below 80 would probably be acceptable.

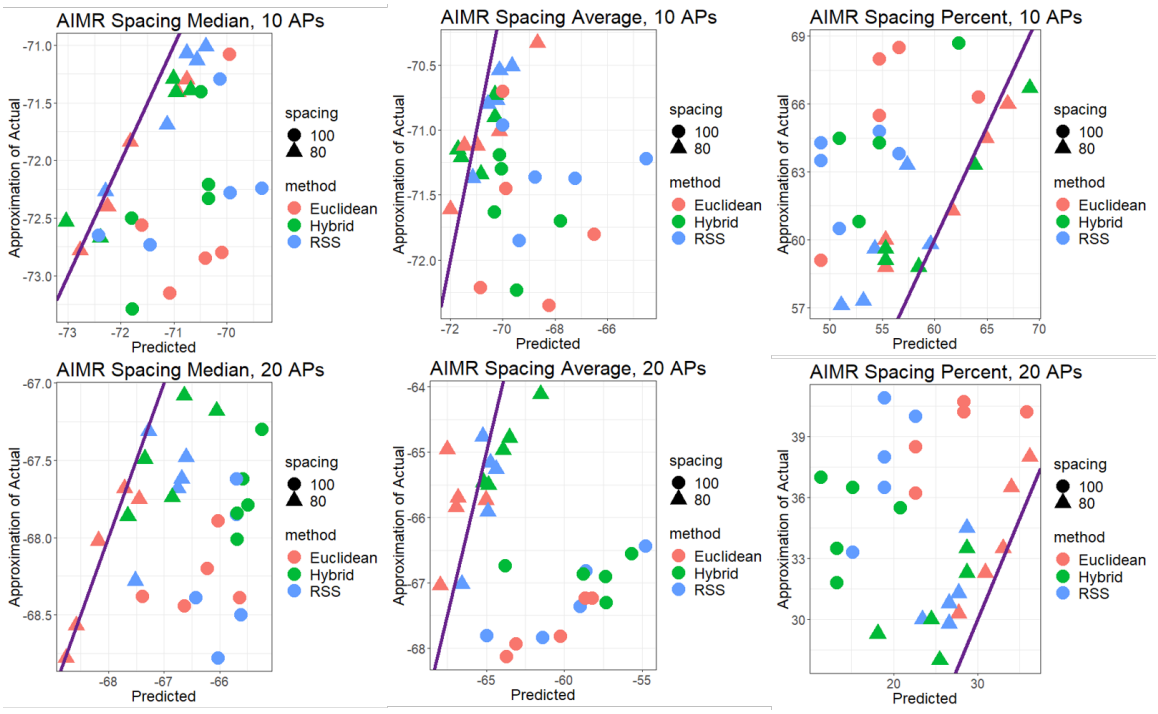


Fig. 29: A comparison of the reported performance of clusters found with a sparse dataset against an approximation of the actual performance. Each of the three performance metrics are shown, with the first row of graphs having 10 APs and the second row having 20. For the Median and Average graphs, any point below the purple line indicates that the reported performance overestimated the quality of the actual performance. For the Percent graphs, any point above the line means the prediction underestimated the percent of the building that has low coverage.

We suspect this discrepancy in performance reporting is due to the representation of the smaller rooms in the building. Smaller rooms are less likely to have an AP placed in them so most of these rooms will have lower coverage compared to the rest of the building. However, with sparser datasets like our 100-pixel spacing, many of the smaller rooms in the building have zero or one data point for that room. This means these low coverage areas are being underrepresented in performance calculations so the performance is overestimated. For this reason we suggest using a spacing level that is dense enough to give decent representation to all of the rooms in the building.

E. Walkthrough of AP Placement Selection

To bring together the results of our analysis for our k-means methods, we outline the process of AP Placement selection that we would recommend to the user.

First the user must produce a pixel image of their floor plan like the left image in Figure 20. Then the user must determine a spacing level for their data that will sufficiently capture the features of the rooms in the building. Ideally the data should

be dense enough that the smallest rooms, assuming they need to have WiFi coverage, should have at least a few data points within the room.

