Manhattan Cutset Sampling and Sensor Networks

by

Matthew A. Prelee

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Electrical Engineering: Systems)
in the University of Michigan
2016

Doctoral Committee:

Professor David L. Neuhoff, Chair
Associate Professor Robert Dick
Professor Jeffrey A. Fessler
Professor Anna C. Gilbert
Professor Thrasyvoulos N. Pappas, Northwestern University

"The story so far:
In the beginning, the Universe was created.
This has made a lot of people very angry and been widely regarded as a bad move."
- Douglas Adams,
The Restaurant at the End of the Universe

2016

For my brother, John.

ACKNOWLEDGMENTS

I am very grateful for the friends, family, and colleagues that shaped me into the person I am today. This thesis would not have been possible without their love and support. Although this list is not exhaustive, I would like to highlight a few people in particular who have profoundly impacted my life while in graduate school.

First and foremost, I would like to thank my thesis advisor, David Neuhoff. Dave was the biggest reason I decided to pursue my graduate studies at Michigan. During my undergraduate senior year, he had suggested an interesting research project to me, the results of which are contained in this document. I also had an opportunity to meet with his current students at the time, and they had nothing but positive things to say about Dave's abilities as a researcher, as a teacher, and as a friend. Having now known and worked with Dave for over five years now, I can confidently say that he is indeed a stellar researcher, a patient teacher, and a good friend. Thank you David, for everything.

Next, I would like to thank my thesis committee: Thrasos Pappas, Jeff Fessler, Robert Dick, and Anna Gilbert. Without their influence, this thesis would not have been possible. Thrasos and his group at Northwestern were very helpful in developing and critiquing the early image processing algorithms presented in this thesis. Additionally, I had the pleasure of taking Jeff's image processing and image reconstruction courses while at Michigan. The image processing course inspired the sampling theorems presented in this thesis, and the convex optimization methods I learned in his special topics course are central to the most successful image reconstruction methods presented in this thesis. I also had the opportunity to take a special topics course taught by Anna in my second year, which helped inspire the cost functions and optimization methods in this work. Lastly, Robert was paramount in providing real-world insight into the sensor network problems presented in the final chapter. Once again, I'd like to thank all four of you for your suggestions and guidance.

While living in Ann Arbor, I was fortunate enough to befriend many wonderful people. To Nick, Madison and Mitch: I will forever remember my visits to the "House of Mitch," especially the yearly pig roasts, monthly beer tastings, and weekly Chipotle runs. To Rob and Mike: Thank you for being such great roommates. I have always enjoyed our many "deep" discussions, and the splitting of a Cottage Inn pizza (or two). For my EECS friends: Mai, Gopal, Brian, Parinaz, Ian, Aaron, JJ, and Steve: Thanks for always being around to share an idea, a walk, or a cup of coffee. I want to thank Steve, Sarah, Cassie, Monica, Giovani, Tim, Ashley and Annie for their friendship and guidance; your wisdom guided me through many a stressful hour. Also, I'd like to thank my friends Darby, Will, Claire, Colin, Nicholas, and Brian, with whom I have always enjoyed a late night board game (or two, or three!).

Finally, there are three family members that I would like to thank: My mother Betsy, my father Dan, and my brother John. I am so incredibly lucky to have such a supportive, loving family. My parents have always encouraged me to follow my passions over the course of my life, and that encouragement has always been a personal source of motivation. So to my parents: Again, thank you for your hard work and sacrifices. I am so glad that I can always turn to you two for life advice, and sometimes even a home-cooked meal. To John: Thank you for your love and support; I always look forward to our visits, when you and I can share in playing the latest video or board game. Once again, I love all three of you with all my heart.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF APPENDICES

ABSTRACT

Manhattan Cutset Sampling and Sensor Networks

by

Matthew A. Prelee

Chair: David L. Neuhoff

Cutset sampling is a new approach to acquiring two-dimensional data, i.e., images, where values are recorded densely along straight lines. This type of sampling is motivated by physical scenarios where data must be taken along straight paths, such as a boat taking water samples. Additionally, it may be possible to better reconstruct image edges using the dense amount of data collected on lines. Finally, an advantage of cutset sampling is in the design of wireless sensor networks. If battery-powered sensors are placed densely along straight lines, then the transmission energy required for communication between sensors can be reduced, thereby extending the network lifetime.

