Protected Region Radio Map Estimation

By

JONATHAN RYAN TIVALD

THESIS

Submitted in partial satisfaction of the requirements for the degree of

MASTERS OF SCIENCE

in

ELECTRICAL ENGINEERING

in the

OFFICE OF GRADUATE STUDIES

of the

UNIVERSITY OF CALIFORNIA

DAVIS

Approved:

PROFESSOR ZHI DING

PROFESSOR KHALED ABDEL-GHAFFAR

PROFESSOR SOHEIL GHIASI

Committee in Charge

2022

To my wife and family who supported and believed in me.

Contents

List of Figures

List of Tables

Abstract

Protected Region Radio Map Estimation

Background. Passive radio frequency (RF) sensors and receivers are highly vulnerable to unintended radio interference from deployment of active RF transmitters in nearby areas of service. Often, these RF receivers may also be susceptible to overloading damages. High likelihood scenarios of overloading damages include ultra-sensitive receivers that cannot afford front-end protection, or receivers deployed while powered down without the ability to measure the environment before powering on. It is often costly to measure RF signal strength and assess potential interference over wide urban/suburban areas among various building structures and complex terrains. Moreover, these passive RF sensors and receivers are sometimes deployed in locations that are difficult to access and to measure radio signal strength from new RF transmissions or those under planning. Consequently, it is important in the service planning stage to estimate a wide area radio map from only limited RF measurement at locations of convenience.

Objective. We propose that a network of cheap and robust RF receivers may be sparsely deployed in a geographical region to estimate a completed radio map. After receiving power measurements from the sparse network of RXs, several different estimation methods may be applied to reconstruct the region's radio map. These estimation methods may be in the form of kernels, random processes, basis functions, and Machine Learning (ML) algorithms. We aim to provide a certain level of confidence in multiple estimation methods that may be used for estimating a completed radio map.

Results. Many of the interpolation methods produced favorable results when estimating a radio map. The Inverse Distance Weighting (IDW) algorithm performed the best overall due to being one of the most accurate estimators, having the fastest processing time, and robust performance with system parameter selection. Overall, the Machine Learning (ML) algorithms processed much faster than the average interpolation method, but performed worse on average. Iterative Shrinkage and Thresholding Algorithm (ISTA) Net performed the best due to estimating the most accurate radio maps.

Acknowledgments

I would like to thank my advisor Dr.Zhi Ding and his team of BRATs (Broadband Radio Access Technologies Laboratory) for supporting my research. I would also like to thank my good friend and colleague Scott Saito for discussing insightful ideas with me regarding my research. Lastly, I would like to thank Applied Physics Lab (APL) of Johns Hopkins University (JHU) and Aerial Experimentation and Research Platform for Advanced Wireless (AERPAW) for providing data to be analyzed.

CHAPTER 1

Introduction

In this thesis, we will explain the different types of estimation methods for estimating a radio map. The different estimation methods, input parameters, and RF scenarios will be discussed and compared. Our findings and comparisons will lead to best practices for estimating radio maps with sparse power measurements in a given region.

1.1. Motivation for model-free Radio Map Estimation

Often RF recievers (RX) are deployed in regions with little to no knowledge of the region's RF activity. In the case of sensitive RF RXs, not knowing the RF acitivty in a region of deployment will likely cause catastrophic failure to the RX front-end. With some research, a comprehensive list of current and near future RF transmitters (TX), as well as geometrical data may be compiled about the region. A straightforward and well-studied approach already exists for predicting RF RX power in the region with TX and geometrical data, RF propagation models [**6**]. Models such as the Free Space Path Loss (FSPL) will quickly and accurately predict how RF energy propagates to a given location. However, FSPL rarely gives a complete picture of RF propagation as there are many factors in an environment that affect how RF energy propagates; obstructions, humidity, reflections, etc. Furthermore, predicting a radio map with a model such as FSPL would require a calculation at every point in space, which may be time consuming. There are a handful of software packages on the market today that effectively consider all the factors needed to accurately predict RF propagation and automate testing multiple points for building radio maps, such as Wireless inSite [**12**]. These software packages normally require a powerful computer to calculate radio maps and may still take multiple hours to finish simulating a single radio map. The power required, simulation processing time, and cost of these software packages are not feasible for embedded or distributed real-time calculations.

Alternatively, model-free estimation methods may be utilized for estimating RF power [**1**]. This thesis explores different estimation methods for estimating RF radio maps without needing powerful computers, expensive software packages and comprehensive knowledge of a region's RF activity. Estimation methods have the benefits of estimating radio maps more quickly and more generally from scenario to scenario when compared to RF propagation models. The estimation methods we will consider in this thesis will rely on sparse RX measurements from the region for generating complete radio maps. However, there are some aspects of applying model-free estimation methods that need to be carefully considered. These aspects include accuracy, processing time, sensor count, sensor placement, and simply the estimation method which works best for the given environment; as different environments affect RF propagation differently.

1.2. Thesis Organization

This thesis is organized in the following manner:

- **Chapter 2** will discuss model-free radio map estimation more in-depth. This chapter will discuss the datasets used for our research and the considerations that needed to be taken into account when working with these datasets. Next, we will discuss how sparse subsets of our datasets are determined. Last, we will give an in-depth overview of all the different estimation methods considered for this research.
- **Chapter 3** will discuss the performance of each model-free estimation method described in chapter 2. The main performance categories which will be discussed are how accurate the radio map estimations are, and how fast the radio map estimations were computed. This chapter will also discuss a recommended set of parameters for applying model-free estimation methods to this problem and similar problems; e.g. recommended sparse dataset, recommended estimation methods, considerations with the estimation accuracy.
- **Chapter 4** will provide a final overview of this research, as well as provide recommended future work to expand upon this research.
- **Appendix A** will discuss how a single interpolation radio map estimation is analyzed.
- **Appendix B** will showcase an example for each ML methods estimation output as the sparse input increases.

1.3. Nomenclature

Tables 1.1 and 1.2 provide a list of acronyms and variables respectively, which are commonly used throughout this thesis.

TABLE 1.1. Acronyms used in this Thesis

Acronym	**Definition**
AERPAW	Aerial Experimentation and Research Platform for Advanced Wireless
AFE	Average Fractional Error
APL	Applied Physics Laboratory
BN	Batch-Normalization
CNN	Convolutional Neural Network
CS	Compressive Sensing/Sampling
DCT	Discrete Cosine Transform
DCU	Dense Compression Unit
ESRGAN	Enhanced Super-Resolution Generative Adversarial Networks
FE	Fractional Error
GP	Gaussian Process
GSP	Graph Signal Processing
IDCT	Inverse Discrete Cosine Transform
IDW	Inverse Distance Weighting
ISTA	Iterative Shrinkage-Thresholding Algorithm
ISTA-Net	Iterative Shrinkage-Thresholding Algorithm Net
JHU	Johns Hopkins University
LBFGS	Limited-memory Broyden-Fletcher-Goldfarb-Shanno
LDPL	Log Distance Path Loss
LIDAR	Light Detection and Ranging
LOS	Line Of Sight
LReLU	Leaky Rectified Linear Unit
MBI	Model Based Interpolation
MDA	Minimum Distance Algorithm
MSE	Mean Square Error
NRMSE	Normalized Root Mean Square Error
PAWR	Platforms for Advanced Wireless Research
PD	Pyramidal Decomposition
PReLU	Parametric Rectified Linear Unit
ProSR	Progressive Super-Resolution
PSNR	Peak Signal-to-Noise Ratio
ReLU	Rectified Linear Unit
RBF	Radial Basis Function
RF	Radio Frequency
RMSE	Root Mean Square Error
RRDB	Residual in Residual Dense Block
RSS(I)	Received Signal Strength (Indicator)
RX	RF Receiver
SR	Super-Resolution
SRGAN	Super-Resolution Generative Adversarial Networks
TX	RF Transmitter

TABLE 1.2. Variables used in this Thesis

Variable	**Definition**
$DB_v^{(k)}$	Virtual database spanning over the region of ground truth.
$DB^{(k)}$	Database of RPs selected from the virtual database.
$DB_{tr}^{(k)}$	Database of nearest RPs in the ground truth to our selected RPs in $DB^{(k)}$.
I^{HR}	Original high resolution image used for Super-Resolution.
I^{LR}	Low-resolution copy of I^{HR}, input to Super-Resolution algorithms.
I^{SR}	Super-Resolution generated image at the output of the algorithm.
RP_{truth}	Reference Points in ground truth
RP_{sparse}	Reference Points in sparse dataset
EP_{sparse}	Reference Points to estimate with sparse dataset
RP_{est}	Reference Points of radio map estimate

CHAPTER 2

Model Free Radio Map Estimation

In this chapter we will discuss the data used for our research, the different ways we found sparse datasets, and the different estimation methods we used for estimating the ground truth. Section 2.1 will discuss the datasets used as RP_{truth}, the problems encountered with these datasets, and how these problems are mitigated. Next, section 2.2 will discuss the different methods for generating RP_{sparse} from RP_{truth}. Finally, sections 2.3 and 2.4 will discuss the different interpolation and machine learning methods that accept RP_{sparse} as an input and estimate EP_{sparse}.

2.1. Ground Truth

2.1.1. Applied Physics Lab's Dataset. The APL dataset used for our research was provided by Johns Hopkins University Applied Physics Laboratory (JHU APL). The dataset provided by APL was generated using Wireless inSite Software with Light Detection and Ranging (LIDAR) information of a region in Atlanta Georgia. The LIDAR data of the region was used in conjunction with RF propagation models to simulate a radio map with an array of simulated RF Transmitters (TX). A 16 by 16 array of TX elements were simulated, transmitting a tone at 2.66 GHz. The TX elements were spaced apart by a half wavelength, which is 56.4 millimeters at 2.66 GHz. The TX array was simulated at a height of 201 meters. The RF RX antennas were simulated at a uniform height of 2.01 meters and were spaced apart uniformly by 0.8 meters. Figure 2.1 shows the mean power plotted from our dataset (left) next to a Google Satellite image of the region (right).

The distances shown in figure 2.1 are referenced from the center of the TX array. Looking at the distances shown in figure 2.1 we may observe that the plotted region is North of the TX array by about 4600 meters and is East of the TX array by about 150 meters. The total size of the region is $481 \times 639 meters = 0.308 km^2$. Furthermore, we may overlay the dataset's power plot on the Google Satellite image to spot check for consistency, figure 2.2.

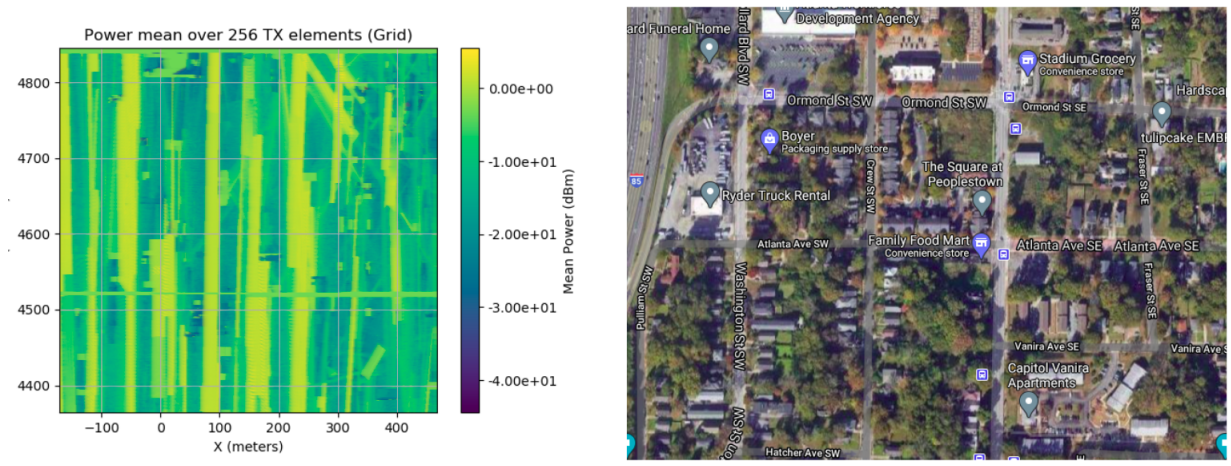

FIGURE 2.1. APL ground truth RX power (dBm) on the left and Google Satellite image of the region on the right.

Here we can see some consistency with our dataset. Specifically, the red circle shows how the broad side of a structure heavily attenuated RF power. The red rectangle shows how RF power stayed consistently high for the length of a street in the region, due to little obstructions along the street.

2.1.2. Aerial Experimentation and Research Platform for Advanced Wireless' Dataset. Aerial Experimentation and Research Platform for Advanced Wireless (AERPAW) is part of the National Science Foundation's (NSF) Platforms for Advanced Wireless Research (PAWR) initiative. Although the AERPAW testbed is currently still under construction, the AERPAW team was gracious enough to lend us some of their collected data used for calibrating the testbed. While other PAWR testbeds are focused on fixed RX and TX stations, AERPAW has an emphasis on testing RX and TX from an aerial platform. The current primary aerial platform uses multirotor

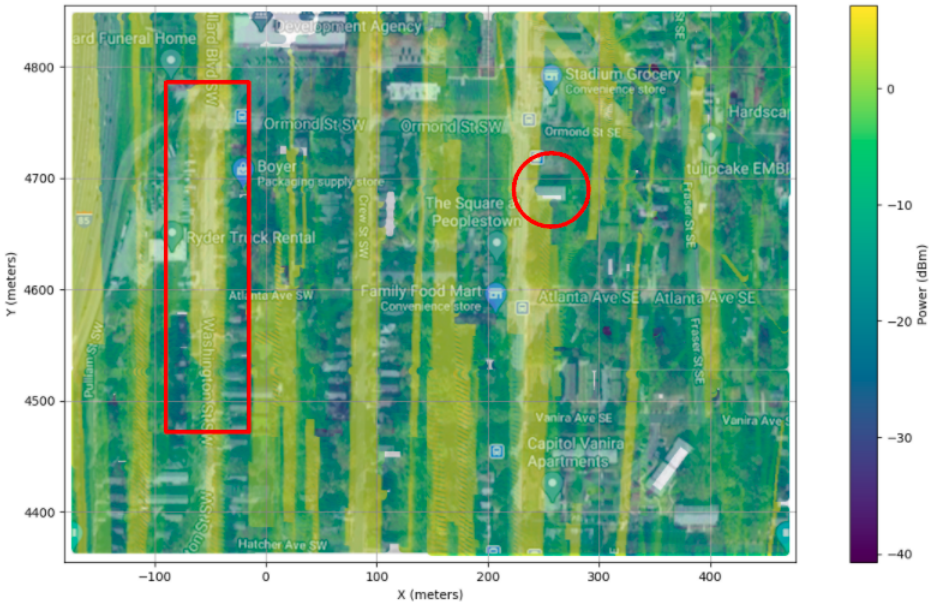

FIGURE 2.2. Superimposed image of APL ground truth RX power (dBm) on top of Google Satellite image of the region. The RX power is averaged over all 256 RX measurements at each location. It can be seen that ground truth is consistent with the region, such as the large structure attenuating RF power (red circle) and the vertical strips of high RF power due to unobstructed paths along the region streets (red rectangle).

drones, such as the hexacopter. Automating collection of real-world, uniform, and dense RF measurements is ideal for radio map estimation datasets, thus a mobile aerial platoform is an ideal RF measurement device.

As stated earlier, AERPAW will not be operational at the time of writing this thesis. We will therefore use a dataset provided by AERPAW which was collected for testbed calibration. Figure 2.3 shows the dataset's Receiver Signal Strength (RSS) plotted in 3-D space. The drone made two separate flights, first launching from the TX tower base. The drone flew to an altitude of 50m then flew out radially 300m and returned next to the TX tower along the same path. Periodic RX measurements were taken to generate this dataset as the drone was flying. Given that this data has dense measurements along two legs of 3-D space, 1-D radio map estimations will be considered; one along the 50m climb and another along the 300m radial arm.

An additional dataset similar to that of figure 2.3, will be considered and will provide more densely populated planar data for 2-D estimation, see figure 2.4.

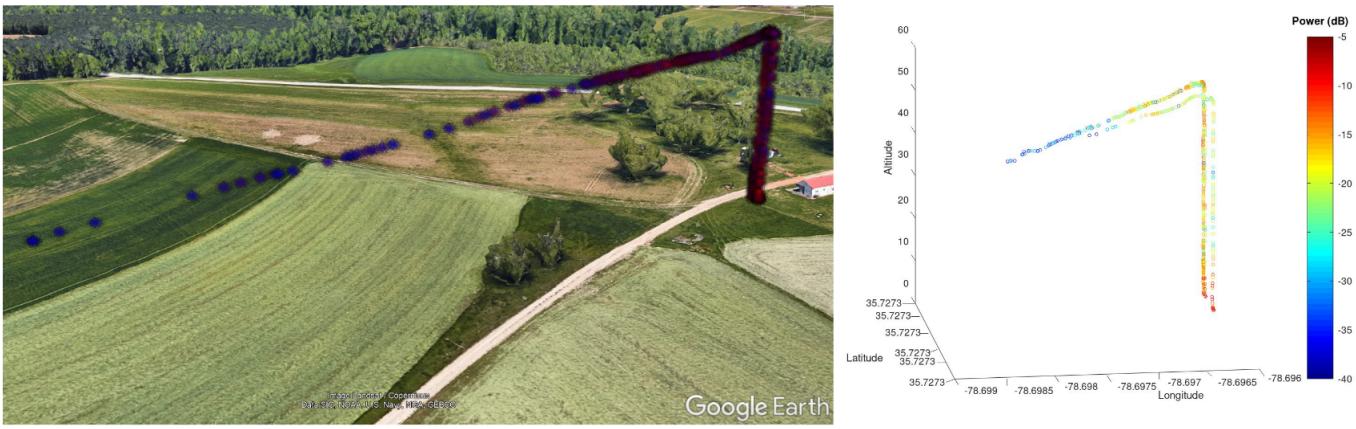

FIGURE 2.3. AERPAW provided data of two experimental flights on their aerial drone. The right shows the RSS measurements in Google Earth. The left shows a color gradient indicating the measured RSS value at a given longitude (x-axis), latitude (y-axis) and altitude (z-axis).