Once this setup is complete, the user can begin running simulations. If the user has not decided how many APs to use, we suggest testing a wide range of APs like we do in Figure 26. These simulations should be done with the Euclidean method, since it is inexpensive to do many iterations for each number of APs being tested. During these simulations, performance data will be accumulated. Once the initial experiments are complete, the user can review a summary of the performance, possibly in the form of a graph like in Figure 26. The performance levels found by the Euclidean method will serve as a lower bound for the performance since we can expect to find better placements from the Hybrid and RSS methods. With this in mind, the user can decide how many APs to use by selecting the number that produced results sufficiently close to their performance requirements. For example, suppose the user was placing APs for the AIMR 4th floor and needed a placement yielding a median coverage of at least -68 dBm. From the Euclidean method results in Figure 26 they would see that having 20 APs would meet this requirement since the Euclidean results achieve this already. However, the results for having just 15 APs are quite close to -68 dBm and a suitable placement is likely possible with the other methods. For this reason the user might choose to go with 15 APs. Because not all numbers of APs were tested in our experiments, the user also might benefit from testing 13 or 14 APs as well.

Now that the number of APs has been selected or narrowed down, the user can use the more expensive Hybrid and RSS methods to produce placement candidates. After the candidates have been found, hopefully one or more of them fulfill the user's coverage requirements. If there are several options, other the performance scores could be examined to choose the best candidate defined by the user's priorities. In the case that none of the candidates are satisfactory, the Hybrid and RSS methods could be run more times if the current candidates are very close to fulfilling the requirements. If there is still a sizable gap between performance and expectation, the user can increase the number of APs and create more candidates with the Hybrid and RSS methods.

F. Human-Generated Placements

One of the most important parts of analyzing our method is determining its potential for finding good AP placements. Ideally, this would involve comparing the performance of our method with existing models or with AP placement and coverage data. Unfortunately we do not have access to any models or data, so the best we could do is compare with placements made by our team members. To get a few different types and qualities of human-generated placements, Eric spent just a few minutes designing a placement for 10 APs and one for 15 APs, and Shizuki spent around 20 minutes designing a 10 AP placement. The performance values are shown in Table V.

Comparing these values with the performance of our k-means methods in Figure 26, we see that the placements made by our team are on the lower end of the performance

Placement	Median	Average	Percent
Eric's 10-AP set	-72.4	-71.87	66.3
Shizuki's 10-AP set	-72.07	-71.0	62.0
Eric's 15-AP set	-70.06	-68.58	50.4

TABLE V: Performance of placements designed by our team. Performance was calculated using our 40-pixel spacing dataset.

range for 10 and 20 APs. Their placements were not worse than all of the k-means placements, but it might require many attempts for a person to create a placement that competes with the best placements we can find through several iterations of k-means. This would be time consuming for the person deciding placements so we claim our k-means methods are preferable to deciding placements by hand.

G. Summary and Future Directions

With our initial analysis we have found our methods to be fairly robust and consistent in terms of trends over several trials. The clear third place method is Euclidean, though in terms of computational cost it is by far the cheapest so this method has its merits as well. There is not a clear winner between the Hybrid and RSS methods. The Hybrid method needs fewer iterations but the RSS method often produces slightly better solutions in terms of considering one metric at a time. The user can use one or more of these methods for their building floor plan to get a good starting estimate of the number of APs required, where to place them, and how good coverage will be with that placement. We were not able to compare our performance with existing methods, but our methods did well against placements designed by hand.

The next steps to take for our methods involve three main components: further analyzing performance, improving computation runtime, and trying modifications to improve performance. As mentioned before, the main performance analysis that remains to be done is a comparison with other models. Testing a wider variety could also be beneficial to understand the performance of our methods. To improve runtime, a first step would be to parallelize our code to run calculations simultaneously. It would also cut our computation time if the RSS calculation was filtered so that for instance, the RSS of a data point would only be calculated for nearby APs instead of for every AP.

We have come up with a few ideas that could potentially improve performance if added to our model. First, we noticed that the large open space at the top of the AIMR floor plan was usually assigned at least a few APs. This is probably more than necessary since most of the points have line of sight to all of the APs in that area. Meanwhile, a lot of the smaller rooms have poor coverage because there is no line of sight to an AP. Using our k-means algorithm it is easy to prioritize these rooms for the Hybrid method and RSS method where RSS coverage is calculated at every iteration. During a k-means iteration, each AP location is updated to be the center of the points assigned to its cluster. This is simply the average of the point locations with uniform weighting, so changing the weighting of the points will change the new AP locations. If we were to weight points with poor coverage more heavily than

points with decent coverage, the AP would move closer to these poor coverage areas and hopefully improve the signal strength of those areas. We believe that using this weighted k-means algorithm might help improve coverage, especially the percent of the building with low coverage. A more direct but more complicated idea with the same goal is calculating if each data point has line of sight to the AP in its cluster and assigning a higher weight if it does not.