A special case of cutset sampling is Manhattan sampling, where data is recorded along evenly-spaced rows and columns. This thesis examines Manhattan sampling in three contexts. First, we prove a sampling theorem demonstrating an image can be perfectly reconstructed from Manhattan samples when its spectrum is bandlimited to the union of two Nyquist regions corresponding to the two lattices forming the Manhattan grid. An efficient "onion peeling" reconstruction method is provided, and we show that the Landau bound is achieved. This theorem is generalized to dimensions higher than two, where again signals are reconstructable from a Manhattan set if they are bandlimited to a union of Nyquist regions. Second, for non-bandlimited images, we present several algorithms for reconstructing natural images from Manhattan samples. The Locally Orthogonal Orientation Penalization (LOOP) algorithm is the best of the proposed

algorithms in both subjective quality and mean-squared error. The LOOP algorithm reconstructs images well in general, and outperforms competing algorithms for reconstruction from non-lattice samples. Finally, we study cutset networks, which are new placement topologies for wireless sensor networks. Assuming a power-law model for communication energy, we show that cutset networks offer reduced communication energy costs over lattice and random topologies. Additionally, when solving centralized and decentralized source localization problems, cutset networks offer reduced energy costs over other topologies for fixed sensor densities and localization accuracies. Finally, with the eventual goal of analyzing different cutset topologies, we analyze the energy per distance required for efficient long-distance communication in lattice networks.

CHAPTER 1

Introduction

The solution to any given problem in science, mathematics, and engineering often relies heavily on the quality and quantity of available data. If, indeed, the data is lacking in some manner, then it may be impossible find a unique solution, or satisfactorily evaluate a hypothesis. Thus, an important issue in any technical endeavor is that of *data acquisition*. In many applications where the data acquisition process can be controlled, a greedy data collection philosophy often dominates. This attitude is apparent when the law of large numbers is invoked, or a wide variety of test cases are used. Such strategies inspire confidence that a solution method is consistent and robust. However, resources are often limited, and a tradeoff appears between the data acquisition process and the quality of a solution. In other applications, the data acquisition process is fixed. In these cases, it is worthwhile to find a method that achieves the best possible solution given the data. All of these ideas are at the heart of this thesis.

The goal of this thesis is to investigate the strengths and weaknesses of a new geometric approach to acquiring data called *cutset sampling*, where data is recorded densely along straight lines or line segments. The term "cutset" is taken from graph theory, where a cutset is defined to be a set of nodes whose removal separates a graph into two or more disjoint sets. In a similar manner, if a two-dimensional function (i.e. an *image*) is sampled on one or more straight lines, then the planar domain of the function is separated into two or more disjoint sets. A specific example of cutset sampling is *Manhattan sampling*, where a two-dimensional function is sampled along evenly spaced rows and columns, thereby mimicking the grid-like geometry of city streets (see Figure 2.1). This idea is contrasted to conventional rectangular lattice sampling, where data is taken at evenly spaced points on a Cartesian grid, or random sampling, where sample points are chosen randomly in the plane. The former sampling approach is common in image upsampling and interpolation. The latter sampling approach often appears in sensor network problems, where data measurements are taken at randomly scattered wireless sensors. Both such applications are considered in this thesis.

1.1 Motivations for Cutset/Manhattan Sampling

At first glance, it seems rather counter-intuitive to investigate cutset or Manhattan sampling. After all, it seems like it would be easier to estimate two-dimensional phenomenon by having an even distribution of sampling points across an area of interest. Although this is true for many applications, there are some other factors to consider in favor of cutset sampling.

First, cutset sampling appears in applications that are restricted to collecting data along straight lines. Some examples include applications where data is collected by vehicles, which must necessarily travel along a continuous path. For example, many researchers in environmental science are interested in *aquatic hypoxia*, which is the depletion of oxygen in bodies of water. Severe hypoxia can lead to the death of fish and other animals in an ecosystem; therefore, it is of interest to measure oxygen levels in bodies of water. Such measurements can be taken by a boat moving through the water [3]. An investigation into cutset sampling can provide useful insight into how to sample and interpolate such data in future research endeavors.

Second, an advantage of cutset sampling is that there exists high correlation between neighboring samples, and this correlation can be exploited. For example, cutset sampling has already been shown to be useful in lossy and lossless image compression, particularly for bilevel images [4–7]. As a first step, these image compression techniques use arithmetic coding to compress neighboring pixels on a Manhattan grid. This technique is efficient because it exploits the high correlation between neighboring pixels. For lossless encoding applications, the remaining pixels are then also compressed conditioned on the pixel values on the Manhattan grid. For lossy compression, the remaining pixels are not encoded, but rather are estimated by the decoder given the pixels on the Manhattan grid. These methods are heavily based on graphical Markov random field models, where a Manhattan grid acts as a cutset, separating the image into blocks of pixels that are conditionally independent given the cutset. This property leads us to the coined term "cutset sampling."