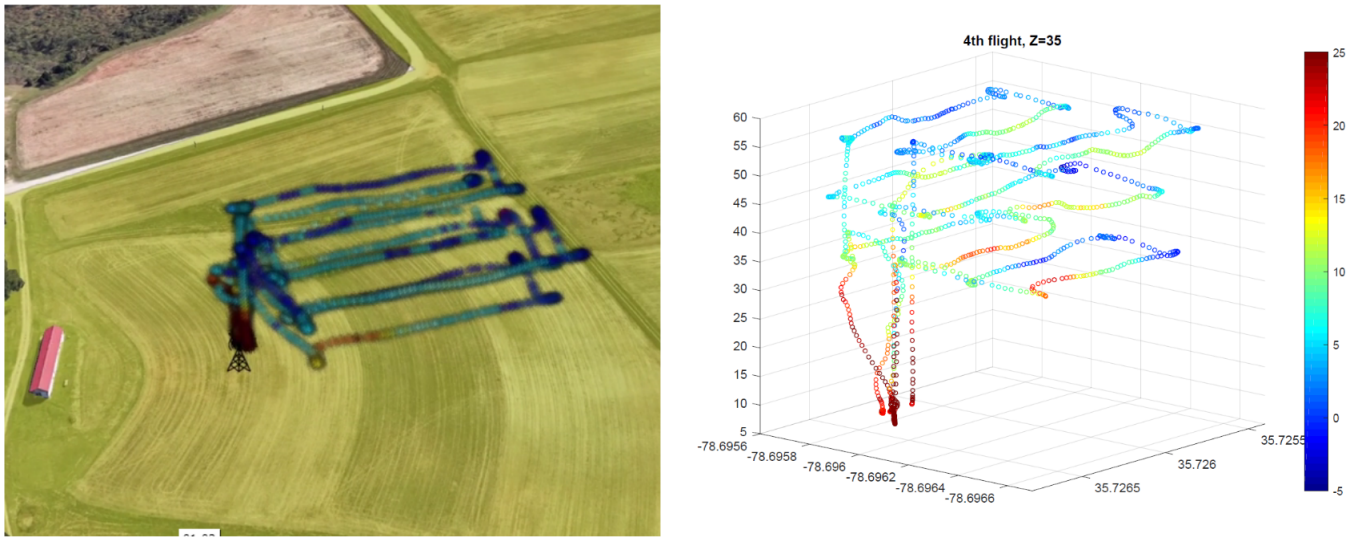

FIGURE 2.4. AERPAW collected additional data at their Lake Wheeler Testbed area. The right shows the RSS measurements in Google Earth. The left shows a color gradient indicating the measured RSS value at a given longitude (x-axis), latitude (y-axis) and altitude (z-axis). These images show three different flights at three different altitudes superimposed.

2.1.3. Grid Conforming. It was necessary that all datasets conformed to some discrete grid of uniformly spaced nodes such that there would be an upper bound on the number of Reference Points in (RP_{truth}) and estimation processing time.

Upon analysis of the dataset provided by APL, we discovered their data was inconsistent and did not line up with a uniform grid. Figure 2.5 shows the minor offsets in both the X and Y location data provided by APL. The plots shown in figure 2.5 are expected to have a continuous column of RX measurements (left) and a continuous row of RX measurements (right). APL's dataset was simulated in multiple patches and later stitched back together, this is one major reason for the inconsistent measurements displayed in figure 2.5.

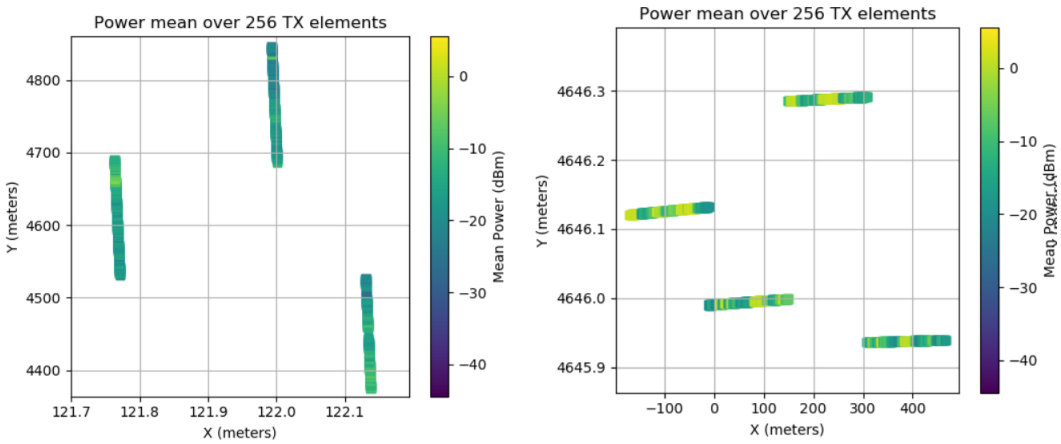

FIGURE 2.5. Inconsistent RX measurement locations in the data provided by APL. The large gaps in the data along each axis are the result of collecting the data in multiple simulations in different patches of the region.

The data was fit to a discrete grid of uniformly spaced nodes with equation 2.1, where RX_{xyz} is the original locations of RPs in RP_{truth} and res_{RX} is the expected resolution used for generating the dataset. Equation 2.1 conformed all the data to a consistent grid, but this process would cause some RP_{truth} to conform to shared locations; the APL dataset has about 98% of all RP_{truth} conforming to a unique location, thus 2% of RP_{truth} conforms to shared locations. All RP_{truth} that were in shared locations after conforming to a grid were averaged together. Furthermore, because APL removed the RP_{truth} that effectively measured zero RX power and to prevent error graphs from being saturated by large errors caused in these regions, we initialized those RP_{truth} to the average power of the ground truth. If a few values in a plot are much higher or lower than the average, the color scheme showing measurement variation would be limited to similar colors, thus making the graph indiscernible and "saturated."

$$DB^{(k)} = \left\lfloor \frac{RX_{xyz}}{res_{RX}} \right\rfloor * res_{RX} \tag{2.1}$$

2.2. Sparse Data

Selecting a subset of RP_{truth} to build the sparse dataset, RP_{sparse}, is nedded to test our estimation methods. Subsections 2.2.1 to 2.2.3 will discuss how we selected different subsets of RP_{truth}.

2.2.1. Random Selection. The first method used for selecting RP_{sparse} was a Gaussian random distribution. This is, of course, a fairly straightforward method of randomly choosing the desired amount of RP_{truth} to include into $DB^{(k)}$. Figure 2.6 shows an example of 500 RP_{truth} being selected from the pool of possible locations. Furthermore, figure 2.6 shows $DB_{tr}^{(k)}$ which has selected the ground truth points closest to all $DB^{(k)}$ points and saved the corresponding RSS value at those points. Note that the color scheme in $DB_{tr}^{(k)}$ is different from the color scheme in the ground truth due to the RSS value ranges being different. However, $DB_{tr}^{(k)}$ is storing the same RSS values as the ground truth for the selected locations.

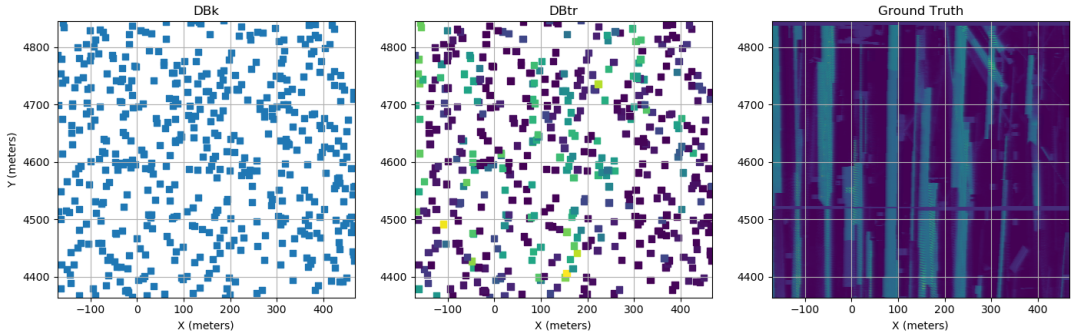

FIGURE 2.6. Random selection of RP_{truth} for $DB^{(k)}$ and $DB^{(tr)}$ while also showing ground truth for comparison. 500 RP_{truth} were selected for this figure. Note the data at each RP_{sparse} is consistent with the ground truth, but the color scheme on $DB^{(tr)}$ and ground truth differ due to the dataset RSS value ranges differing.

2.2.2. Grid Selection. Grid selection finds a uniform grid of points based on the aspect ratio of the ground truth and the amount RP_{truth} requested. Grid selection will first find a divisor based on the aspect ratio of the ground truth, equation 2.2.

(2.2) $$divisor_{aspect} = \frac{range(x)}{range(y)}$$

where $range(\cdot)$ refers to finding the range of axis locations. Equation 2.2 is applied when the range on the x-axis is greater and inverted when the range on the y-axis is greater. Next, the number of RP_{truth} to be selected along the smaller axis is found by equation 2.3.

(2.3) $$RP_{small} = \left\lfloor \sqrt{RP_{requested}/divisor_{aspect}} \right\rfloor$$

The number of RP_{truth} to be selected along the larger axis is found by equation 2.4.

(2.4) $$RP_{large} = \lfloor RP_{small} * divisor_{aspect} \rfloor$$

Finally, the number of RP_{truth} to be used in the sparse dataset may be found by equation 2.5.

(2.5) $$RP_{count} = RP_{small} * RP_{large}$$

An example of generating a RP_{sparse} with the grid selection method is shown in figure 2.7. In this example $RP_{requested} = 500$.

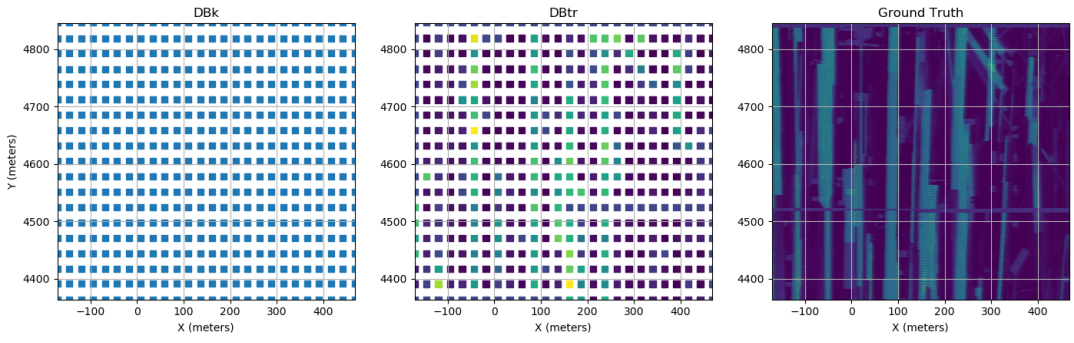

FIGURE 2.7. Grid selection of RP_{truth} for $DB^{(k)}$ and $DB^{(tr)}$ while also showing ground truth for comparison. 500 RP_{truth} were selected for this figure.

For the example presented in figure 2.7, our $DB_v^{(k)}$ has 604×800 possible RP_{truth}. Here $range(x) = 800$ and $range(y) = 604$ so following equation 2.2, $divisor_{aspect} = 800/604 = 1.32$. Next we find the range of the small and large axis for $DB^{(k)}$ using equations 2.3 and 2.4. In our example, $RP_{small} = \left\lfloor \sqrt{500/1.32} \right\rfloor = 19$ and $RP_{large} = \lfloor 19 * 1.32 \rfloor = 25$. This will finally lead us to our RP_{sparse} count using equation 2.5, $RP_{count} = 19 * 25 = 475$. We requested a $DB^{(tr)}$ with 500 RP_{truth} in a uniform grid, and we were given an RP_{sparse} with 475 RP_{truth}.

2.2.3. Minimum Distance Algorithm. Minimum Distance Algorithm (MDA) finds RP_{truth} that are the most spaced apart from all other selected RP_{truth}. MDA is a low-complexity algorithm proposed by reference [**14**] for finding these uniformly spaced RP_{truth}. First, a virtual grid is solved, $DB_v^{(k)}$, for a list of possible positions to select within the dataset; this is the same virtual grid discussed in section 2.1.3. Next, One RP_{truth} is randomly selected to start the process, and then each additional RP_{truth} is selected one at a time in a location that is most separated from all previously selected RP_{truth}. As each RP_{truth} is selected from the virtual grid, the selected RP_{truth} is removed from the pool of possible selections. MDA will provide the exact number of requested RP_{truth} for estimation. The pseudocode for MDA is provided by algorithm 1 and was first proposed in reference [**14**].

Algorithm 1 MDA

Require: the area P, the distance $\lambda^{(k)}$ between neighbor virtual RPs is $DB_v^{(k)}$, the number $n^{(k)}$ of RPs want to select.
Ensure: select RPs every $\lambda^{(k)}$ meters in P to build $DB_v^{(k)}$
Ensure: randomly select RP_c from $DB_v^{(k)}$, $DB^{(k)} = RP_c$
1: **while** $DB^{(k)} \neq n^{(k)}$ **do**
2: **for all** $RP_i \subset DB_v^{(k)}$ **do**
3: Calculate M_i
4: **end for**
5: $RP = \arg\min_{RP_i \subseteq DB_v^{(k)}} M_i$
6: $DB^{(k)} \leftarrow RP$
7: **end while**

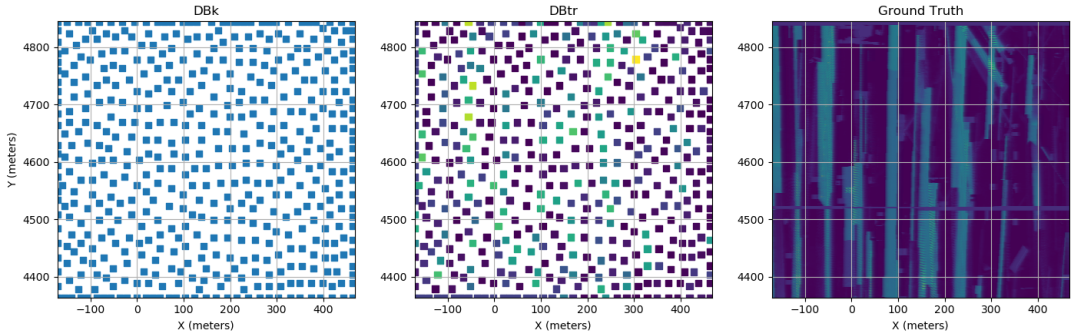

FIGURE 2.8. MDA selection of RP_{truth} for $DB^{(k)}$ and $DB^{(tr)}$ while also showing ground truth for comparison. 500 RP_{truth} were selected for this figure.

2.3. Interpolation Estimation

2.3.1. Uniform Localization. Due to the many obstructions in the region where the APL dataset was generated, discussed in section 2.1.1, we found that localizing the interpolation estimations performed better than global interpolations. The interpolation estimations only considered the Q closest RP_{sparse} when estimating EP_{sparse}. This type of localization will generate a Voronoi diagram like the one shown in figure 2.9.

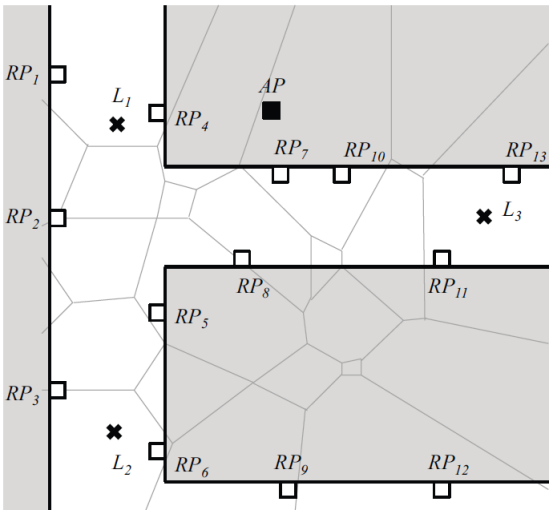

FIGURE 2.9. Q^{th}-order Voronoi diagram where $Q = 2$. This diagram shows the regions of unknown locations that need to be estimated with the two closest RP_{sparse}. For example, unknown location L_2 falls into the region where $RP_{sparse,3}$ and $RP_{sparse,6}$ are the closest RP_{sparse} [**10**].

All the sparse data selection methods discussed in section 2.2 find uniformly spaced RP_{truth}. Therefore, we may make a valid assumption of uniformly spaced RP_{sparse} when selecting the order of Q.

Selecting the order of Q was achieved by simply considering the average area covered by a single RP_{sparse}, and by specifying the diameter of localization we want to consider when estimating EP_{sparse}. For example, APL's dataset was collected in a region filled with trees and buildings. The RX was placed at a height of 2 meters which is about 6ft or the height of a tall human. Therefore, all the obstructions in this region will affect RF propagation and we cannot assume Line Of Sight (LOS) propagation. If we consider that this region has a grid of streets, we may reasonably consider an EP_{sparse} localization of 9 meters in diameter (about the width of a neighborhood street). This will give us a localization area of $pi * \left(\frac{9m}{2}\right)^2 = 63m^2$. Now considering the area covered by each RP_{sparse} may be determined by equation 2.6.

$$RP_{area} = \left(\frac{RP_{truth}}{RP_{sparse}}\right) * r_x * r_y \tag{2.6}$$

where RP_{sparse} and RP_{truth} are the number of RPs in the total dataset and sparse dataset respectively, and r_x and r_y are the x and y resolution of our virtual database, $DB_v^{(k)}$, respectively. We finally determine the order of our localization, Q, by using equation 2.7.

$$Q = \left\lceil \frac{localization_{area}}{RP_{area}} \right\rceil \tag{2.7}$$

If we consider our example with the APL dataset and assume a sparse dataset of 25% , then we will find an $RP_{area} = (4.0) * 0.8^2 = 2.56m^2$, where $r_x = r_y = 0.8$ in the APL dataset. Finally finishing this example, we will find that the order of Q will be $Q = \lceil 63m^2/2.56m^2 \rceil = 25$.