Our other idea involves breaking the building into a subproblem for each AP after implementing k-means. With the AP locations and clusters found by k-means, each subproblem would aim to slightly shift an AP so that the coverage is maximized over just the points in its current cluster. This could be done by testing a selection of points near the current AP location and choosing the location with the best signal for that cluster. The reason we think this might improve performance is because the center point of a collection of data minimizes distance to all of the points but does not necessarily maximize coverage. Admittedly this type of method would likely be a part of fine tuning rather than dramatically improving the placement. Optimizing each AP would also take some time, but depending on how many candidate placements are tested for each AP, it would probably cost about the same as a couple of RSS or Hybrid iterations.

VII. Conclusion

Our findings corroborate much of what the previous literature on indoor localization has confirmed. Namely, the complications due to multipath propagation, diffraction, attenuation, and non-line of sight pose challenges for distance estimation. Nevertheless, our simulations and experiments seem to suggest that multipath propagation effects are limited to paths which admit up to one reflection. This simplifies multipath modelling dramatically. Another important finding is that incorporating global features of the RSS data appears to greatly improve path loss models in some cases. We believe that sharp discontinuities are one of many statistics that could be used to improve distance estimation. Our experiments seem to suggest that the variance of the RSS data is correlated with the severity of the line of sight. Measuring how sensitive this relationship is could help determine initial path loss exponents and could further reduce the amount of calibration needed for path loss models. Lastly, having simulations that qualitatively mimic measured RSS allows one to optimally place access points based on a variety of metrics. This has the potential to improve indoor localization and result in cost savings in purchasing access points without sacrificing signal coverage. We leave these improvments along with more quantitative analysis of these methods for future research.

References

[1] Editorial Board. *Indoor Environmental Plan*. Number v. 22 in Architectural Studies Large. Shokokusha, 1969.

[2] Abderrahim Chariete, Valéry Guillet, and Jean-Yves Thiriet. Optimization of wi-fi access point placement in an indoor environment. In *New Technologies for Distributed Systems (NOTERE), 2016 13th International Conference on*, pages 1–6. IEEE, 2016.

[3] Ingrid Daubechies. *Ten lectures on wavelets*, volume 61. Siam, 1992.

[4] Pavel Davidson and Robert Piché. A survey of selected indoor positioning methods for smartphones. *IEEE Communications Surveys & Tutorials*, 19(2):1347–1370, 2017.

[5] David L Donoho and Jain M Johnstone. Ideal spatial adaptation by wavelet shrinkage. *biometrika*, 81(3):425–455, 1994.
[6] R.A Fisher. Annals of eugenics, 1936.
[7] Steven J Fortune, David M Gay, Brian W Kernighan, Orlando Landron, Reinaldo A Valenzuela, and Margaret H Wright. Wise design of indoor wireless systems: practical computation and optimization. *Computing in Science & Engineering*, 1:58–68, 1995.
[8] Katsuyuki Haneda, Lei Tian, Henrik Asplund, Jian Li, Yi Wang, David Steer, Clara Li, Tommaso Balercia, Sunguk Lee, YoungSuk Kim, et al. Indoor 5g 3gpp-like channel models for office and shopping mall environments. In *Communications Workshops (ICC), 2016 IEEE International Conference on*, pages 694–699. IEEE, 2016.
[9] Elliott Kaplan and Christopher Hegarty. *Understanding GPS: principles and applications*. Artech house, 2005.
[10] Moayeri Greenstein Kaya, Chizhik and Valensuela. Emitter localization and visualization (elvis): A backward ray tracing algorithm for locating emitters. *Information Sciences and Systems 2007 CISS'07. 41st Annual Conference*, 2007.
[11] Hui Liu, Houshang Darabi, Pat Banerjee, and Jing Liu. Survey of wireless indoor positioning techniques and systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 37(6):1067–1080, 2007.
[12] Alan Mc Gibney, Martin Klepal, and Dirk Pesch. A wireless local area network modeling tool for scalable indoor access point placement optimization. In *Proceedings of the 2010 Spring Simulation Multiconference*, page 163. Society for Computer Simulation International, 2010.
[13] Nila Feby Puspitasari, Hanif Al Fatta, and Ferry Wahyu Wibowo. Layout optimization of wireless access point placement using greedy and simulated annealing algorithms. *network*, 2:3, 2016.
[14] Silex. Wlan and communication distance(2). http://www.silex.jp/blog/wireless/2011/12/post.html.
[15] Jukka Talvitie, Markku Renfors, and Elena Simona Lohan. Distance-based interpolation and extrapolation methods for rss-based localization with indoor wireless signals. *IEEE transactions on vehicular technology*, 64(4):1340–1353, 2015.
[16] Tanoi and Kawase. A study on electromagnetic shielding characteristic of reinforced concrete wall. *Tokyu Kensetsu technical report*, 37:61–64, 2011.