Another motivation for cutset sampling is that it may allow better reconstruction edges and level sets of an image or function. This is especially important in the area of image processing, since human beings are often very sensitive to slight changes in edge information, such as the blurring or oversharpening of edges. As we will see in Chapter 3, by using an appropriate algorithm, images reconstructed from their Manhattan samples can retain much of their edge information

It is possible that Manhattan sampling or cutset sampling is optimal for representing or reconstructing certain classes of functions or images. If this class of images can be identified, then Manhattan or cutset sampling can be applied in these instances to great success.

Chapter 2 discusses in detail just such a class of images that can be perfectly reconstructed from their Manhattan grid samples.

Finally, the battery life of sensor networks can be extended greatly if the sensors are placed along straight lines, such as a Manhattan grid. To see why, first consider the fact that wireless transmission circuitry typically dominates the battery usage of sensor nodes in wireless sensor networks, while the actual sensing circuitry requires little to no energy ("sensing is cheap"). When nodes attempt to share their data, a multi-hop strategy is employed to relay messages through the network. Such a strategy is desirable because the decay of wireless transmission power over distance follows an inverse power law. If the energy overhead for turning on the transmission circuitry is negligible, then it will be cheaper to relay a message through the network than to communicate directly with a destination node. Under a power-law assumption for communication (see equation (4.1)), the transmission power per hop is directly proportional to the distance between neighboring sensors. If this inter-sensor spacing can be reduced, then the energy-per-hop can in turn be reduced, potentially reducing the overall energy usage of the network. It can be shown that when sensor nodes are placed along a Manhattan grid, the inter-sensor spacing between neighboring sensors is much smaller than that of a rectangular lattice network or random network for some fixed node density. Such arguments are described in more detail in Chapter 4. Chapter 4 also discusses energy transmission models in general, and gives conditions under which relaying is an optimal strategy, and when it is not.

1.2 Overview and Summary of Contributions

Cutset sampling is a relatively new area; as such, the majority of this thesis work is concerned specifically with the special case of Manhattan sampling. In some cases, we will also consider other common sampling patterns and sensor deployments, such as random sampling or lattice sampling. Overall, this thesis

1. Determines a frequency region such that images bandlimited to it can be reconstructed perfectly from their Manhattan samples, both in two and higher dimensions. This thesis also presents an efficient "onion-peeling" algorithm, and demonstrates that Manhattan sampling is optimal in the Landau sense.

2. Provides efficient methods for estimating non-bandlimited images as accurately as possible from samples on a Manhattan grid, and demonstrates that the best of these methods outperform existing methods for reconstructing images from non-lattice sampling patterns.

3. Demonstrates that Manhattan/cutset sensor networks require less communication energy to solve centralized and decentralized source localization problems at certain fixed sensor densities and desired estimation accuracies; additionally, the *Midpoint Algorithm* is proposed for decentralized source localization on a Manhattan grid, which attains lower energy costs than a competing decentralized localization algorithm.

4. Presents a general method for predicting the required energy-per-distance cost of long distance communication in a lattice sensor network when the model for transmission energy is *relay efficient*. In particular, we predict that efficient communication path only consist of one or two "hop types", and we also demonstrate that these predictions match the output of a shortest path algorithm.

We will now discuss these problems in more detail.

The first contribution, concerning (perfect) image reconstruction, is addressed in Chapter 2. In particular, we determine a class of two-dimensional continuous functions that are reconstructable from their Manhattan-grid samples. To solve this problem, it is useful to view a Manhattan grid as the union of two lattices, one densely sampled in the horizontal direction, and the other densely sampled in the vertical direction. It is then shown that the set of images whose spectra are bandlimited to the union of Nyquist regions corresponding to these two lattices can be perfectly reconstructed from their Manhattan samples, as shown in Figure 2.1(c). As far as we are aware, this is the first time it has been demonstrated that an image bandlimited to a union of Nyquist regions can be reconstructed from a union of lattices. Furthermore, it is shown that this set of images is maximal in the Landau sense [8]. This result is extended to the discrete infinite- and finite-support cases. Reconstruction algorithms are provided for all instances. In Section 2.4, this result is extended to the multi-dimensional case. This high-dimensional problem is interesting because Manhattan sampling sets take numerous forms in more than two dimensions, such as those shown in Figure 2.4. Again, the reconstructable set of images includes those multidimensional images bandlimited to a union of Nyquist regions corresponding to the lattices that make up the Manhattan sampling set. To aid us in the high-dimensional setting, *bi-step lattices* are introduced, which are lattices whose points are either spaced coarsely or densely along each dimension. The binary nature of bi-step lattices allows them to be characterized by binary *bi-step vectors*, where the ith element of a bi-step vector indicates whether the spacing of a bi-step lattice is coarse or dense along dimension i. Finally, a recursive "onion-peeling" algorithm is proposed for reconstruction. This onion-peeling algorithm is very elegant, and it can be performed using basic Fourier Transforms, filtering operations, and sampling operations.