2.3.2. Gaussian Process. A 2-dimensional Gaussian Process (GP) will be derived and applied for estimating the ground truth. The derivation of the GP will follow the works described in [**3**] and [**14**]. Consider a vector of realistic and noisy measurements of $\mathbf{y}$ that was measured at locations $\mathbf{X} \in \mathbb{R}^d$. Taking n measurements will result in $\mathbf{y}$ equalling a vector with n values and $\mathbf{X}$

being a $d \times n$ matrix. A key GP property is that each measurement in $\mathbf{y}$ has some correlation to the neighboring measurements [**3**]. For this GP derivation, the assumed correlation between radio map mean RSS values will be the Gaussian Kernel as described in equation 2.8

$$\kappa(x_p, x_q) = \sigma_f^2 exp\left(-\frac{1}{2l^2}|x_p - x_q|^2\right) \tag{2.8}$$

where σ_f^2 is the signal variance, l is the length scale (this determines the correlation strength drop off between points [**3**]) and x_p/x_q are two dataset points. The variables σ_f^2 and l are hyperparameters and may be estimated based on the sparse dataset provided to the GP. σ_f^2 and l denote the signal variance and length scale of RP_{sparse} respectively. These hyperparameters determine the correlation of one RP's RSS measurement to neighboring RPs [**14**]. Section 2.3.2.1 will discuss how to estimate the GP hyperparameters. Using the Gaussian kernel as the correlation between points, the covariance of a single data point from the sparse dataset may be written as equation 2.9.

$$cov(y_p, y_q) = \kappa(x_p, x_q) + \sigma_n^2 \delta_{pq} \tag{2.9}$$

where σ_n^2 is the variance with all measurements in the sparse dataset and δ_{pq} equals 1 when $p = q$ and 0 otherwise. More appropriately, the covariance matrix of the entire sparse dataset may be written as equation 2.10.

$$cov(\mathbf{y}) = \mathbf{K} + \sigma_n^2 \mathbf{I} \tag{2.10}$$

where $\mathbf{K}$ is a $n \times n$ matrix composed of the evaluated kernel between all the points in the sparse dataset, $\mathbf{K}[p, q] = \kappa(x_p, x_q)$. Given the correlation between measurements have an assumed correlation following the Gaussian Kernel, all measurements in $\mathbf{y}$ are jointly Gaussian with $\mathbf{y}$ proportional to $\mathcal{N}(0, \mathbf{K} + \sigma_n^2 \mathbf{I})$ [**3**]. However, the more important observation is that an estimated measurement at any arbitrary position, x_*, may be modeled as a Gaussian random variable defined by equation 2.11 [**14**].

$$p(RSS(x_*)|x_*, \mathbf{X}, \mathbf{y}) = \mathcal{N}(RSS(x_*); \mu_{x_*}, \sigma^2_{x_*}) \tag{2.11}$$

where $RSS(x_*)$ simply refers to an RSS measurement at x_*, μ_{x_*} is the mean of the RSS measurement at x_* defined by equation 2.12, and $\sigma^2_{x_*}$ is the variance of the RSS measurement at x_* defined by equation 2.13.

$$\mu_{x_*} = \mathbf{k}_*^\top (\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{y} \tag{2.12}$$

$$\sigma^2_{x_*} = \kappa(x_*, x_*) - \mathbf{k}_*^\top (\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{k}_* \tag{2.13}$$

2.3.2.1. *Gaussian Process Hyperparameter Estimation.* Hyperparameters need to be estimated for the GP based on the set of input data, $\mathbf{y}$ and $\mathbf{X}$. The hyperparameters are estimated by maximizing the log-likelihood of $\mathbf{y}$ as defined in equation 2.14 [**15**]. Equation 2.14 is finding the

$$logp(\mathbf{y}|\mathbf{X}, \theta) = -\frac{1}{2}\mathbf{y}^\top (\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{y} - \frac{1}{2} log|\mathbf{K} + \sigma_n^2 \mathbf{I}| - \frac{n}{2} log 2\pi \tag{2.14}$$

where $\theta = \langle \sigma_n^2, l, \sigma_f^2 \rangle$ and σ_n^2 is the variance of measurements $\mathbf{y}$ while l and σ_f^2 are the two hyperparameters that must be estimated. Equation 2.14 will be used to find the hyperparameters, θ, maximizing the probability that our sparse data, $\mathbf{y}$, may be evaluated by our location data, $\mathbf{X}$. Equation 2.14 may be maximized using conjugate gradient descent [**3**]. Maximizing the log-likelihood via conjugate gradient descent may be accomplished with algorithms such as the Limited-memory Broyden-Fletcher-Goldfarb-Shanno (LBFGS) and will require the partial derivatives of the log-likelihood function with respect to all parameters in θ as defined by equation 2.15 [**3**].

$$\frac{\partial}{\partial \theta_j} logp(\mathbf{y}|\mathbf{X}, \theta) = \frac{1}{2} tr\left((\mathbf{K}^{-1}\mathbf{y})(\mathbf{K}^{-1}\mathbf{y})^\top \frac{\partial \mathbf{K}}{\partial \theta_j} \right) \tag{2.15}$$

where the partial derivatives of $\mathbf{K}$ are the partial derivatives of the Gaussian Kernel with respect to all parameters in θ and are defined by equations 2.16 - 2.18.

$$\frac{\partial \mathbf{K}}{\partial \sigma_f^2} = 2\sigma_f exp\left(-\frac{1}{2}\left(\frac{d}{l}\right)^2\right) \tag{2.16}$$

$$\frac{\partial \mathbf{K}}{\partial l} = \sigma_f^2 exp\left(-\frac{1}{2}\left(\frac{d}{l}\right)^2\right)\frac{d^2}{l^3} \tag{2.17}$$

$$\frac{\partial \mathbf{K}}{\partial \sigma_n^2} = 2\sigma_n \delta_{pq} \tag{2.18}$$

where $d = x_p - x_q$. The most complex computation when estimating the GP hyperparameters is the matrix inversion of $\mathbf{K}$ in equation 2.15. This computation has the complexity of $O(n^3)$, where n denotes the number of RP_{sparse} in $\mathbf{y}$. Furthermore, this same matrix inversion is needed when estimating each x_* as defined in equation 2.12. Section 3.5 will show estimating with the GP has significant processing time when compares to all other estimation methods.

2.3.3. Radial Basis Function. The Radial Basis Function is similar to the Gaussian Process in section 2.3.2 and will be applied for estimation in a similar fashion. The key difference with the RBF is the kernel used. Equation 2.8 defined a Gaussian Kernel, where equation 2.19 defines the RBF kernel.

$$\kappa(x_p, x_q) = exp\left(-\frac{1}{2l^2}|x_p - x_q|^2\right) \tag{2.19}$$

Furthermore, what is apparent when using the RBF for estimation is that there is one less hyperparameter to estimate with our sparse dataset. The RBF only needs to estimate the length-scale, l, and does not have the signal variance hyperparameter, σ_f^2. A slightly different derivation for applying the RBF to RSS estimation is provided in reference [**7**].

2.3.4. Discrete Cosine Transform Basis. Due to the sharp edges of APL's dataset, we then tried to use the Discrete Cosine Transform (DCT) as a basis. Using the DCT as a basis was originally posed to bring back some of the sharpness in our dataset estimation which the GP did a poor job on estimation. This was accomplished by using the information provided by the sparse dataset to solve for DCT coefficients. As a simple example, suppose $\mathbf{A}$ is a 3×3 matrix to be estimated with a DCT basis. Figure 2.10 shows $\mathbf{A}$ and how each value of $\mathbf{A}$ is indexed. Furthermore, figure 2.10 shows a sparsely sampled version of $\mathbf{A}$, where only 3 RP_{truth} were selected, $\mathbf{A}_s$.

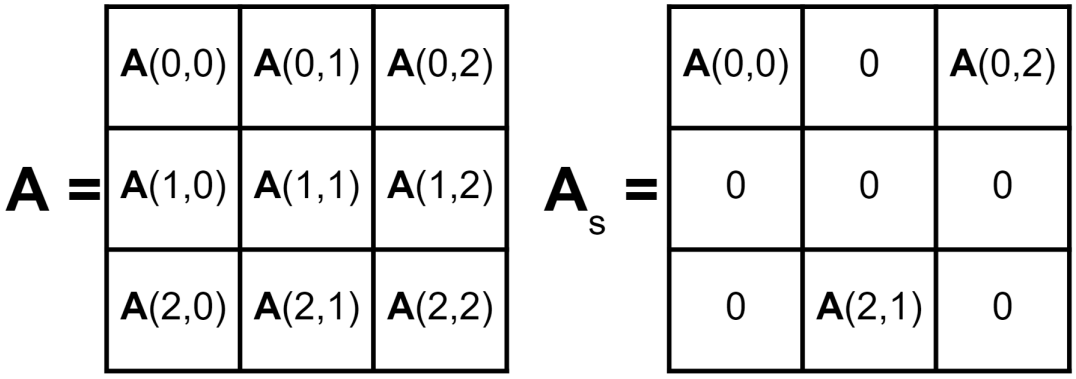

FIGURE 2.10. $\mathbf{A}$ should be considered an example matrix that stores some ground truth. $\mathbf{A}_s$ is a sparsely sampled matrix of the ground truth $\mathbf{A}$.

If $\mathbf{A}$ is known, then finding $\mathbf{B}$ is trivial. Simply take the 2-D DCT of $\mathbf{A}$. However, finding some estimated matrix of coefficients for the original ground truth is not quite as trivial. Consider the definition of the 2-D Inverse DCT (IDCT) given in equation 2.20 [**4**] [**11**].

$$\mathbf{A}_s(m,n) = \sum_{p=0}^{P-1} \sum_{q=0}^{Q-1} \mathbf{B}(p,q)\phi(p,m,P)\phi(q,n,Q) \tag{2.20}$$

where $\mathbf{B}$ is the DCT coefficient matrix, P is the total length of the first axis, and Q is the total length of the second axis (in figure 2.10's example, $P = Q = 3$), and where $\phi(\cdot)$ is defined by equation 2.21.

$$\phi(p,m,P) = \alpha_m cos \frac{\pi(p)(2m+1)}{2P} \tag{2.21}$$

where α_m is defined by equation 2.22.

$$\alpha_m = \begin{cases} \frac{1}{\sqrt{P}} & m = 0 \\ \sqrt{\frac{2}{P}} & 1 \leq m \leq P-1 \end{cases} \tag{2.22}$$

A system of equations may be built to solve for some estimated coefficient matrix, $\mathbf{B}_s$. The system of equations to solve for $\mathbf{B}_s$ is shown in equation 2.23.

(2.23)

$$\mathbf{A}(0,0) = \mathbf{B}(0,0)\phi(0,0,P)\phi(0,0,Q) + \mathbf{B}(0,1)\phi(0,0,P)\phi(1,0,Q) + ... + \mathbf{B}(2,2)\phi(2,0,P)\phi(2,0,Q)$$

$$\mathbf{A}(0,2) = \mathbf{B}(0,0)\phi(0,0,P)\phi(0,2,Q) + \mathbf{B}(0,1)\phi(0,0,P)\phi(1,2,Q) + ... + \mathbf{B}(2,2)\phi(2,0,P)\phi(2,2,Q)$$

$$\mathbf{A}(2,1) = \mathbf{B}(0,0)\phi(0,2,P)\phi(0,1,Q) + \mathbf{B}(0,1)\phi(0,2,P)\phi(1,1,Q) + ... + \mathbf{B}(2,2)\phi(2,2,P)\phi(2,1,Q)$$

The system of equations in equation 2.23 is underdetermined and has many solutions, not just one. This example has $\binom{9}{3} = 84$ different solutions, assuming that only 3 coefficients are solved for and all other coefficients are set to 0. The underdetermined nature of this problem is how the DCT is being used as the basis for estimating the ground truth. The final step is choosing which DCT coefficients to solve for with our estimation. The answer to this question simply is to choose the lower frequency, and more impactful, DCT coefficients. These are the DCT coefficients in the upper left corner due to how the indexing of the DCT works. Figure 2.11 shows the pattern for selecting DCT coefficients with ascending frequency (i.e. lower frequency coefficients first).

In this example, the DCT coefficients that are selected to be solved for are $\mathbf{B}_s = \{\mathbf{B}(0,0), \mathbf{B}(0,1), \mathbf{B}(1,0)\}$. The final step for estimating the ground truth, is simply to take the 2-D IDCT of $\mathbf{B}_s$.

2.3.5. Model Based Interpolation. As a mediation between model free and model based estimation methods, we wanted to look at the performance of a Log Distance Path Loss (LDPL) Model Based Interpolation (MBI) method. This method begins with the LDPL model shown in equation 2.24 and makes a parameterized version of this model shown in equation 2.25 [**10**].

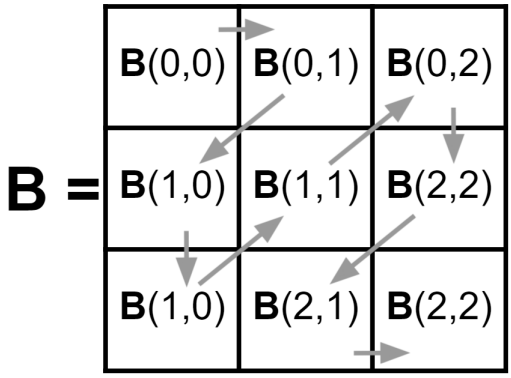

FIGURE 2.11. The order for choosing which DCT coefficients to solve for when using a DCT basis for estimating ground truth.

$$P(d) = P(d_0) - 10\theta log_{10}(d/d_0) \tag{2.24}$$

where $P(d), \theta, d_0, P(d_0)$ denote an EP_{sparse} at a distance d, path loss exponent, reference distance for an RP_{sparse}, and RSS value respectively.

$$Z_k(x,y) = \hat{c}_k - 10\hat{\theta}_k log_{10}\left(\sqrt{(\hat{x}_k - x)^2 + (\hat{y}_k - y)^2}\right) \tag{2.25}$$

where k denotes the k^{th} TX. $Z_k(x,y), \hat{\theta}, \hat{c}_k$ denote the $P(d), \theta, P(d_0)$ when estimating for the k^{th} TX. Lastly, the $\frac{TX_k}{EP}$ distance from the k^{th} TX is represented as the Euclidean distance. Fortunately in this problem, there is only 1 TX and the location of that TX is known, so equation 2.25 then becomes equation 2.26.

$$Z(x,y) = \hat{c} - 10\hat{\theta} log_{10}\left(\sqrt{(x_{TX} - x)^2 + (y_{TX} - y)^2}\right) \tag{2.26}$$

Using equation 2.26, the unknown parameters of the problem may be directly solved with RP_{sparse} by using equations 2.27, 2.28 and 2.29.

$$(\hat{\theta}, \hat{c}) = \arg\min_{\theta,c} \sum_{i=1}^{N} \frac{w_i}{\sum_{j=1}^{N} w_j} \gamma_i^2 \tag{2.27}$$

where w_i denotes residual weighting and γ_i denotes the residual for $RP_{sparse,i}$.

$$w_i = \frac{1}{|s_i|^p} \tag{2.28}$$

where s_i denotes RSS value at $RP_{sparse,i}$, and p denotes the power parameter for residual weighting. Selection of the power parameter takes special care to obtain expected parameters.

$$\gamma_i = s_i - c + 10\theta log_{10}\left(\sqrt{(x_{TX} - x_i)^2 + (y_{TX} - y_i)^2}\right) \tag{2.29}$$

where x_i, y_i denotes the x location and y location respectively for $RP_{sparse,i}$. Once the best parameters $\hat{\theta}, \hat{c}$ are solved for with the provided sparse dataset, we then use these parameters to estimate all EP_{sparse}.

2.3.6. Inverse Distance Weighting Interpolation. Inverse Distance Weighting (IDW) interpolation, also known as Shepard's method, is a simple interpolation method where EP_{sparse} are calculated by the sum of RP_{sparse} RSS values multiplied by a normalized weight. The weight is determined by Euclidean distance between any given EP_{sparse} and RP_{sparse}. Equations 2.30 and 2.31 show the definition for IDW interpolation as presented in [**8**].

$$P_r(l, b_j) = \frac{1}{\sum_{l_i \in \mathcal{L}} w_i} \sum_{l_i \in \mathcal{L}} (w_i \times v_{i,j}) \tag{2.30}$$

where l_i is the $RP_{sparse,i}$ from the RP_{sparse}, $v_{i,j}$ is the RSS value from the j^{th} TX at the $RP_{sparse,i}$, and w_i is defined by equation 2.31.

$$w_i = \| l, l_i \|_2^{-\lambda} \tag{2.31}$$

where l is the location of an EP_{sparse}, l_i is the location of the $RP_{sparse,i}$ and λ is identified as the system paramter. Typically $\lambda > 0$ and is usually set to a samll value such as 0.01 [**8**].

Similary to the LDPL-MBI method in section 2.3.5, we may reduce equation 2.30 to equation 2.32 because our datasets only have a single TX.

$$P_r(l,b) = \frac{1}{\sum_{l_i \in \mathcal{L}} w_i} \sum_{l_i \in \mathcal{L}} (w_i \times v_i) \tag{2.32}$$

2.4. Machine Learning Estimation

A few Machine Learning (ML) algorithms were considered for estimating the region's radio map. In the next few sections, each ML algorithm will be discussed and compared, as well as how the input and output for ML algorithms were generated.

2.4.1. Enhanced Super-Resolution Generative Adversarial Networks.

2.4.1.1. *Super-Resolution Generative Adversarial Networks.* Enhanced Super-Resolution Generative Adversarial Networks (ESRGAN) [**17**] provided follow on work to the project Super-Resolution Generative Adversarial Networks (SRGAN) [**9**]. The main contributions made by SRGAN are to generate natural images that do not look like they were generated by a computer. Many SR algorithms prior to SRGAN would generate high-resolution images by minimizing pixel comparison error metrics between the generated high-resolution image and the original high-resolution image (i.e. RMSE or PSNR). SRGAN proposed that these algorithms generate the most accurate high-resolution images, but that the generated images lose a lot of perceptual features that would allow the human eye to recognize the image as computer-generated. Therefore, SRGAN developed a new way of generating high-resolution images that not only has a network for generating the high-resolution images (Generative Network), but also a second network to discriminate between images that look computer-generated and images that look authentic and natural (Adversarial/Discriminator Network).

The generative network is developed as a feed-forward Convolutional Neural Network (CNN), denoted as G_{θ_G}, that is parameterized by θ_G [**9**]. The parameters, $\theta_G = \{W_{1:l}, b_{1:l}\}$, define the weights and biases of an L-layer deep network [**9**]. θ_G may be obtained by optimizing the generative

network with a specific loss function, l^{SR} [**9**]. Given high resolution images, $I_n^{HR}, n = 0, 1, ..., N$ (ground truth), and corresponding low resolution copies, $I_n^{LR}, n = 0, 1, ..., N$, equation 2.33 shows how SRGAN obtains $\hat{\theta_G}$ when training [**9**].

$$\hat{\theta}_G = \arg\min_{\theta_G} \frac{1}{N} \sum_{n=0}^{N} l^{SR}(G_{\theta_G}(I_n^{LR}), I_n^{HR}) \tag{2.33}$$

Here θ_G are the parameters being trained by minimizing the average loss function of a generated high resolution image, I^{SR}, and the corresponding original high resolution image, I^{HR}. $\hat{\theta_G}$ is simply the best generative network parameters found by equation 2.33. The loss function, l^{SR}, used in SRGAN was the main contribution made by reference [**9**]. SRGAN defined their l^{SR} in equation 2.34.

$$l^{SR} = l_X^{SR} + 10^{-3} l_{Gen}^{SR} \tag{2.34}$$

where l_X^{SR} is the "content loss" described by equation 2.35 and l_{Gen}^{SR} is the "adversarial loss" described by equation 2.36.

$$l_{VGG/i,j}^{SR} = \frac{1}{W_{i,j} H_{i,j}} \sum_{x=1}^{W_{i,j}} \sum_{y=1}^{H_{i,j}} (\phi_{i,j}(I^{HR})_{x,y} - \phi_{i,j}(G_{\theta_G}(I^{LR}))_{x,y})^2 \tag{2.35}$$

$$l_{Gen}^{SR} = \sum_{n=1}^{N} -logD_{\theta_D}(G_{\theta_G}(I^{LR})) \tag{2.36}$$

The term "VGG" in equation 2.35 was simply the name of a 2nd place team in the ILSVRC-2014 competition [**16**]. The CNN used by the team, VGG, has since become the name for the CNN itself. In equation 2.35 VGG19 is used, which is a pre-trained 19 layer VGG CNN [**9**]. Here $\phi_{i,j}$ denotes the feature map obtained by the j-th convolution (after activation) before the i-th max-pooling layer within the VGG19 CNN; this information is a given with the pre-trained VGG19 network [**9**].