The second contribution, concerning the development of algorithms for reconstructing arbitrary images from their Manhattan grid samples, is addressed in Chapter 3. To begin, two exploratory algorithms are reviewed (Sections 3.1 and 3.2) that are based on Markov Random field (MRF) models. Each algorithm first attempts to segment the entire image (both known and unknown pixels) into regions of similar pixels. The first algorithm is prior work that segments pixels into regions of similar intensity; the second algorithm models the image as piecewise planar plus noise, and tries to identify the planar regions and corresponding planes using the so-called *k-planes algorithm.* After these regions are identified, the textures within each region are modeled using a Gaussian MRF, and a linear MMSE estimator is used to identify the missing pixels. Segmentation approaches result in reconstructions that look very artificial, with a "color by numbers" look. However, both algorithms reconstruct sharp (hard) edges well, and the latter algorithm also reconstructs gradual (soft) edge-transitions well.

In Sections 3.3 and 3.4, we greatly improve upon the early methods by introducing the Orthogonal Gradient (OG) and Locally Orthogonal Orientation Penalization (LOOP) algorithms. Both algorithms are alternating algorithms that switch between solving a convex optimization problem and updating the parameters of the optimization problem. These two steps continue until the estimated image converges. The performance of both algorithms is examined in Section 3.5. Since our work is the first to consider reconstructing images from Manhattan sets, we compare our algorithms to methods that have been designed to reconstruct images from arbitrary sampling patterns [9,10]. We show that the OG and LOOP algorithms outperform their competition; in particular, the LOOP algorithm performs best, both subjectively and in terms of mean-squared error. We also note that both the OG and LOOP algorithms are flexible enough to reconstruct images from arbitrary sampling patterns. To demonstrate this, we apply the LOOP algorithm to the classic problem of lattice interpolation, and we find that the LOOP algorithm is competitive with a recent lattice interpolation method [11]. Overall, we believe that the LOOP algorithm is a very promising approach that can potentially be applied to a wide variety of image reconstruction applications. Furthermore, we demonstrate that the quality of edges reconstructed by the LOOP algorithm is similar to edges reconstructed from square lattice samples using edge-adaptive algorithms, such as [11].

Chapter 4 covers the third and fourth contributions regarding sensor networks. Specifically, the third contribution is covered in Sections 4.1 through 4.4, where under certain power law models for communicating a packet of data over a certain distance, e.g., a square-law, we demonstrate that Manhattan/cutset sensor networks require less communication energy to solve centralized and decentralized source localization problems at certain fixed sensor

densities and desired estimation accuracies. First, in Section 4.1, we provide formulas that estimate the smallest energy required to transmit a packet of data at a certain distance and angle through a cutset network. These formulas are in turn used to predict the smallest total energy required for all nodes in the network to communicate with a central hub. These formulas are compared to the output of a cheapest path algorithm (Dijkstra's Algorithm) and it is found that they approximate the true path costs to within a few percentage points of error. It is also found that the cutset networks require less energy than lattice networks, and the energy decreases as the integer cutset parameter k increases (for example, in a Manhattan network, k is the number of sensors per "square"). This gain in energy efficiency is desirable, but we would also like to see how the layout of a cutset network affects the performance of the sensor network in performing a signal processing task.

Beginning in Section 4.2, we consider the task of received signal strength (RSS)-based source localization on a sensor network, where the goal is to estimate the position of a source that is emitting electromagnetic or acoustic waves. We assume that each sensor in our sensor network obtains a noisy power measurement (Section 4.2), and these measurements can be used to produce a position estimate of the source. Our goal, therefore, will be to minimize the root mean-squared error between our source position estimate and the true source position. In the centralized estimation scenario, all measurements are communicated to a central hub for processing, where a method such as Maximum Likelihood Estimation (MLE) can be performed (Section 4.3). In the decentralized estimation scenario, sensors make local decisions about (a) whether a source is present, and (b) how to share their data with neighbors. A decentralized algorithm, such as the POCS algorithm [2], can be used; in the case of a Manhattan sensor network, we will propose the *Midpoint algorithm* (Section 4.4). However, as we demonstrate in both Section 4.3 and 4.4, that there is a tradeoff in estimation error and the communication energy required to compute the estimate. In the centralized case, we find that the hexagonal tessellation networks ("honeycomb" networks) allow for the greatest reduction in energy, at the expense of very little estimation error. In the decentralized case, It is shown that while using the POCS algorithm for localization, Manhattan sensor networks use less energy than the random network or lattice network, but at the cost of increased estimation error. The proposed Midpoint Algorithm further reduces the energy required to estimate the source, but again at reduced accuracy. Thus, a fundamental energy-vs-accuracy tradeoff is observed.