Lastly, the terms $W_{i,j}$ and $H_{i,j}$ in equation 2.35 describe the dimensions of the respective feature maps within the VGG network [**9**].

The Discriminator Network, denoted as D_{θ_D}, also is a parameterized CNN much like the generative network with similar parameters, $\theta_D = \{W_{1:l}, b_{1:l}\}$. The discriminator network is trained simultaneously with the generative network. Equation 2.37 shows how the discriminator network is trained with respect to the trained parameters of the generative network, θ_G [**9**] [**5**].

(2.37)

$$\min_{\theta_G} \max_{\theta_D} l^{SR}_{Dis}(\theta_D, \theta_G) = \mathbb{E}_{I^{HR} \sim p_{train}(I^{HR})} \left[log D_{\theta_D}(I^{HR}) \right] + \mathbb{E}_{I^{LR} \sim p_G(I^{LR})} \left[log(1 - D_{\theta_D}(G_{\theta_G}(I^{LR}))) \right]$$

Equation 2.37 essentially shows that the Generative Network is trying to minimize θ_G such that the Generated image is indistinguishable from a real image (i.e. $I^{SR} \cong I^{HR}$), while the discriminator network is trying to maximize θ_D such that the network probability of accuracy for discriminating I^{HR} from I^{SR} increases.

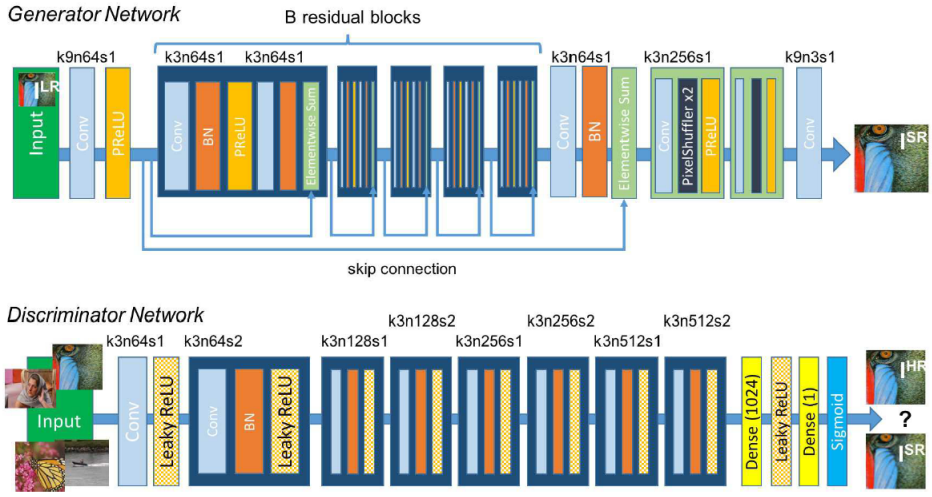

FIGURE 2.12. SRGAN Generative Network (top) and Adversarial/Discriminator Network (bottom) block diagrams [**9**].

Figure 2.12 shows the block diagrams for both the generative network (top) and discriminator network (bottom) used in SRGAN. The Generative Network is comprised of what are called "B

residual blocks" which all have identical layouts. Each B residual block is comprised of two convolutional layers with small 3×3 kernels and 64 feature maps followed by Batch-Normalization (BN) layers and Parametric Rectified Linear Unit (PReLU) as the activation function [**9**]. Finally, the generative network increases the resolution of the input image with two trained sub-pixel convolution layers [**9**]. The discriminator network is comprised of eight convolutional layers with an increasing number of 3×3 filter kernels, doubling filter kernels layer by layer, from 64 to 512. Seven of the convolutional layers are followed by a BN layer, and all convolutional layers are followed by a Leaky Rectified Linear Unit (LReLU) activation layer with $\alpha = 0.2$ [**9**]. Finally, the resulting 512 feature maps are followed by two dense layers and a sigmoid activation function to obtain a probability used for input image classification [**9**].

2.4.1.2. *Enhanced Super-Resolution Generative Adversarial Networks.* ESRGAN made four major adjustments to SRGAN. First, ESRGAN modified the architecture of SRGAN as shown in figure 2.13.

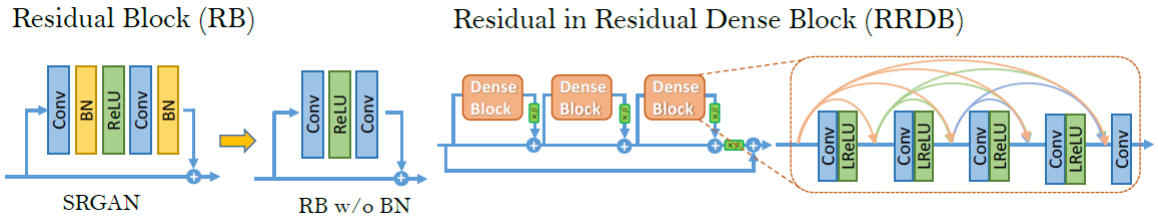

FIGURE 2.13. ESRGAN update to the B residual blocks (left), and block diagram of RRDB [**17**].

The BN layers were removed from the B residual blocks in SRGAN. The BN layers were removed because these layers may increase computational complexity, introduce undesired artifacts when the network is deeper and trained under GAN, and reduce performance stability as well as generalization of the network [**17**]. Also, ESRGAN added Residual in Residual Dense Blocks (RRDB) in their deeper model with β as a residual scaling parameter (a multiplier at the output of each Dense Block). RRDB was used on ESRGAN's deeper model because the authors observed that more layers and connections could boost performance.

Next, ESRGAN adjusted the discriminator network and based it on a Relativistic GAN. The discriminator network in SRGAN estimated the probability that an image is generated or real,

where a relativistic discriminator network predicts the probability of how much more I^{HR} is real relative to I^{SR}. SRGAN may be expressed as $D_{\theta_D}(I^{SR}) = \sigma(C(I^{SR}))$, where σ is the sigmoid activation layer used in the discriminator network and $C(\cdot)$ is the non-transformed discriminator output. However, ESRGAN may be expressed as $D_{\theta_{Ra}}(I^{HR}, I^{SR}) = \sigma(C(I^{HR}) - \mathbb{E}_{I^{SR}}C(I^{SR}))$, where $\mathbb{E}_{I^{SR}}$ represents the expectation of all the generated images in a batch of inputs. With this information, the discriminator loss function then becomes equation 2.38.

$$l_{Dis}^{Ra} = -\mathbb{E}_{I^{HR}}\left[log(D_{\theta_{Ra}}(I^{HR}, I^{SR}))\right] - \mathbb{E}_{I^{SR}}\left[log(1 - D_{\theta_{Ra}}(I^{SR}, I^{HR}))\right] \tag{2.38}$$

The adversarial loss, which SRGAN defines by equation 2.36, then becomes equation 2.39.

$$l_{Gen}^{Ra} = -\mathbb{E}_{I^{HR}}\left[log(1 - D_{\theta_{Ra}}(I^{HR}, I^{SR}))\right] - \mathbb{E}_{I^{SR}}\left[log(D_{\theta_{Ra}}(I^{SR}, I^{HR}))\right] \tag{2.39}$$

One key difference with ESRGAN is that the adversarial loss takes advantage of the gradients from both generated data, I^{SR}, and real data, I^{HR} [**17**].

ESRGAN also adjusted the perceptual loss function which SRGAN previously defined by equation 2.34. Equation 2.34, specifically the adversarial loss defined by equation 2.36, operates on the feature maps obtained by the j-th convolution after the activation layer. The feature map after the activation layer is very sparse, especially after a very deep network, which provides weak supervision and leads to inferior performance [**17**]. Therefore, ESRGAN proposes a generative network loss function defined by equation 2.40.

$$l_G = l^{SR} + \lambda l_{Gen}^{Ra} + \eta l_1 \tag{2.40}$$

where $l_1 = \mathbb{E}_{x_i}(\|G(x_i) - y\|)_1$ is the content loss that evaluates the L_1-norm distance between the estimated image I^{SR} and the ground truth I^{HR}, and λ,η are the coefficients to balance different loss terms [**17**].

Lastly, ESRGAN adjusted the generative network parameters, θ_G. ESRGAN gave a set of parameters that are a hybrid of the optimized parameters for a GAN solution, θ_G^{GAN}, and a PSNR solution, θ_G^{PSNR}. The hybrid parameters, θ_G^{INTERP}, are defined in equation 2.41 [**17**].

$$\theta_G^{INTERP} = (1-\alpha)\theta_G^{PSNR} + \alpha\theta_G^{GAN} \tag{2.41}$$

where $\alpha \in [0, 1]$ is the interpolation parameter. The generative network interpolation was used in ESRGAN because the interpolation did not introduce any artifacts and fine-tuning a balance between optimal accuracy and perceptual quality does not require retraining the network.

2.4.2. Progressive Super-Resolution. Progressive SR (ProSR) developed a model using a Pyramidal Decomposition (PD). Figure 2.14 shows the composition of ProSR's network and the PD construction is apparent with smaller and deeper layers (u_0) becoming progressively larger and shallower (u_2).

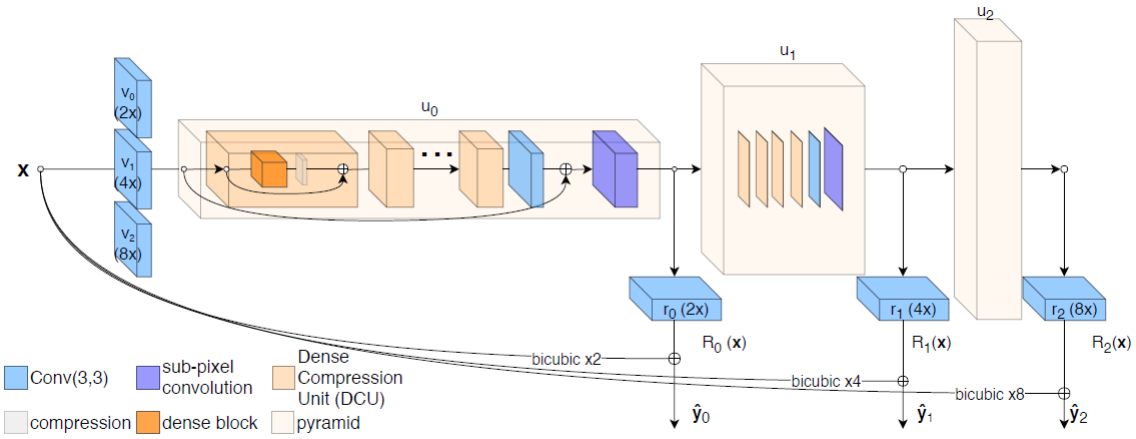

FIGURE 2.14. ProSR Block Diagram [**18**].

ProSR comes with a pre-trained model without GAN and a pre-trained model with GAN. For the focus of this research, we are comparing the model without GAN as this will generate images with minimal error and maximum PSNR. In this application of SR, minimizing error is more important than generating high perceptual quality. This is because our application is trying

to generate a radio map as accurately as possible, and is not trying to generate a natural image to fool a human into believing the generated image is a real image.

ProSR begins by trying to solve some upscaling function u such that $u(I^{LR}) = I^{SR} \cong I^{HR}$. However, the larger the upscaling ratio, the more complex the parameterization of u is required. For this reason, ProSR proposes to use a PD scheme where each level will refine the feature representation and upscale the input image by a factor of 2. That is, u is broken up into simpler levels $u_0, u_1, ..., u_n$ for this PD architecture. As shown in figure 2.14, each level is comprised of a Dense Compression Unit (DCU) and followed by a sub-pixel convolutional layer. The DCUs have a similar structure as the B Residual blocks used in ESRGAN, figure 2.13; 1×1 convolutional layer, Rectified Linear Unit (ReLU) layer, 3×3 convolutional layers. Furthermore, the PD architecture proposed by ProSR follows an asymmetric structure where the first level, u_0, will have the most DCUs, and the number of DCUs will progressively decrease going into the higher levels. This architecture differs from a symmetric structure because symmetric structures will have a symmetric number of DCUs with respect to the level number. Due to ProSR's asymmetric structure having more DCUs with smaller inputs, ProSR will decrease memory consumption, decrease run-time and increase the quality of I^{SR} due to an increased receptive field with respect to I^{HR} [**18**]. ProSR also uses two scale-specific sub-networks, denoted by v_n and r_n, which allow for an individual transformation between scale-varying image space and a normalized feature space [**18**]. I^{SR} may be solved by first finding the residuals with equation 2.42.

$$\mathcal{R}_n(I^{LR}) = (r_n \circ u_n \circ ... \circ u_1 \circ u_0 \circ v_n)(I^{LR}) \tag{2.42}$$

where $A \circ B$ denotes the output of block B will be the input of block A. For ProSR, we have the ability for a x2, x4 and x8 resolution increase. Therefore we will have the following residuals, $\mathcal{R}_0(I^{LR}) = (r_0 \circ u_0 \circ v_0)(I^{LR})$, $\mathcal{R}_1(I^{LR}) = (r_1 \circ u_1 \circ u_0 \circ v_1)(I^{LR})$, and $\mathcal{R}_2(I^{LR}) = (r_2 \circ u_2 \circ u_1 \circ u_0 \circ v_2)(I^{LR})$. Then adding the residuals with a fixed upsampling of the input, $\varphi(I^{LR})$, will result in a generated image, equation 2.43. The upsampling of the input, $\varphi(I^{LR})$, may be a bicubic interpolation for example.

$$I^{SR} = \mathcal{R}_n(I^{LR}) + \varphi(I^{LR}) \tag{2.43}$$

For an example where we want to have a x8 resolution increase, our ProSR output will have the following formula, $I^{SR} = (r_2 \circ u_2 \circ u_1 \circ u_0 \circ v_2)(I^{LR}) + \varphi(I^{LR})$.

2.4.3. Iterative Shrinkage-Thresholding Algorithm Net. Iterative Shrinkage-Thresholding Algorithm Net (ISTA-Net) attempts to use CNNs to solve the general underdetermined problem presented in Compressive Sensing (CS). CS proposes the idea of reconstructing a dataset that was undersampled less than Nyquist-Shannon's famous theorem. Imagine having a vector $\mathbf{x} \in \mathbb{R}^N$, such as an image. Then a severely undersampled (less than 50% of the data) vector of $\mathbf{x}$ is produced as $\mathbf{y} = \Phi\mathbf{x} \in \mathbb{R}^M$. Here $\Phi \in \mathbb{R}^{M \times N}$ is the sampling matrix of $\mathbf{x}$. In some situations, $\mathbf{x}$ may be a known vector, but using CS to reduce data size may be important, such as compressing an image [**21**]. In other situations $\mathbf{x}$ could not feasibly be a known vector and CS would be a necessary tool, such as sampling RX values in every square meter of a square kilometer of space. In the latter scenario, $\mathbf{y}$ could not be solved directly, but rather a placement of multiple RXs to take a measurement simultaneously will generate $\mathbf{y}$. In this situation, the locations of RXs will be the definition of Φ. After $\mathbf{y}$ and Φ are determined, a new vector $\tilde{\mathbf{x}}$ that is the same size as $\mathbf{x}$, will be used as the variable for reconstructing $\mathbf{x}$ in $\mathbf{y} = \Phi\tilde{\mathbf{x}}$. However, solving for $\tilde{\mathbf{x}}$ is an underdetermined system of equations with infinitely many solutions. Fortunately, given that a Φ sampling matrix is chosen under the correct conditions, $\mathbf{x}$ may be reconstructed best by solving for equation 2.44.

$$\arg\min_{\tilde{\mathbf{x}}} \frac{1}{2} \parallel \Phi\tilde{\mathbf{x}} - \mathbf{y} \parallel_2^2 + \lambda \parallel \Psi\tilde{\mathbf{x}} \parallel_1 \tag{2.44}$$

Here Ψ is an $N \times N$ matrix of a basis and λ is a regularization parameter. Ψ may be the basis of the FFT, DCT, or Wavelets for example [**2**]. Reconstruct $\mathbf{x}$ is done best by solving for an $\tilde{\mathbf{x}}$ with the minimum l_1 norm (as depicted in equation 2.44), Φ needs to be sufficiently incoherent with the basis matrix Ψ and must obey the Restricted Isometry Property [**2**]. Some examples of a Φ following these conditions are a random Gaussian Distribution, a Bernoulli Distribution, or

a partial Fourier Matrix [**2**]. Keep in mind that a Φ following a Gaussian Distribution or partial Fourier Matrix will generate weighted samples with $weights \in [0, 1]$ when normalized, whereas Φ following a Bernoulli Distribution will have $weights \in \{0, 1\}$.

A common way to solve the minimum l_1 norm is with an algorithm called Iterative Shrinkage-Thresholding Algorithm (ISTA). By applying ISTA algorithms to solve for equation 2.44, equations 2.45 and 2.46 would be applied iteratively until an acceptable minimum $\| \tilde{\mathbf{x}} \|_1$ is computed.

$$\mathbf{r}^{(k)} = \tilde{\mathbf{x}}^{(k-1)} - \rho\Phi^\top(\Phi\tilde{\mathbf{x}}^{(k-1)} - \mathbf{y}) \tag{2.45}$$

$$\tilde{\mathbf{x}}^{(k)} = \arg\min_{\tilde{\mathbf{x}}} \frac{1}{2} \| \tilde{\mathbf{x}} - \mathbf{r}^{(k)} \|_2^2 + \lambda \| \Psi\tilde{\mathbf{x}} \|_1 \tag{2.46}$$

where k denotes which iteration of ISTA the result is on and ρ denotes a constant step size for the ISTA algorithm [**20**]. Furthermore, $\mathbf{r}^{(k)}$ may be regarded as some type of noisy observation of $\mathbf{x}$ [**21**]. Solving for the term $\| \Psi\tilde{\mathbf{x}} \|_1$ in equation 2.46 presents the greatest complexity when solving equation 2.44 with ISTA [**20**] [**21**].