Finally, our fourth contribution is covered in Sections 4.5 and 4.6. With the goal of understanding the ability of a periodic network topology to provide a communication infrastructure (agnostic to the sensor network task), we present a general method for predicting the required energy-per-distance cost of long distance communication in a lattice sensor

network when the model for transmission model is *relay-efficient*. An example of a relay-efficient model is the aforementioned power-law model. We define the *relay region* to be the region between two sensors where a third sensor can be placed to reduce the overall energy cost required for transmission, and we say that a model is relay-efficient if the relay region always exists for sufficiently large distances. Note that this is a critical property to check when arguing that a cutset network is energy-efficient, since the energy-efficiency claim of a cutset network relies on the assumption that relaying a packet through neighboring sensors is more efficient than direct transmission. To determine relay-efficiency, we prove sufficient conditions that can be checked, and also give geometric bounds on the size and shape of the relay region. We also calculate the "best" separation distance for two communicating sensors, which is the distance that minimizes the energy-per-distance function.

Finally, in Section 4.6, under the assumption that our model is relay-efficient, we characterize the cost of paths of minimal energy through a lattice sensor network. Good approximations to these paths can be obtained by solving a linear program, and we prove that the solution to this linear program converges to the cost of the actual best path at long distances. However, our simulations show that this approximation can work well even for short paths. We also conjecture that for long-distance communication, the repetition of at most two "hop types" are needed to describe an energy-efficient path. We then propose a method for predicting these paths and the cost of these paths in closed form. We conclude the chapter with numerical simulation which support this conjecture.

We note that most of the notation in this thesis is relatively self-contained to each chapter, and in some cases, each section.

CHAPTER 2

Multidimensional Manhattan Sampling

This chapter introduces Manhattan sampling in two and higher dimensions, and proves sampling theorems. In two dimensions, Manhattan sampling, which takes samples densely along a Manhattan grid of lines, can be viewed as sampling on the union of two rectangular lattices, one dense horizontally, the other vertically, with the coarse spacing of each being a multiple of the fine spacing of the other. The sampling theorem shows that images bandlimited to the union of the Nyquist regions of the two rectangular lattices can be recovered from their Manhattan samples, and an efficient procedure for doing so is given. Such recovery is possible even though there is overlap among the spectral replicas induced by Manhattan sampling.

In three and higher dimensions, there are many possible configurations for Manhattan sampling, each consisting of the union of special rectangular lattices called bi-step lattices. This chapter identifies them, proves a sampling theorem showing that images bandlimited to the union of the Nyquist regions of the bi-step rectangular lattices are recoverable from Manhattan samples, and presents an efficient onion-peeling procedure for doing so. Furthermore, it develops a special representation for the bi-step lattices and an algebra with nice properties. It is also shown that the set of reconstructable images is maximal in the Landau sense.

While most of the chapter deals with continuous-space images, Manhattan sampling of discrete-space images is also considered, for infinite, as well as finite, support images.

Finally, we note that the 2D work of Section 2.3 was originally presented at ICASSP 2012 [12], and the extensions to higher dimensions have been submitted for review to IEEE Transactions in Information Theory.

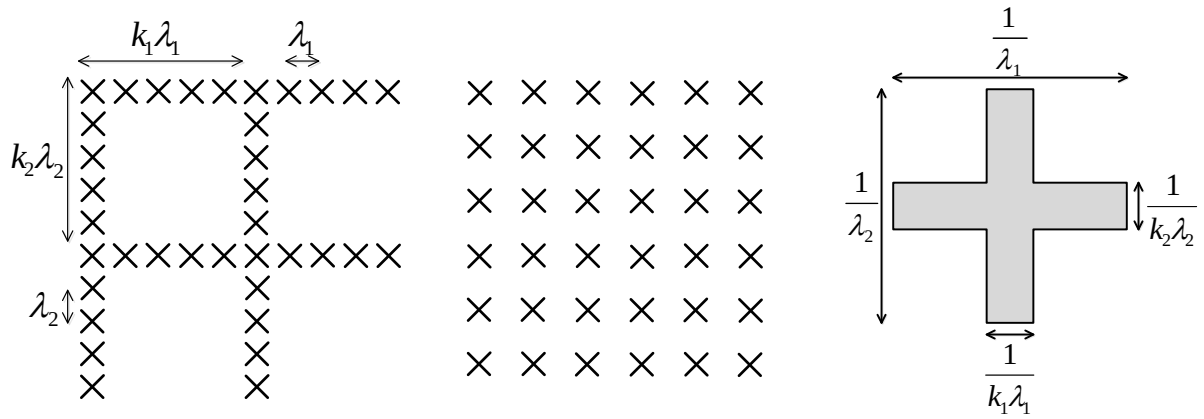


Figure 2.1: (a) 2D Manhattan-grid sampling sites with parameters $k_1 = k_2 = 5$ and $\lambda_1 = \lambda_2$. (b) Square lattice sampling at the same density. (c) Cross-shaped frequency support (centered at the origin) of images recoverable with Manhattan sampling.