Therefore, ISTA-Net proposes to reduce the complexity of an ISTA algorithm by replacing the l_1 norm minimization in equation 2.46 with CNNs. Specifically, ISTA-Net will replace the l_1 norm minimization with two CNNs and a ReLU as an activation function. See figure 2.15 for a block diagram of ISTA-Net.

ISTA-Net replaces the term Ψ with $\mathcal{F}(\cdot)$, where $\mathcal{F}(\mathbf{x}) = \mathbf{B}ReLU(\mathbf{A}\mathbf{x})$ ($\mathbf{A}$ and $\mathbf{B}$ correspond to the two CNNs for each phase shown in figure 2.15). By replacing Ψ in equation 2.46, ISTA-Net reconstructs $\mathbf{x}$ by following equation 2.47.

$$\tilde{\mathbf{x}}^{(k)} = \arg\min_{\tilde{\mathbf{x}}} \frac{1}{2} \| \tilde{\mathbf{x}} - \mathbf{r}^{(k)} \|_2^2 + \lambda \| \mathcal{F}(\tilde{\mathbf{x}}) \|_1 \tag{2.47}$$

2.4.4. Semantic Image Inpainting with Deep Generative Models. Semantic inpainting algorithm attempts to fill in a complete region of missing data which has been removed from an image. Similar to sections 2.4.1 and 2.4.2, Semantic Image Inpainting uses generative and

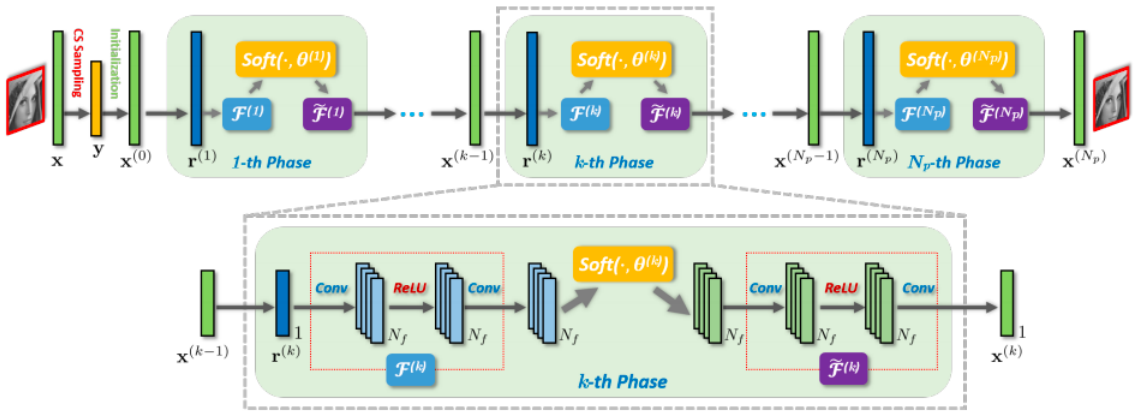

FIGURE 2.15. ISTA-Net block diagram. $\mathbf{x}$ is the original input image, $\mathbf{y}$ is the undersampled vector of $\mathbf{x}$, k is the phase/iteration number for each iteration of ISTA, N_P is the total number of ISTA iterations, $r^{(k)}$ is an iterative noisy observation of $\mathbf{x}$, and $x^{(k)}$ is an iterative reconstruction of $\mathbf{x}$ [**20**]

discriminator networks. The generative network will train by generating images and receiving feedback from whether or not the discriminator network was fooled. Figure 2.16 shows a block diagram overview of the Semantic inpainting algorithm.

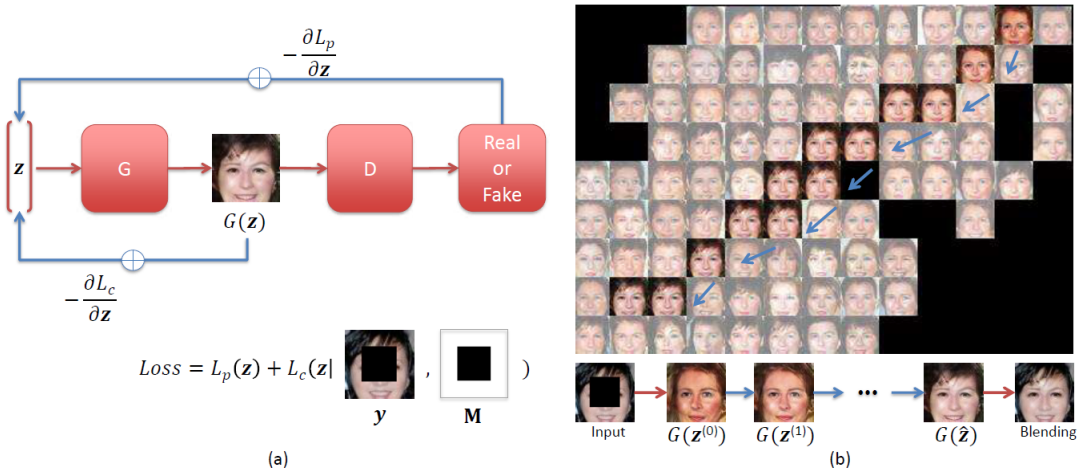

FIGURE 2.16. a) shows the proposed GAN network proposed by Semantic Inpainting and b) shows the encoding manifold that is traversed when iteratively updating $\hat{z}$ using backpropagation. $\mathbf{z}^{(0)}$ will be randomly initialized, $\mathbf{z}^{(k)}$ denotes the k^{th} iteration, and $\hat{z}$ denotes the final solution [**19**]

Figure 2.16 may be implemented by first considering equation 2.48.

$$\hat{z} = \arg\min_{z} \{\mathcal{L}_c(z|y, M) + \mathcal{L}_p(z)\} \tag{2.48}$$

where $\mathcal{L}_c$ denotes the "content loss", y denotes the image with lost data due to a mask M, and $\mathcal{L}_p$ denotes the loss of the prior generated image (penalizing unrealistic images). Furthermore, we have the content loss defined by equation 2.49.

$$\mathcal{L}_c(z|y, M) = \| \mathbf{W} \odot (G(z) - y) \|_1 \tag{2.49}$$

where $G(z)$ denotes the output of the generative network given the latest iteration of z, $\odot$ denotes an element wise multiplication, and $\mathbf{W}$ is defined by equation 2.50.

$$W_i = \begin{cases} \sum_{j \in N(i)} \frac{(1-M_j)}{|N(i)|} & if M_i \neq 0, \\ 0 & if M_i = 0 \end{cases} \tag{2.50}$$

where i denotes the pixel index, and $N(i)$ the set of neighbors to pixel i in a local window, and $|N(i)|$ denotes the cardinality of $N(i)$ [**19**]. Semantic Inpainting uses a window size = 7 [**19**].

The prior loss, $\mathcal{L}_p$, is the second term in equation 2.48 and is defined by equation 2.51.

$$\mathcal{L}_p(z) = \lambda log(1 - D(G(z))) \tag{2.51}$$

where $D(\cdot)$ denotes the discriminator network, and λ denotes a parameter used by Semantic Inpainting to balance between the two loss functions, $\mathcal{L}_c$ and $\mathcal{L}_p$. Without taking the prior loss function into account when iteratively solving for $\hat{z}$, the final solution may result in a perceptually implausible result [**19**]. For estimating our dataset, we will keep $\mathcal{L}_p$ in our iterative solution.

Finally, after iteratively solving for $\hat{z}$, Semantic Inpainting may be tested simply by running $G(\hat{z})$ and re-inserting y. However, [**19**] determined that the output of the trained generative network may not correctly predict the same intensities as the surrounding pixels in y. Therefore, a Poisson blending is performed on the output for a final result described in equation 2.52.

$$\hat{x} = \arg\min_{x} \| \nabla x - \nabla G(\hat{z}) \|_2^2 \; s.t. x_i = y_i for M_i = 1 \tag{2.52}$$

where ∇ denotes the gradient operator.

2.4.5. Machine Learning Input.

2.4.5.1. *Super Resolution Input.* SR algorithms main objective is to read in a low-resolution image and expand on the image while still maintaining the same detail as accurately as possible. Therefore, to test the estimation performance on different SR algorithms, an image was generated from RP_{sparse} based on the watt and dBm power levels. The size of each SR input image was determined by the sparsity of each test, and the reduction of the dataset was determined by equation 2.53.

$$Divisor = \left\lfloor \sqrt{\frac{RP_{truh}}{RP_{sparse}}} \right\rfloor \tag{2.53}$$

where RP_{truth} represents the total count of power measurements in the total dataset, and RP_{sparse} represents the total count of power measurements in the sparse dataset. The Divisor calculated in equation 2.53 shows how many power measurements from the ground truth to average together along each axis and form a low-resolution input for the SR algorithms.

After the ground truth has been averaged together based on equation 2.53, then the averaged data is used for generating an 8-bit and grayscale image. The minimum and maximum power levels need to be saved along with each SR image such that the up-scaled output from the SR algorithm may convert 8-bit pixel values back to power levels. Some noise is introduced to the SR algorithms in the form of encoding 64-bit floating-point power measurements to integer 8-bit pixel color values. Figure 2.17 shows an input image with a Divisor factor of 5 (25 ground truth measurements averaged together) before (left) and after (right) being expanded from SR algorithms. The top two photos show grayscale images of RF measurements in watt, and the bottom two photos show grayscale images of RF measurements in dBm.

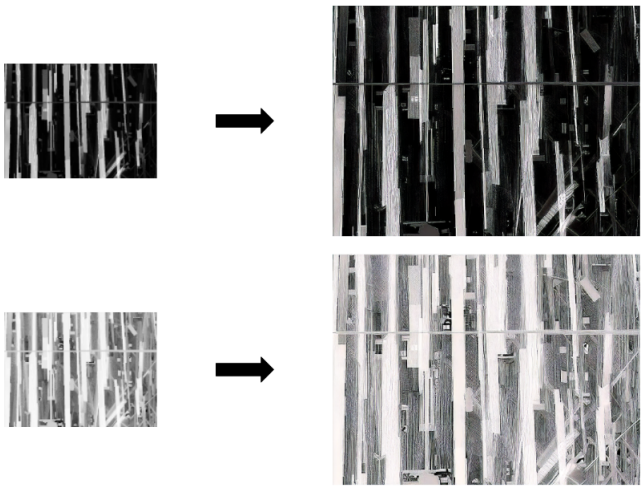

FIGURE 2.17. SR algorithm input and output. The left photos show grayscale images of the RF ground truth averaged down to 1/25 the amount of data. The right photos show grayscale images of SR algorithm output. The top photos were generated with watt data and the bottom photos were generated with dBm data. Proportional sizes of left and right photos are not accurate, in order to have an appropriately sized figure.

2.4.5.2. *ISTA-Net Input.* ISTA-Net's algorithm generated RP_{sparse} by selecting RPs from RP_{truth} based on the selected CS ratio. Therefore we simply provided ISTA-Net with an image of the entire dataset's watt and dBm measurements.

2.4.5.3. *Semantic Inpainting Input.* Semantic Inpainting was designed to accept a maximum sized grayscale image with 96×96 pixels. Moreover, Semantic Inpainting does not perform any up-scaling or growth to the input image as the SR algorithms do. Therefore, we split up the ground truth into multiple blocks of 96×96 RPs from RP_{truth}. The watt and dBm measurements of an input block were converted to grayscale images for processing.

2.4.6. Machine Learning Output.

2.4.6.1. *Super Resolution Output.* Some SR algorithms worked with 24-bit color images. Therefore, we needed to convert the 24-bit color images back to 8-bit grayscale images before converting the data back to power levels for performance measurements. This was accomplished with Python's Pillow library which converts color images to grayscale images with the ITU-R 601-2 luma transform.

Additionally, due to the nature of some of these SR algorithms, the final output image may have been too small or too large for a direct comparison to the ground truth. Mismatched dataset sizes were caused by the limited integer expansion of each axis for a given low-resolution image. If the output image were smaller than the ground truth, then the SR algorithm was simply applied to the output image again until a large enough image was generated. If the output image was larger than the ground truth, then evenly spaced rows/columns were identified and averaged with their neighboring row/column. Averaging the rows/columns were executed until the SR output image matched the size of our ground truth exactly.

2.4.6.2. *ISTA-Net Output.* The output of ISTA-Net matched the input image size exactly. The output of ISTA-Net is a 24-bit color image which was converted back to an 8-bit grayscale image with the ITU-R 601-2 luma transform.

2.4.6.3. *Semantic Inpainting Output.* The output of Semantic Inpainting matched the input image size exactly. The output of Semantic Inpainting is a 24-bit color image which was converted back to an 8-bit grayscale image with the ITU-R 601-2 luma transform.

CHAPTER 3

Performance Evaluation

This chapter will define key metrics for evaluating performance in Section 3.1, discuss how different key parameters are chosen in Section 3.3, discuss the performance of all estimation methods in Section 3.4 and finally discuss the processing time of all estimation methods in Section 3.5. The performance and processing time discussed in this chapter shows that some methods perform more favorably than others; this is true for the interpolation methods and the ML methods. Moreover, localization shows optimal performance and processing time across all methods with smaller locales. IDW and ISTA-Net provided the best overall performance for the interpolation and ML methods respectively.

3.1. Performance Metrics

This section will discuss the key performance metrics for comparing the estimation methods.

3.1.1. Fractional Error. The fractional error is defined by equation 3.1.

$$FE = \frac{|RP_{truth,i} - RP_{est,i}|}{RP_{truth,i}} \tag{3.1}$$

where $RP_{truth,i}, RP_{est,i}$ refer to the i^{th} RP in the ground truth and estimation respectively. Furthermore, we can define the average fractional error by taking the mean of all fractional errors, equation 3.2.

$$AFE = mean(FE) = mean\left(\frac{|RP_{truth,i} - RP_{est,i}|}{RP_{truth,i}}\right) \tag{3.2}$$

3.1.2. Root Mean Square Error. First, we defined the Squared Error (SE) by equation 3.3.

$$SE = (RP_{truth,i} - RP_{est,i})^2 \tag{3.3}$$

Then equations 3.4 and 3.5 define the Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) respectively.

$$MSE = mean\left((RP_{truth,i} - RP_{est,i})^2\right) \tag{3.4}$$

$$RMSE = \sqrt{mean\left((RP_{truth,i} - RP_{est,i})^2\right)} \tag{3.5}$$

Finally the Normalized RMSE (NRMSE) is defined by equation 3.6.

$$NRMSE = \frac{RMSE}{RP_{truth,max} - RP_{truth,min}} \tag{3.6}$$

3.1.3. Peak Signal to Noise Ratio. Peak Signal to Noise Ratio is defined by equation 3.7.

$$PSNR = 20 * log_{10}\left(\frac{max(RP_{truth})}{RMSE}\right) \tag{3.7}$$

where $max(RP_{truth})$ is the maximum possible value in the ground truth.

3.2. Dataset Reduction

3.2.1. Interpolation Methods. Due to the size of our datasets and the complexities of some algorithms, we needed to reduce our datasets primarily for ensuring we have feasible processing times.

3.2.1.1. *APL's Dataset.* The JHU APL dataset used for our research covered a large region and was very dense. Therefore the results presented in this section will be processed with $(1/100)^{th}$ of the original ground truth. Figure 3.1 shows the block of ground truth that is used for all the interpolation results presented in this chapter unless otherwise stated.

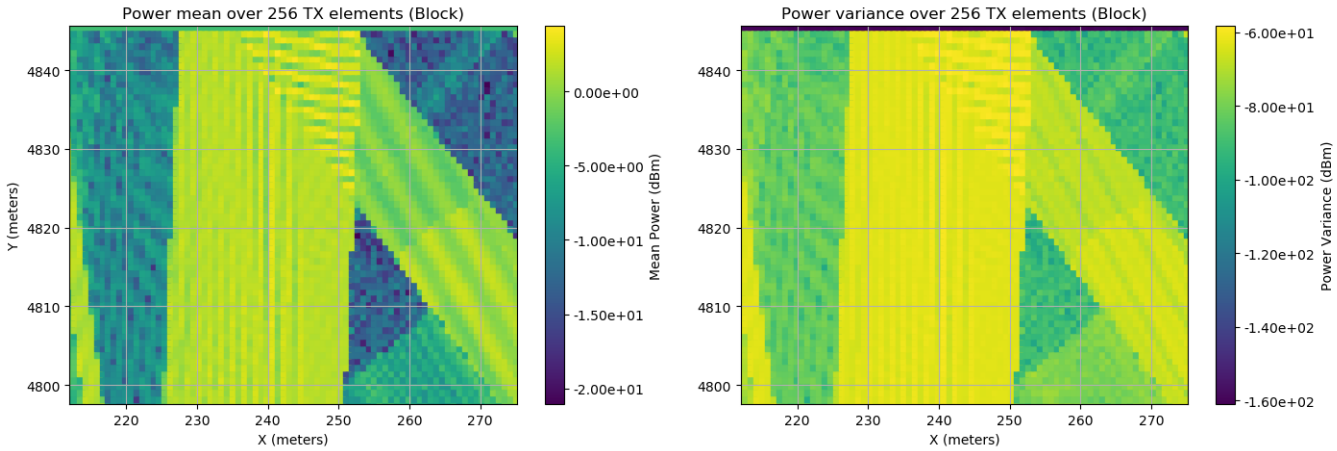

FIGURE 3.1. The block of ground truth used for interpolation estimations. This block size was selected to allow feasible processing times. This block of the ground truth was selected for the signal diversity.

3.2.1.2. *AERPAW's Dataset.* This dataset is too sparse and not suitable for the ML algorithms.

3.2.2. Machine Learning Methods. There is no concern for processing time with the ML algorithms. As previously stated in section 2.4.5, all ML methods were tested with an image generated from watt measurements and dBm measurements.

3.2.2.1. *APL's Dataset.* Semantic Inpainting will only accept a 96×96 pixel image, so we may only consider blocks of 96×96 RPs from RP_{truth}. Therefore, all the ML methods will be tested with a block of 96×96 RPs from RP_{truth}. Figure 3.2 shows the 50^{th} block of 96×96 RPs in RP_{truth}. This block was chosen for the diverse RF power measurements, similarly to why we selected the block shown in figure 3.1.

However, the performance on the 50^{th} block seemed quite poor. This result prompted us to also test the performance of the 13^{th} block. This block was selected due to how smooth and consistent the RF power measurements are. Figure 3.3 shows the RF power measurements for the 13^{th} block of RP_{truth}.

3.2.2.2. *AERPAW's Dataset.* This dataset is too sparse and not suitable for the ML algorithms.

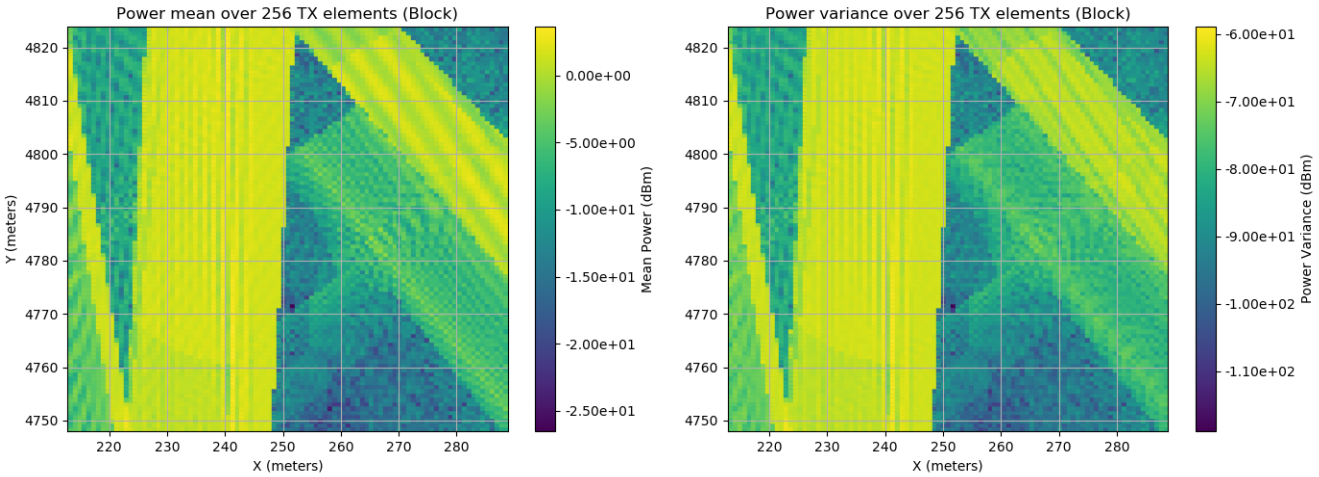

FIGURE 3.2. The 50^{th} block of 96×96 RPs in RP_{truth} used for ML estimations. This block size is limited by the max input for Semantic Inpainting. This block of the ground truth was selected for the signal diversity.

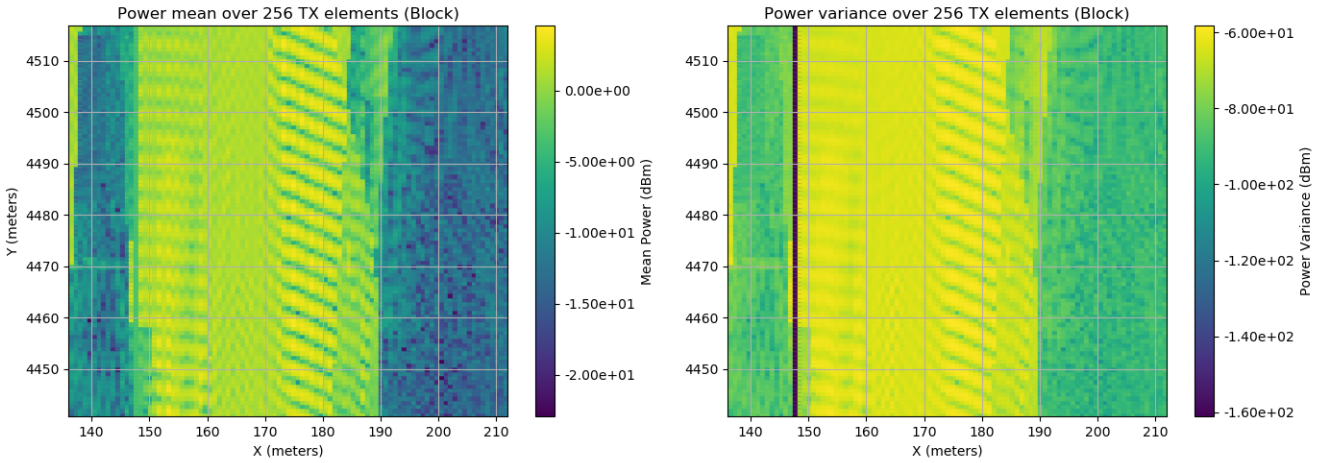

FIGURE 3.3. The 13^{th} block of 96×96 RPs in RP_{truth} used for ML estimations. This block of the ground truth was selected for signal uniformity.

3.3. Parameter Consideration

The localization diameter and some interpolation methods had parameters without a clear indication for how to set them. In this section we will discuss how these parameters were explored.

3.3.1. LDPL-MBI Power Parameter. Figure 3.4 shows how the performance of LDPL-MBI changes when adjusting the power parameter, p.

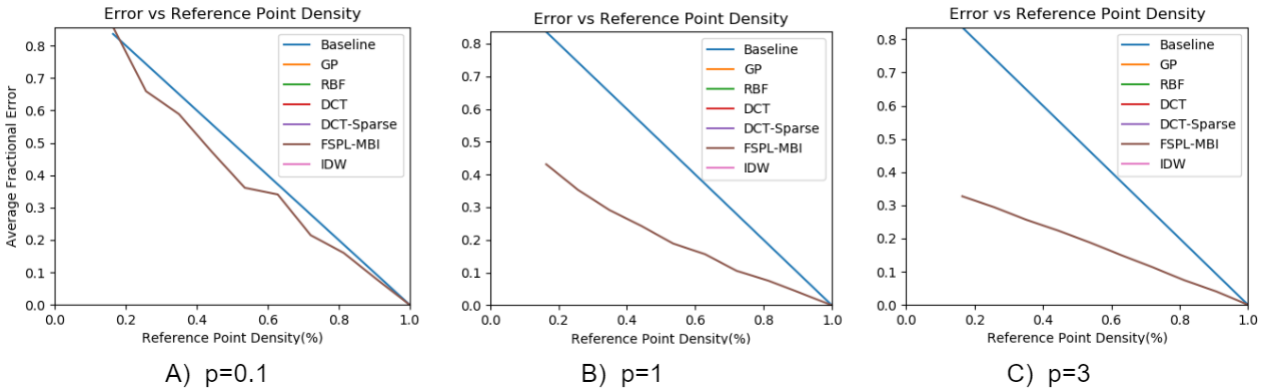

FIGURE 3.4. LDPL-MBI estimation performance with respect to setting the power parameter, p. Figures A to C shows increasingly larger values of p where the performance increases linearly with p increasing.

As the power parameter increases, the performance of the estimation increases. Around $p = 3$ the performance increase starts to plateau with increased p. Furthermore, we also observed that for large values of p, such as $p = 10$, the performance starts to suffer.

3.3.2. IDW System Parameter. Figure 3.5 shows how the performance of IDW changes when adjusting the system parameter, λ.

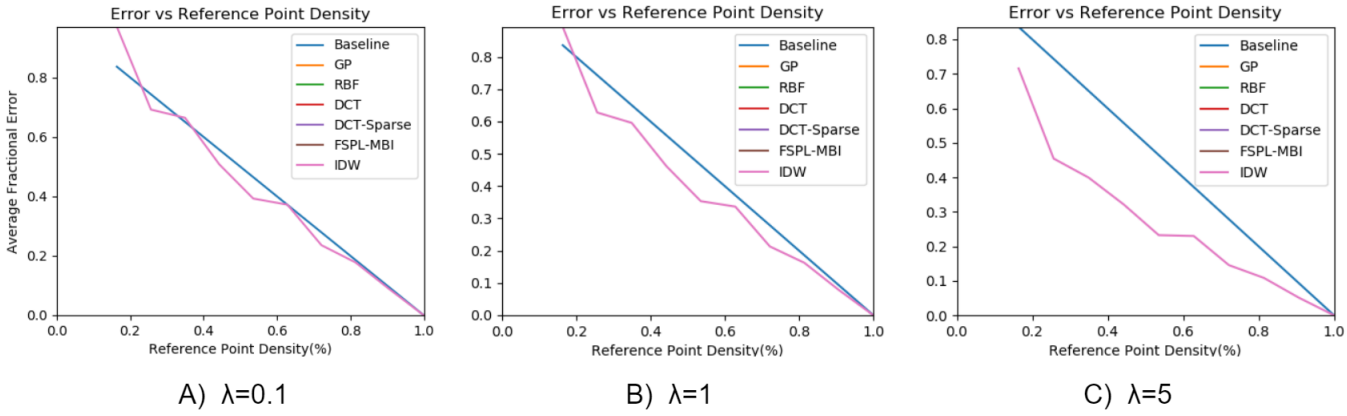

FIGURE 3.5. IDW estimation performance with respect to setting the system parameter, λ. Figures A to C shows increasingly larger values of λ where the performance increases linearly with λ.

As the system parameter increases, the performance of the estimation increases. We have observed that any value greater than $\lambda = 5$ or less than $\lambda = 0.1$ results in similar performance to $\lambda = 5$ and $\lambda = 0.1$ respectively.

3.3.3. Interpolation Localization. We discovered that localization of our interpolation methods yielded better results, but how much we needed to localize our methods was not clear. In order to determine how localized we wanted our interpolation method for the best performance, we plotted the performance of our interpolation methods with sparse dataset density on one axis and varying localization diameters on the other axis, figure 3.6.

What can be observed from figure 3.6 is that our interpolation methods perform the best when localization is minimized. One caveat to this observation is that if we had non-uniformly spaced RP_{sparse}, there would be a risk of making EP_{sparse} localization so small such that some EP_{sparse} will have no RP_{sparse} in their localized diameter. In this case, we suspect that performance will start to dramatically decrease. Fortunately for our sparse datasets, we have uniformly distributed RP_{sparse} and will not run into this issue. For the performance discussed in section 3.4, a localization diameter of $d = 6m$ was selected.

Figure 3.7 shows an intriguing behavior exhibited by the IDW interpolation method when adjusting the system parameter.

As the system parameter is set sufficiently large enough, localization no longer has an impact on the performance of our estimations.

3.4. Estimation Performance

Figure 3.8 shows the performance of all interpolation methods described in Chapter 2.

All of the legend labels should be clear after discussing the different interpolation methods. Additionally, a baseline is plotted in the performance graphs which consist of the performance when no estimation is computed. In this case, we are graphing the performance of strictly the sparse dataset padded with zeros. Figure 3.8 shows the same error, but with different metrics to consider. Furthermore, the performance shown in figure 3.8 was generated from 10 different RP_{sparse} selected randomly and then averaged together.

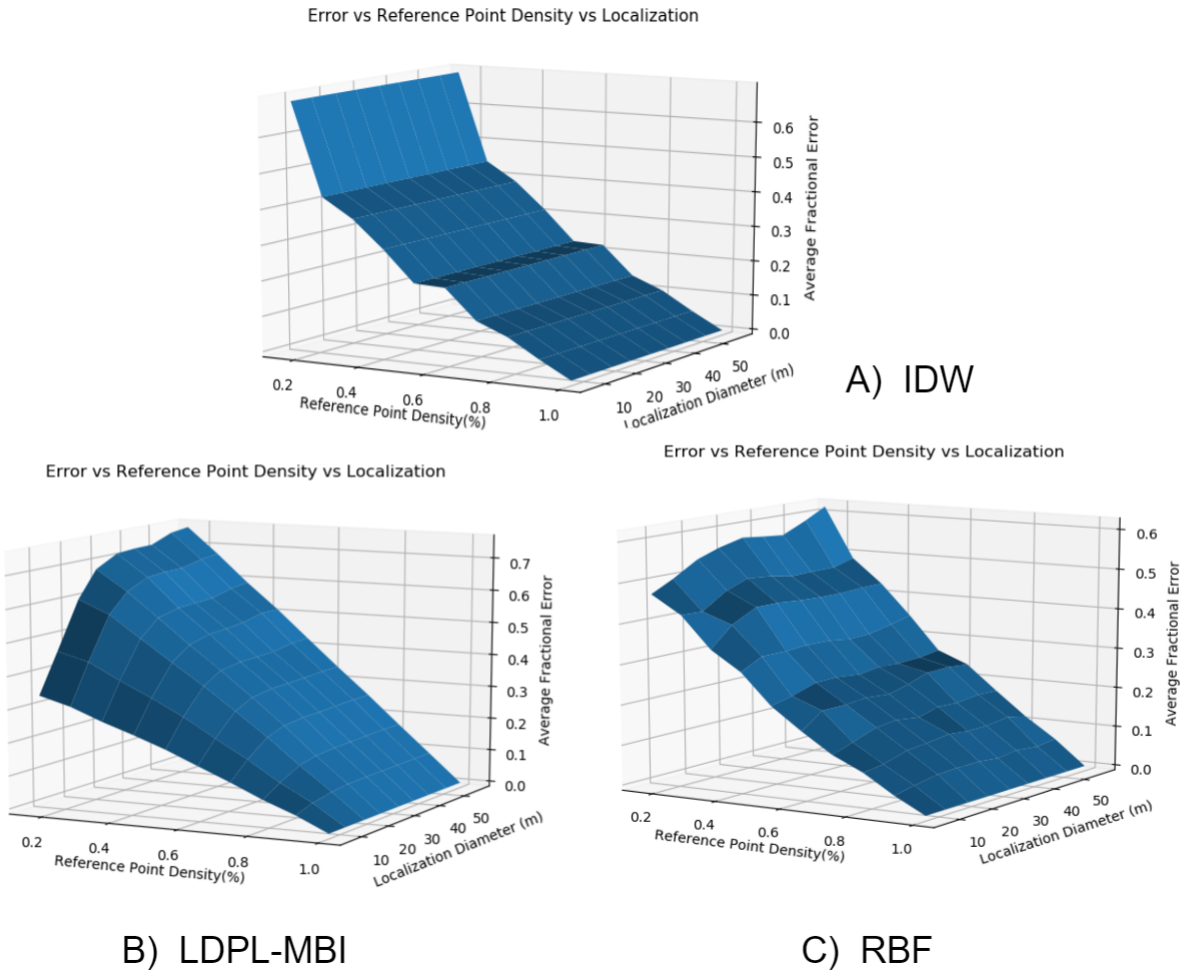

FIGURE 3.6. Interpolation performance as we sweep through increasing orders of localization. This graph shows how lower orders of localization and higher RP_{sparse} densities both produce better performance. More importantly, this graph shows that there is no dependence between localization order and RP_{sparse} density.

We also wanted to consider the performance when different methods of selecting RP_{sparse} were used, as described in Chapter 2. Figure 3.9 shows how well the interpolation methods performed with different methods for selecting our sparse dataset.

The performance data shown in figure 3.9 was generated from averaging 10 random RP_{sparse} (A), a single grid RP_{sparse} (B) and averaging 10 MDA RP_{sparse} (C). When RP_{sparse} is selected at random or with MDA, then averaging the performance of multiple RP_{sparse} will approach the true mean performance of a given density. However, when RP_{sparse} is selected with the grid method,

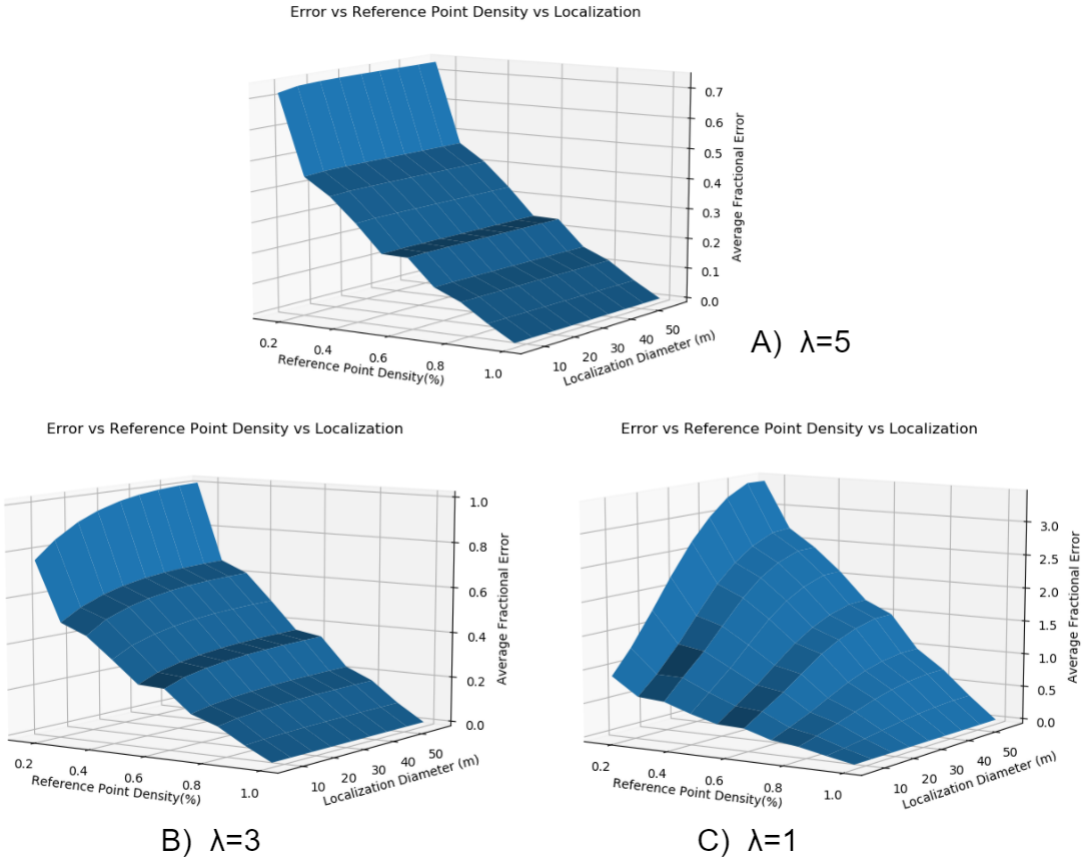

FIGURE 3.7. Sufficiently large λ for the IDW method will generate consistent estimation performance regardless of the localization order.