2.1 Introduction

In the two-dimensional (2D) setting, *Manhattan sampling* (or *M-sampling* for short) is a recently proposed form of image sampling in which data is taken along evenly spaced rows and columns; the set of sample locations will be called a *Manhattan grid*. In particular, as illustrated in Fig. 2.1(a), given sampling intervals $\lambda_1, \lambda_2 > 0$ and integers $k_1, k_2 > 1$, samples are taken at intervals of λ_1 along horizontal rows spaced $k_2\lambda_2$ apart, and also at intervals of λ_2 along vertical columns spaced $k_1\lambda_1$ apart.

Manhattan sampling has been used to good effect in both lossy and lossless bilevel image compression [4–6]. These methods losslessly compress the samples in a Manhattan grid, for example with arithmetic coding (AC), which can be done with very few bits per M-sample because the samples are closely spaced and, hence, highly correlated. For lossless compression, the other pixels are then AC encoded, conditioned on those in the Manhattan grid, while for lossy compression there is no further encoding, and the decoder estimates the remaining pixels from those in the Manhattan grid. Markov random field models have been used to guide both the arithmetic coding and the estimation.

M-sampling has also been proposed [1, 13–15] as a new approach to sampling grayscale images and other two-dimensional fields, with the motivations that (a) dense sampling along lines might capture edge transitions more completely than conventional lattice sampling with the same density, (b) sensor networks with a Manhattan deployment geometry need less power or less wire to transmit data than conventional lattice or random deployments at the same density [14, 15], and (c) there are physical scenarios for which M-sampling is far

more natural than traditional lattice sampling, such as when sampling from a moving vehicle, e.g., a ship sampling oxygen levels in a body of water. Similarly motivated by sampling from vehicles, the recent related work of Unnikrishnan and Vetterli [16, 17] considers sampling continuously along a grid of lines, i.e., with asymptotically large sampling rate.

Methods for approximately reconstructing typical (non-bandlimited) images from M-samples have been developed in [1, 13, 18]. The present chapter focuses on identifying a bandlimited set of images that can be perfectly reconstructed, as well as efficient methods for doing so.

Manhattan sampling with parameters $\lambda_1, \lambda_2, k_1, k_2$ can be viewed as sampling on the union of the horizontally dense rectangular lattice consisting of all locations of the form $(n_1\lambda_1, n_2k_2\lambda_2)$, where n_1, n_2 are arbitrary integers, and the similarly defined vertically dense rectangular lattice consisting of all locations of the form $(n_1k_1\lambda_1, n_2\lambda_2)$. For brevity, we call these the *horizontal* and *vertical* lattices, respectively.

By the conventional 2D sampling theorem [19] (see also [20, p. 72], [21, Chap. 3], [22, p. 43]), the samples on the horizontal lattice are sufficient to distinguish and reconstruct any image bandlimited to the Nyquist region $\left\{(u,v) : |u| < \frac{1}{2\lambda_1}, |v| < \frac{1}{2k_2\lambda_2}\right\}$. Likewise the samples on the vertical lattice are sufficient to distinguish and reconstruct any image bandlimited to the Nyquist region $\left\{(u,v) : |u| < \frac{1}{2k_1\lambda_1}, |v| < \frac{1}{2\lambda_2}\right\}$. Each of these samplings is maximally efficient in the Landau sense [8] that their sampling densities are as small as the area of the Nyquist region. Equivalently, the set of images bandlimited to the Nyquist region is maximal for the given sampling scheme.

The first result of the present chapter is a sampling theorem in Section 2.3 showing that images bandlimited to the union of these two Nyquist regions can be reconstructed from their samples on the union of the two rectangular lattices, i.e., on the Manhattan grid, and an efficient procedure for doing so is given. It is also shown that the images bandlimited in this way form a maximal reconstructable set for the Manhattan grid samples. As illustrated in Fig. 2.1(c), the union of the two Nyquist regions is the cross-shaped *Manhattan region*. We say that images whose spectra are confined to such a region are *Manhattan-bandlimited*. Given the relevance of Manhattan-bandlimiting, a figure in Section 2.3 will display the effect of several instances of such on a typical image.