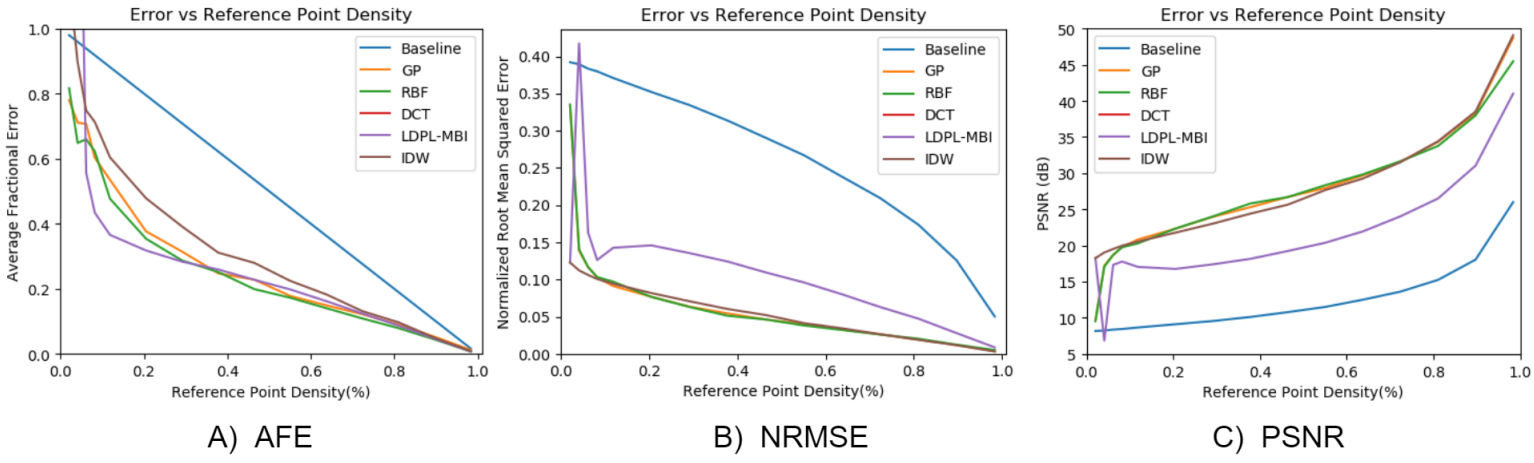

FIGURE 3.8. The error of the interpolation methods. Figure A shows AFE, figure B shows the NRMSE, and figure C shows the PSNR.

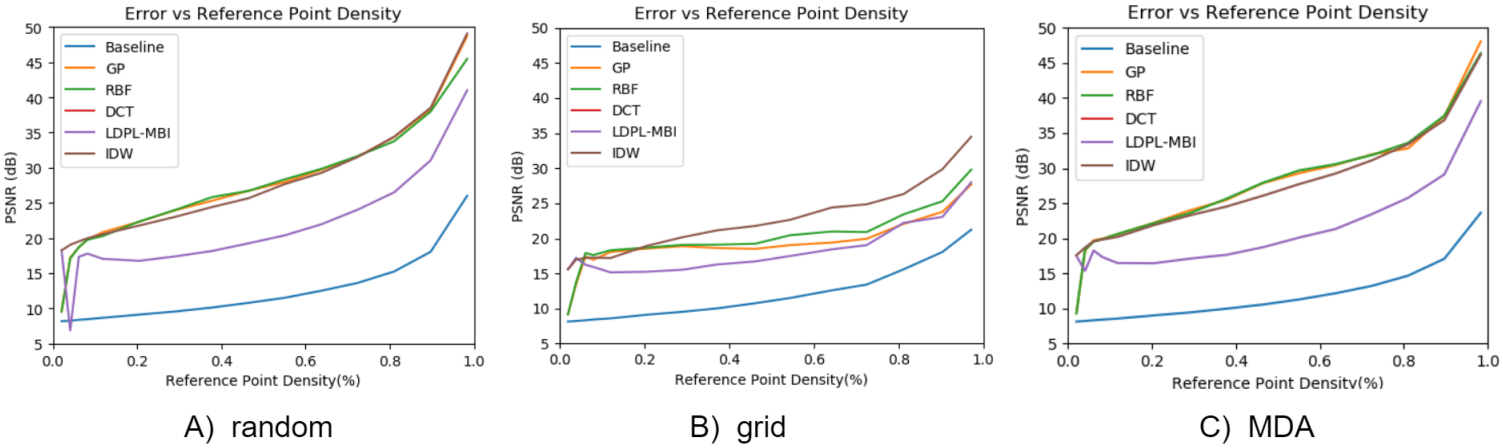

FIGURE 3.9. The interpolation estimation performance for the three ways to select RP_{sparse} discussed in Chapter 2.

then RP_{sparse} will never change for a given density and averaging performance will be the same as a single estimation. Figure 3.9 shows that the GP is the most affected by how we select RP_{sparse}. RBF and MBI are slightly affected by how RP_{sparse} is selected, and IDW shows the most tolerance to how RP_{sparse} is selected.

Figures 3.10 and 3.11 shows the performance for the different ML methods when estimating the 13^{th} and 50^{th} block of 96×96 RPs from RP_{truth}.

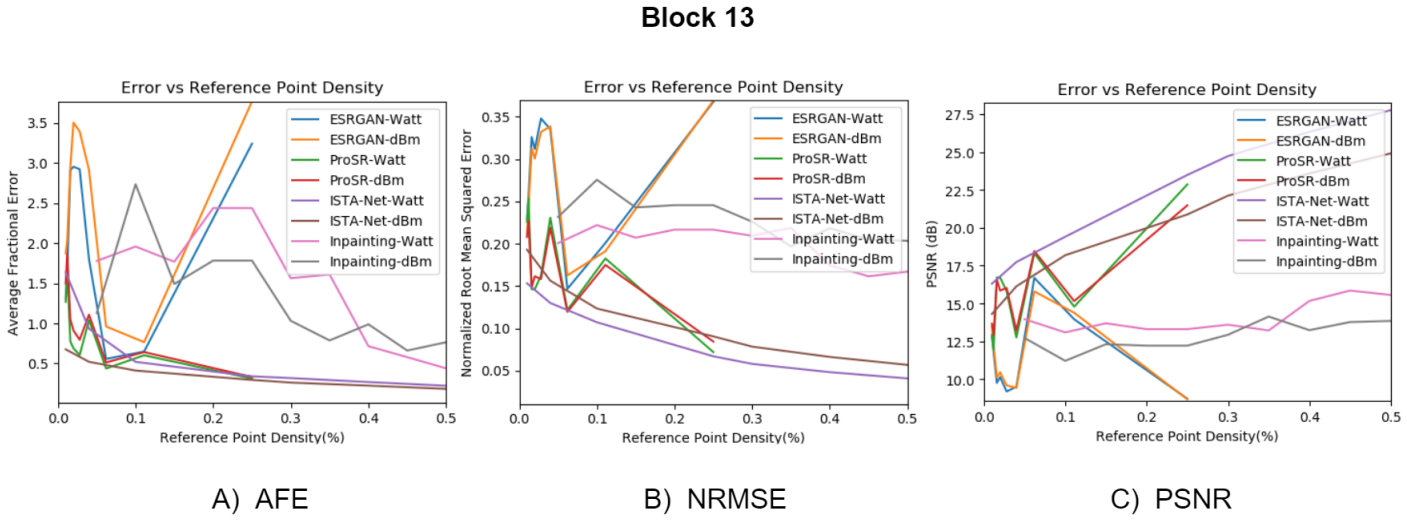

FIGURE 3.10. The error of the ML methods when estimating the 13^{th} block. Figure A shows AFE, figure B shows the NRMSE, and figure C shows the PSNR.

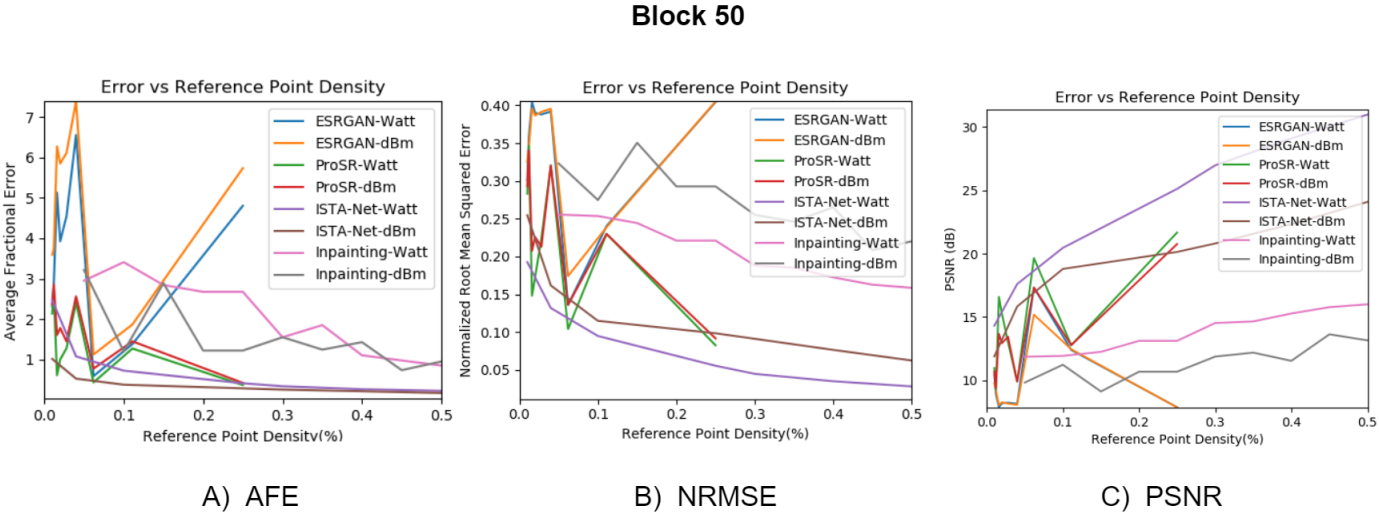

FIGURE 3.11. The error of the ML methods when estimating the 50^{th} block. Figure A shows AFE, figure B shows the NRMSE, and figure C shows the PSNR.

RP_{sparse} density used for ESRGAN and ProSR has an upper bound of 25%, ISTA-Net has an upper bound of 50%, and Semantic Inpainting has an upper bound of 100% (these upper bounds are determined by how the algorithms are designed). From the results presented in figures 3.10 and 3.11, we can see that ISTA-Net clearly performs the best out of all the ML methods. ProSR shows promising performance as more data is added to RP_{sparse}, but has poor performance with small RP_{sparse} densities. Semantic Inpainting shows consistent performance as we add more data to RP_{sparse}, but still has much worse performance when compared to ISTA-Net. ESRGAN exhibits the worst overall performance, showing inconsistent trending and the worst accuracy of all ML methods. The trend in performance for any estimation method is expected to decrease in error as the denisty of RP_{sparse} increases. However, ESRGAN has severely inconsistent trending in performance than what we would expect.

We used pre-trained models for ESRGAN and ProSR. These models were trained on natural images. Moreover, as stated in sections 2.4.1 and 2.4.2, ESRGAN and ProSR algorithms are trained to upscale with high perceptual quality rather than accurate predictions of additional pixels based on the value of input pixels. We believe that these two reasons are the cause of severe inconsistent trending in ESRGAN's performance as well as some inconsistent trending in ProSR's performance.

3.5. Processing Time

Figure 3.12 shows the processing time required by the interpolation methods.

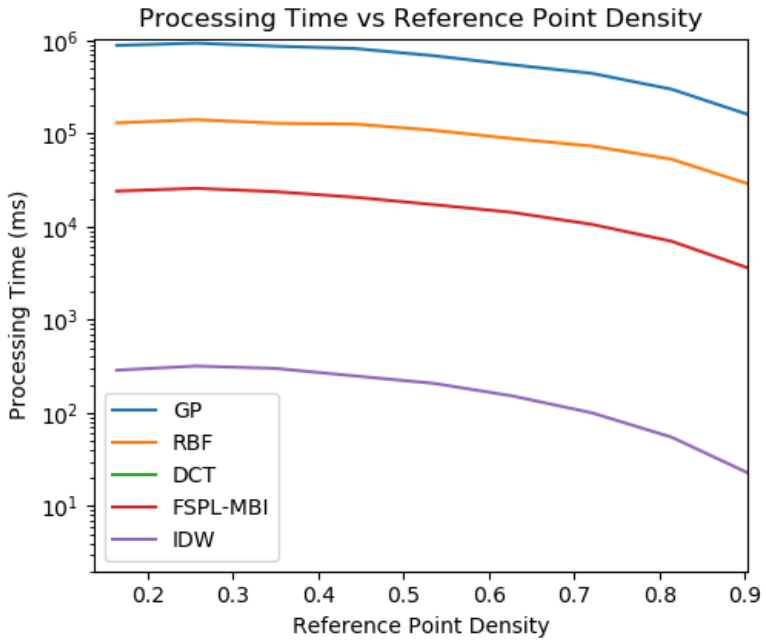

FIGURE 3.12. The processing time required by interpolation methods in milliseconds.

The GP method takes the longest amount of time to compute, an order of magnitude longer than the next interpolation method. Many of the graphs in this chapter do not include the GP as this method is prohibitively too long to compute. Furthermore, the methods requiring parameter fitting seemed to RP_{sparse} requiried the most processing time. The parameter fitting for these methods came in the form of maximizing or minimizing interpolation error with RP_{sparse} while adjusting the targeted parameter, and we believe that this process is what adds the most processing time in these methods. The IDW and DCT methods, which do not require parameter fitting, were the fastest interpolation methods tested. Figure 3.13 shows the processing time vs localization and RP_{sparse} density for the IDW, MBI, and RBF methods respectively.

Clearly, the more global an interpolation method becomes, the more processing time is required. Increased processing time with increased localization is expected as each interpolation method will now consider more RP_{sparse} when estimating each EP_{sparse}. Furthermore, the hump observed in

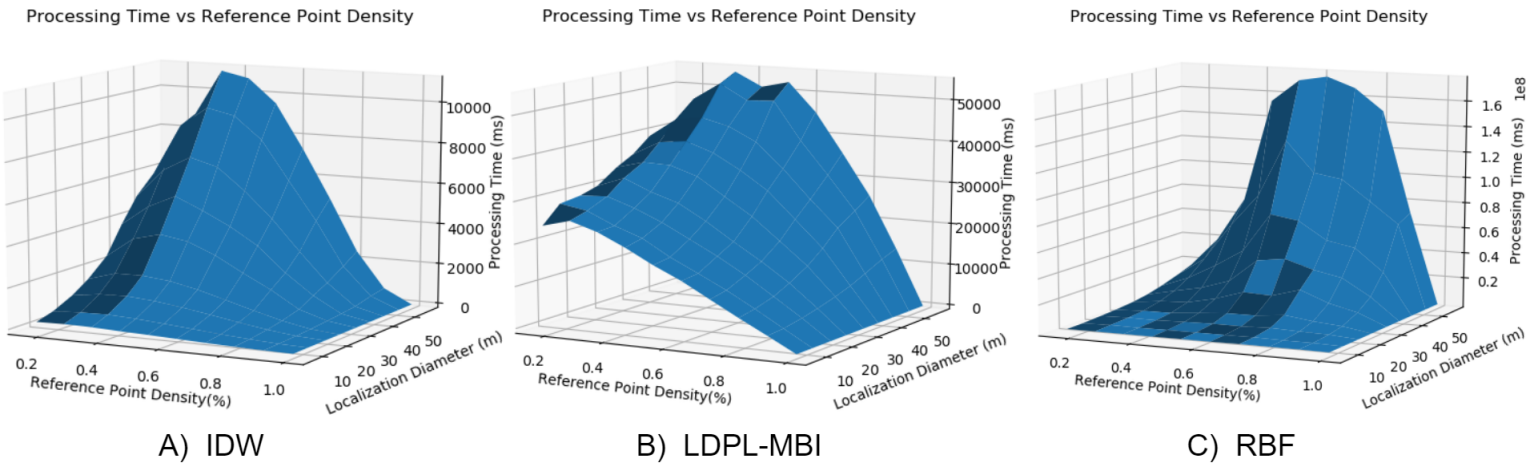

FIGURE 3.13. The processing time required by interpolation methods with varying localization diameters in milliseconds.

Figure 3.12 and Figure 3.13 along the RP_{sparse} density axis is an expected phenomenon. When the $EP_{sparse} \gg RP_{sparse}$, then there will be few unique localization groups (these are the outlined groups in a Voronoi diagram, such as in Figure 2.9). All EP_{sparse} belonging to a unique localization group are processed quickly with matrix operations rather than loop operations. As the RP_{sparse} count increases, so will the amount of unique localization groups, and each unique group adds an additional inefficient loop iteration of our interpolation methods. Finally, as $RP_{sparse} \gg EP_{sparse}$, then there will be much less EP_{sparse} to estimate as well as much less unique localization groups.

The processing time for all ML methods is about the same. The ML methods would estimate RP_{est} on the order of seconds to 10s of seconds, but always less than 1 minute. This processing time was true regardless of the chosen RP_{sparse} density.

CHAPTER 4

Conclusion and Future Directions

4.1. Conclusion

In this thesis, we explained different types of estimation methods for predicting a radio map. Our focus was on comparing different model-free estimation methods when provided with limited input data. We observed that many of the interpolation methods performed favorably for estimating RP_{est} with a sparse input, RP_{sparse}. From the performance shown, we show that IDW is an ideal candidate for processing our dataset. IDW showed the most favorable interpolation method accuracy, computation time, and tolerance to system parameters. The success of IDW also indicates that our dataset has some underlying graph properties and may be a good candidate for Graph Signal Processing (GSP) [**13**]. The GP was the worst candidate to consider for our application. GP had the worst accuracy, worst computation times, and the performance is susceptible to parameter variation. The absolute difference between all performance metrics in IDW and GP was surprising because no trade-offs need to be considered which is usually the case. The favorable performance with MBI suggests that our dataset has little RF interference in the LOS path between the TX and RX. MBI performing favorably suggest that our dataset is well described by RF models and our analysis may benefit from using models (as a hybrid model-free and model analysis).

Moreover, we explored different types of ML methods used for estimating RP_{est}. The performance results from all the ML methods clearly show that ISTA-Net performs much better than the rest of the ML methods. ISTA-Net performing the best was not a surprising observation, as ESRGAN, ProSR, and Semantic Inpainting are focused on generating perceptually accurate images rather than numerically accurate images. ESRGAN, ProSR, and Semantic Inpainting all used GANs to improve the perceptual quality of the output images while disregarding the numerical accuracy of the output images. Of the three worst performing ML methods, Semantic Inpainting did provide well trending results as RP_{sparse} density increased. Overall ISTA-Net was the best

ML method as all ML methods had the same processing time, ISTA-Net showed well trending results with increased RP_{sparse} densities, and ISTA-Net performed comparably to IDW (the best performing interpolation method). The favorable performance of ISTA-Net suggests that APL's dataset may be composed of a few basis from the basis matrix Ψ, as described in section 2.4.3.