The principal goals of the remainder of the chapter are to formulate M-sampling in three and higher dimensions, and to derive a sampling theorem and a reconstruction procedure. M-sampling in three dimensions can be motivated by the need to spatially sample a three-dimensional volume with a vehicle, or to spatio-temporally sample a two-dimensional region, as in video, or a spatio-temporal sensor network. Four-dimensional sampling can be motivated by the need for spatio-temporal sampling of a three-dimensional spatial region.

In three and higher dimensions, M-sampling can take a variety of forms. In order to describe two of these in three dimensions, consider the partition of 3D space into $k_1\lambda_1 \times k_2\lambda_2 \times k_3\lambda_3$ orthotopes (3D rectangles). As illustrated in Fig. 2.4(a), one form of M-sampling takes samples uniformly along each edge of each of these orthotopes — with spacing λ_i along edges parallel to axis i. Another form (Fig. 2.4(c)) takes samples uniformly on each face of each orthotope — with the samples on the face orthogonal to axis i taken according to a $\lambda_j \times \lambda_k$ rectangular lattice, where j and k denote the other dimensions. In other words, the first form samples densely along lines and the second samples densely along hyperplanes. Neither of these takes samples in the interior of any of the aforementioned orthotopes.

More generally, as described in Section 2.4, M-sampling in an arbitrary dimension d is defined as taking samples on the union of some collection of d-dimensional *bi-step* lattices, which are rectangular lattices defined by step sizes that in dimension i are restricted to λ_i or $k_i\lambda_i$. Thus, there are many possible M-samplings in d dimensions, even when λ_i's and k_i's are fixed. We call such unions of d-dimensional bi-step lattices *Manhattan sets*.

The main results of Sec. 2.4 are (a) a sampling theorem showing that images bandlimited to the union of the Nyquist regions of the d-dimensional bi-step lattices comprising the Manhattan set can be distinguished by their M-samples, (b) efficient, onion-peeling procedures for perfectly reconstructing d-dimensional images, bandlimited as in (a), from their M-samples (one in frequency domain and one in spatial domain), and (c) a proof that the set of such bandlimited images is maximal in the Landau sense.

The development of the sampling theorem and reconstruction procedures are enabled by an efficient parameterization of a bi-step lattice (with a given set of λ_i's and k_i's) by a binary vector $\mathbf{b} = (b_1, \ldots, b_d)$ indicating the dimensions i along which the spacing between lattice points is the smaller value, λ_i, rather than the larger value, $k_i\lambda_i$. This enables any Manhattan set to be compactly described by a finite set of $\mathbf{b}_i$'s (in addition to the λ_i's and k_i's). A number of properties and relationships are enabled by this parameterization. For example, the computation of the density of a d-dimensional Manhattan set is enabled by a spatial partition whose 2^d atoms are indexed by $\mathbf{b}$'s. Similarly, the onion-peeling reconstruction procedures mentioned previously are keyed to a partition of frequency space whose 2^d atoms are indexed by $\mathbf{b}$'s. The frequency-domain version reconstructs the image spectrum one atom at a time, beginning with "highest frequency" atoms (whose $\mathbf{b}$'s contain the most 1's), and working towards the lower frequency atoms (whose $\mathbf{b}$'s contain fewer ones).

In particular, as will be shown, the spectrum $X^{\mathbf{b}}(\mathbf{u})$ in the atom indexed by $\mathbf{b}$ is computed via

$$X^{\mathbf{b}}(\mathbf{u}) \;=\; X_{\mathbf{b}}(\mathbf{u}) \;-\; \sum_{\mathbf{b}':\, \|\mathbf{b}'\| > \|\mathbf{b}\|} X_{\mathbf{b}}^{\mathbf{b}'}(\mathbf{u})\,, \tag{2.1}$$

where $X_{\mathbf{b}}(\mathbf{u})$ is the spectrum of the image samples in the bi-step lattice parameterized by $\mathbf{b}$ (a subset of the Manhattan set), the sum is over all $\mathbf{b}'$ with more ones than $\mathbf{b}$, and $X_{\mathbf{b}}^{\mathbf{b}'}(\mathbf{u})$ is the spectrum of the samples (taken with the same bi-step lattice) of the image component $x^{\mathbf{b}'}(\mathbf{t})$ corresponding to atom $\mathbf{b}'$, which has previously been reconstructed.

A discrete-space version of this requires only DFTs of the subsampling of the Manhattan samples and the previously reconstructed image components specified in the above, as well as summing and subtracting. Then an inverse DFT computes the newly reconstructed component. Summing all such components yields the reconstructed image.

The method characterized by (2.1), and the discrete-space version thereof, can also be carried out in the spatial domain by applying the right-hand side of (2.1) to the corresponding sampled images, rather than their spectra, and then applying an ideal bandpass filter that extracts just the frequency component corresponding to atom $\mathbf{b}$. The impulse responses of these filters will be given later. As will be seen, these impulse responses depend on the k_i's and λ_i's, but not the choice of bi-step lattices that comprise the Manhattan set. Moreover, the λ_i's have only a simple spatial scaling effect on the filters.