4.2. Future Work

4.2.1. Model and Model-Free Hybrid Analysis. Our dataset may be well defined by RF propagation models such as the LDPL model. The LDPL-MBI interpolation method fits a parametric LDPL model to the dataset which performed well with estimating EP_{sparse}. This indicates that the TX had good LOS to the RX. Using simple RF propagation models without the need to consider complex RF interference may be a viable option for estimating our dataset. Furthermore, a model plus model-free hybrid analysis may show promising results.

4.2.2. Tailoring ISTA-Net's Φ. ISTA-Net comes with pre-optimized sampling matrices, Φ, for natural images. We may be able to achieve better performance from ISTA-Net if we tailor Φ for different RF environments (ie. tailor the RX placement).

4.2.3. Graph Signal Processing. Our dataset may be a good candidate for Graph Signal Processing. the IDW method was the best performing interpolation method. As defined in Chapter 2, IDW estimates EP_{sparse} based on the spatial relationship with RP_{sparse}. This observation implies that there is a strong correlation between the RSS data and the spatial rleationship of RXs. Therefore, this observation also indicates that our dataset should have some underlying graph features and yield favorable results when analyzing with GSP.

APPENDIX A

Interpolation Radio Map Estimation

Figure A.1 shows what a single estimation looks like when provided RP_{sparse}.

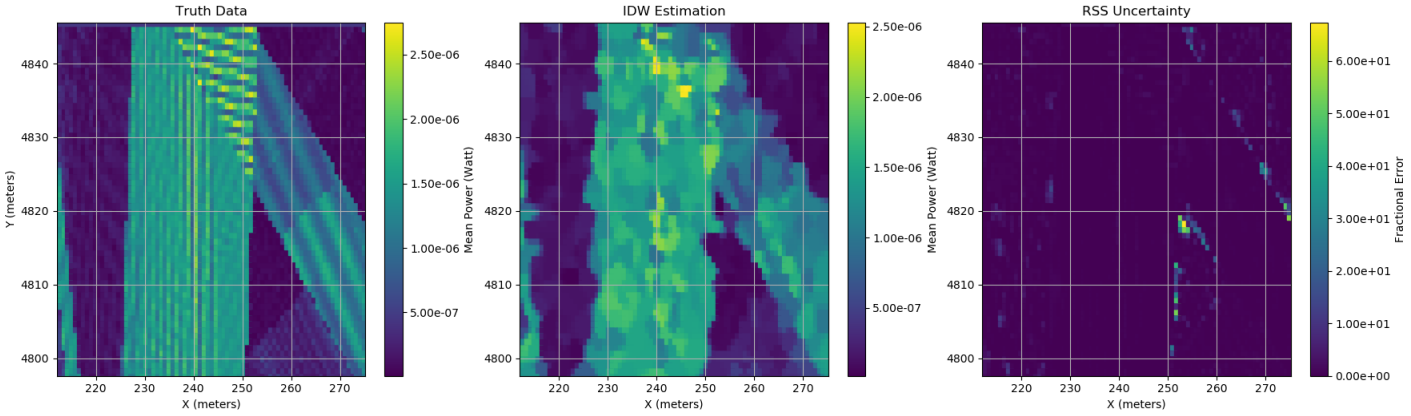

FIGURE A.1. Single IDW estimation where RP_{sparse} has a density of about 10%. The RSS Uncertainty plot shows how much FE there is at each RX location. From this plot, we can observe that RXs just to the bottom right of the center will contribute the most to our estimations AFE.

Figure A.1 shows an IDW estimation with a RP_{sparse} that has 500 RPs from RP_{truth}. In this case, RP_{truth} has about 4880 RPs, therefore RP_{sparse} density is about 10%. RP_{sparse} is shown in figure A.2.

The IDW estimation follows the same parameters as discussed in Chapter 3, $\lambda = 10$, and localization diameter is $6m$. The Voronoi diagram is apparent in the IDW estimation.

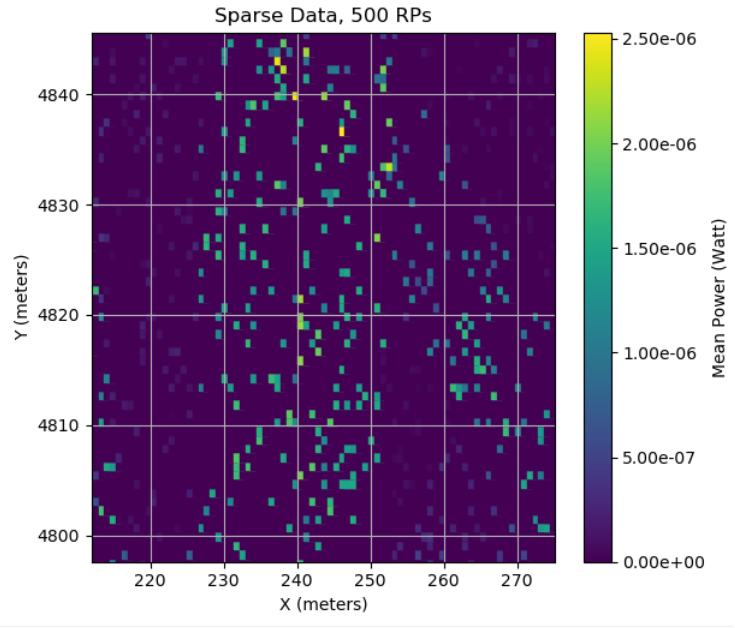

FIGURE A.2. RP_{sparse} used for the IDW estimation displayed in figure A.1. This RP_{sparse} contains 500 RPs from RP_{truth}

APPENDIX B

ML Radio Map Estimation

Figures B.1 to B.4 shows the progression of ESRGAN, ProSR, ISTA-Net, and Semantic Inpainting respectively. The figures are showing the output of each ML method as the desnity of RP_{sparse} is increased. These figures are showing the estimation of the 13th block of 96×96 RPs from RP_{truth}.

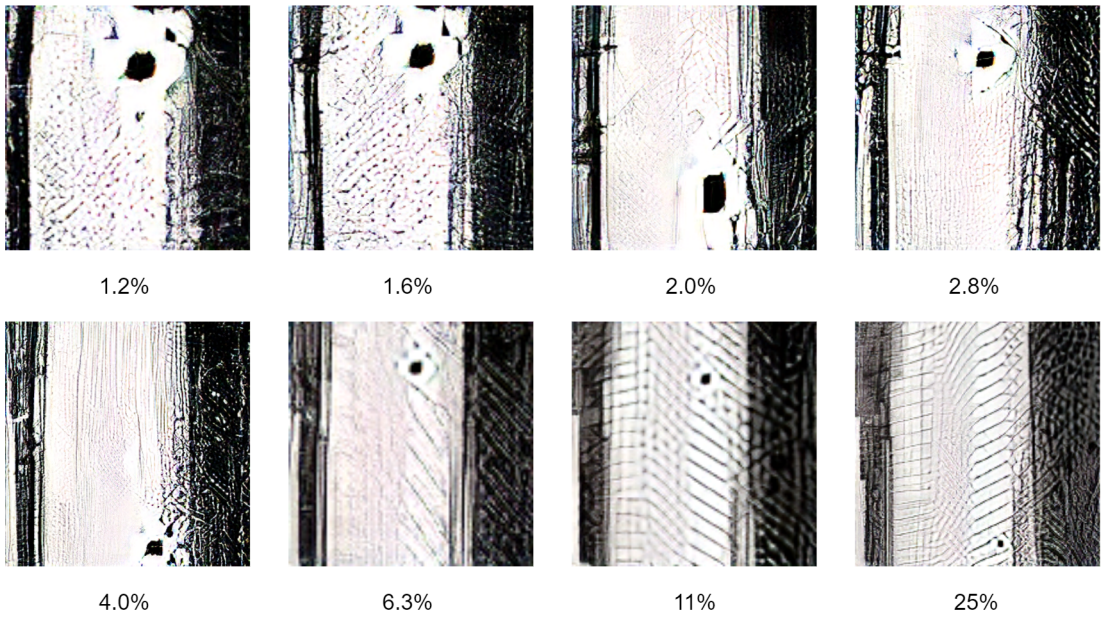

FIGURE B.1. Progression of ESRGAN estimating RP_{est} as the density of RP_{sparse} increases (as labeled by percentage values).

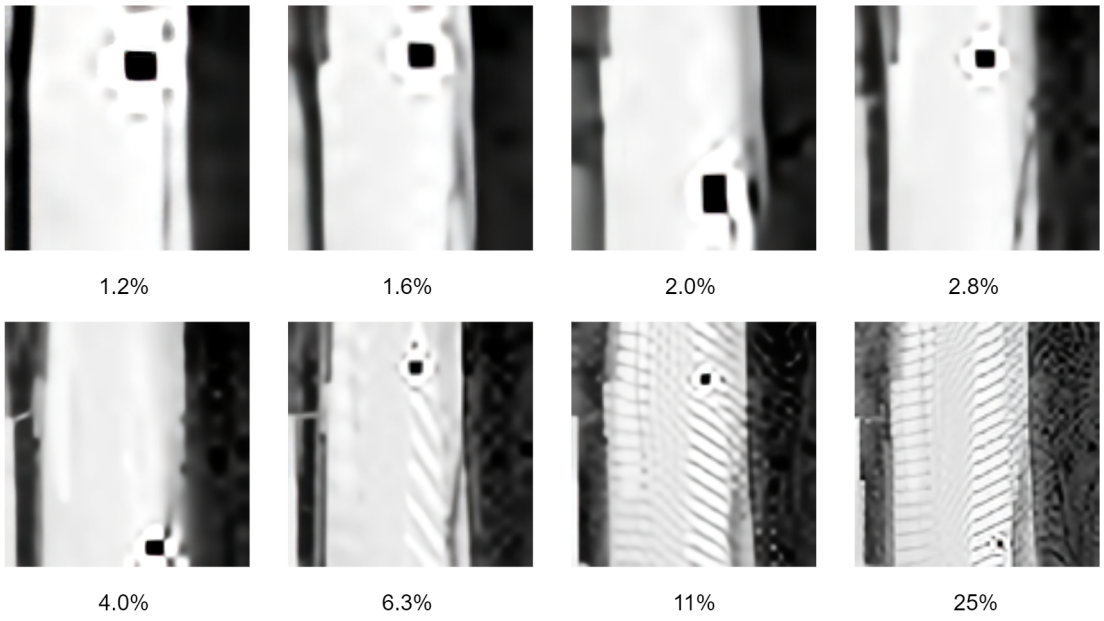

FIGURE B.2. Progression of ProSR estimating RP_{est} as the density of RP_{sparse} increases (as labeled by percentage values).

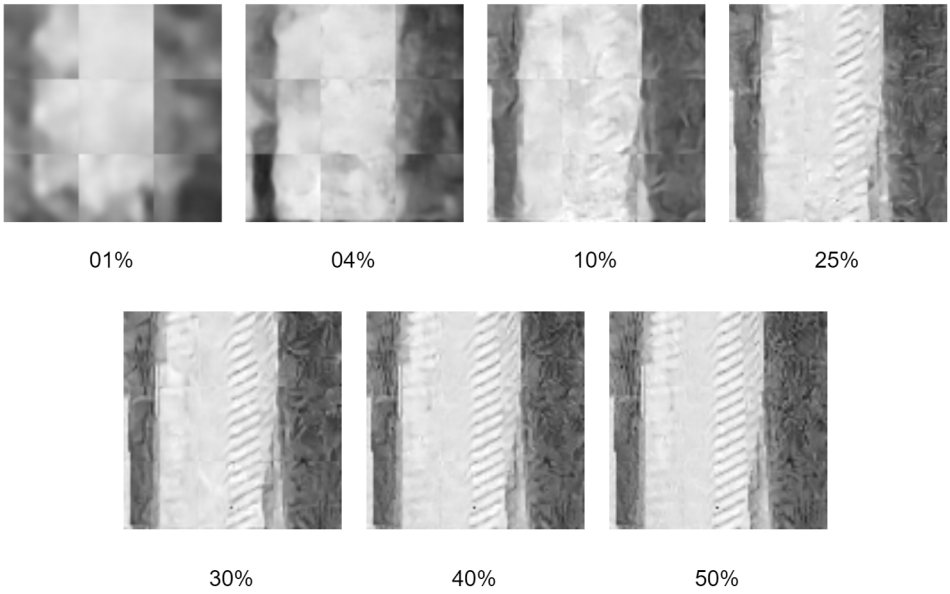

FIGURE B.3. Progression of ISTA-Net estimating RP_{est} as the density of RP_{sparse} increases (as labeled by percentage values).

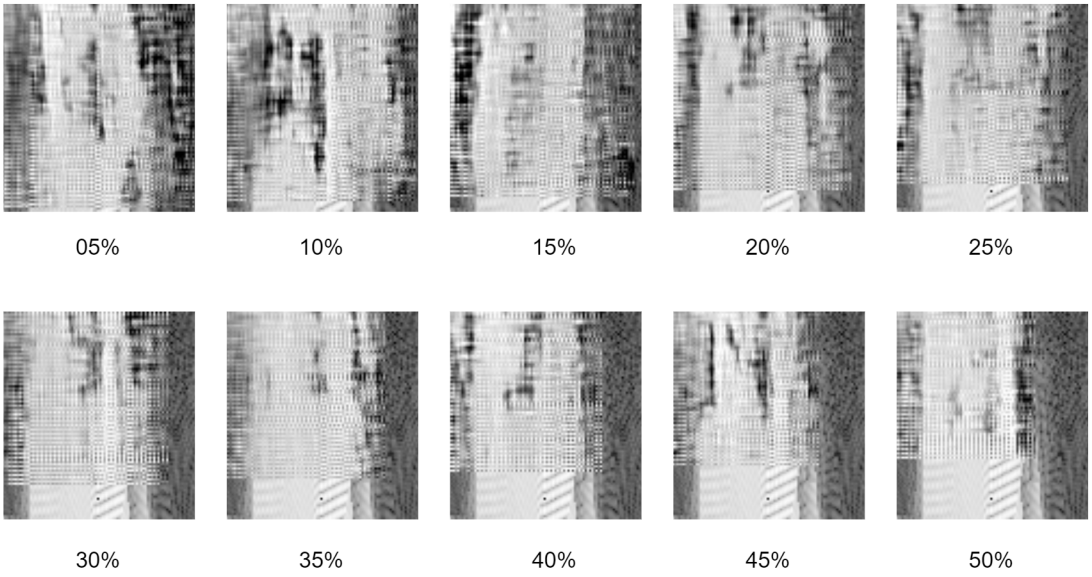

FIGURE B.4. Progression of Semantic Inpainting estimating RP_{est} as the density of RP_{sparse} increases (as labeled by percentage values).

Bibliography

[1] S. Bi, J. Lyu, Z. Ding, and R. Zhang, *Engineering radio maps for wireless resource management*, IEEE Wireless Communications, 26 (2019), pp. 133–141.

[2] E. Candès, *Compressive sampling*, 2006.

[3] B. Ferris, D. Haehnel, and D. Fox, *Gaussian processes for signal strength-based location estimation*, In proc. of robotics science and systems, (2006), pp. 1–1.

[4] G. Foster and N. Namazi, *Interpolation and gradient estimation of images using the discrete cosine transform*, in Proceedings of the Thirty-Fourth Southeastern Symposium on System Theory (Cat. No.02EX540), 2002, pp. 167–170.

[5] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, *Generative adversarial nets*, Advances in neural information processing systems, 27 (2014).

[6] O. Katircioğlu, H. Isel, O. Ceylan, F. Taraktas, and H. B. Yagci, *Comparing ray tracing, free space path loss and logarithmic distance path loss models in success of indoor localization with rssi*, in 2011 19thTelecommunications Forum (TELFOR) Proceedings of Papers, 2011, pp. 313–316.

[7] J. Krumm and J. Platt, *Minimizing calibration effort for an indoor 802.11 device location measurement system*, Microsoft Research, (2003), p. 8.

[8] S.-P. Kuo and Y.-C. Tseng, *Discriminant minimization search for large-scale rf-based localization systems*, IEEE Transactions on Mobile Computing, 10 (2010), pp. 291–304.

[9] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, *Photo-realistic single image super-resolution using a generative adversarial network*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), July 2017.

[10] M. Lee and D. Han, *Voronoi tessellation based interpolation method for wi-fi radio map construction*, IEEE Communications Letters, 16 (2012), pp. 404–407.

[11] J. Makhoul, *A fast cosine transform in one and two dimensions*, IEEE Transactions on Acoustics, Speech, and Signal Processing, 28 (1980), pp. 27–34.

[12] P. Medeaovic, M. Veletic, and Z. Blagojevic, *Wireless insite software verification via analysis and comparison of simulation and measurement results*, in 2012 Proceedings of the 35th International Convention MIPRO, 2012, pp. 776–781.

[13] A. Ortega, P. Frossard, J. Kovačević, J. M. F. Moura, and P. Vandergheynst, *Graph signal processing: Overview, challenges, and applications*, Proceedings of the IEEE, 106 (2018), pp. 808–828.

[14] C. Qiang, S. Van de Velde, and H. Steendam, *How to get the best out of your fingerprint database: hierarchical fingerprint indoor positioning for databases with variable density*, IEEE Access, (2019), pp. 1–1.

[15] C. E. Rasmussen and C. Williams, *Gaussian Processes for Machine Learning*, The MIT Press, 2006.

[16] K. Simonyan and A. Zisserman, *Very deep convolutional networks for large-scale image recognition*, arXiv preprint arXiv:1409.1556, (2014).

[17] X. Wang, K. Yu, S. Wu, J. Gu, Y. Liu, C. Dong, Y. Qiao, and C. Change Loy, *Esrgan: Enhanced super-resolution generative adversarial networks*, in Proceedings of the European Conference on Computer Vision (ECCV) Workshops, September 2018.

[18] Y. Wang, F. Perazzi, B. McWilliams, A. Sorkine-Hornung, O. Sorkine-Hornung, and C. Schroers, *A fully progressive approach to single-image super-resolution*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, June 2018.

[19] R. A. Yeh, C. Chen, T. Yian Lim, A. G. Schwing, M. Hasegawa-Johnson, and M. N. Do, *Semantic image inpainting with deep generative models*, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 5485–5493.

[20] J. Zhang and B. Ghanem, *Ista-net: Interpretable optimization-inspired deep network for image compressive sensing*, 2018.

[21] J. Zhang, D. Zhao, F. Jiang, and W. Gao, *Structural group sparse representation for image compressive sensing recovery*, in 2013 Data Compression Conference, 2013, pp. 331–340.