Finally, we note that the development for three dimensions benefits greatly from the efficient parameterization of bi-step lattices mentioned earlier, and that with such, it is possible to derive the M-sampling theorem and reconstruction procedure in arbitrary dimensions with essentially no additional effort or notation.

We conclude this introduction by relating the present work to previous work. Multidimensional sampling theorems, showing that images with certain spectral support regions can be reconstructed from certain samplings sets, appeared first for lattice sampling sets in Peterson and Middleton [19], then later for unions of shifted lattices, i.e., lattice cosets, [23–34], although they were not always described as such.

The earliest work [19, 23] required the spectral support region and sampling set to be chosen so that the spectral replicas induced by sampling did not overlap, and consequently, reconstruction could proceed simply by lowpass filtering the sampled image. For example, the approach of [23] could be used to reconstruct images from M-samples. However, it would require the images to be bandlimited to the Nyquist region of the *coarse (rectangular) lattice*, which is the intersection (rather than union) of the bi-step lattices comprising the Manhattan set.

Non-overlapping spectral replicas were not required in later work [24–34], and more complex reconstruction procedures were proposed. Though not specifically intended for images, a seminal contribution stimulating a number of advances in image sampling was the multichannel, generalized sampling introduced by Papoulis [35]. For example, Papoulis' framework is broad enough to include all image sampling schemes based on lattices and unions of

shifted lattices.

One difference between the present work and much past work is that we focus on a particular sampling set, namely a Manhattan set, and seek a largest possible frequency region such that any image bandlimited to such can be reconstructed from the samples. In contrast, much of the past work [24–26, 30, 31, 36] focused on a particular frequency support region and sought a smallest possible sampling set, constructed from lattices and shifts thereof, such that images bandlimited to this region could be reconstructed from such sampling sets. Nevertheless, some of the latter approaches could be used to reverse engineer reconstruction procedures and/or spectral support regions for Manhattan sets, as we now discuss.

One substantial line of past work applies to sampling sets that consist of a sublattice of some specified *base lattice*, together with some of its cosets, each of which is a shift of the sublattice by some base lattice point. In this case, the subsampling corresponding to each coset (including the sublattice itself) can be viewed as a *channel* in a Papoulis multichannel, generalized sampling scheme. Consequently, the method of [35] can be applied. This is the approach taken by Marks and Cheung [24–26]. Since a Manhattan set can be viewed as the union of what we earlier called the coarse (rectangular) lattice and some number of its cosets with respect to the *dense (rectangular) lattice*, which contains all points $\mathbf{t}$ such that for each i, its ith coordinate is an integer multiple of λ_i, the Papoulis-Marks-Cheung (PMC) approach can be applied to Manhattan sets.

In particular, Marks and Cheung focused on images with a given spectral support region and an initial base sampling lattice such that the induced spectral replicas of this support region do not overlap. They then showed that cosets of some sublattice could be removed from the base lattice until the sampling density was minimal (in the Landau sense) or approached minimal. Their method involved (a) partitioning the Nyquist region of the initial base lattice into atoms the size and shape of the Nyquist region of the sublattice, (b) counting the number of atoms of this partition that are not overlapped by any spectral replica of the designated support region induced by the initial base sampling lattice, and (c) showing that this number of sublattice cosets can be removed from the initial base lattice due to their samples being linearly dependent on other samples. If the atoms of the partition are too coarse to closely match the set of frequencies not contained in any spectral support replica, then choosing a sparser sublattice will enable a finer partitioning, resulting in a higher fraction of the base samples being removed, which allows the sampling rate to be reduced until it equals or approaches the Landau minimum.

With hindsight, one can apply their approach to a Manhattan sampling set. For simplicity, consider a 2D case and assume $k_1 = k_2 = k$. Suppose images are bandlimited to the cross-shaped Manhattan region, and let the initial base sampling lattice and the sub-

lattice be the dense and coarse rectangular lattices mentioned earlier. In this case, there are k^2 cosets of the sublattice (including itself). One can then see that in the partition of the Nyquist region of the base/dense lattice into atoms having the size and shape of the Nyquist region of the coarse lattice, the number of atoms that are not contained in any sampled spectra is $k^2 - (2k-1)$. Thus, it is possible to remove all but $2k-1$ cosets, which is precisely the number of Manhattan samples in one $k \times k$ fundamental cell of the coarse lattice. Unfortunately, the PMC approach does not determine which cosets can be removed, so it does not directly tell us if the Manhattan samples are sufficient to recover an image. While it does provide a matrix invertibility test that one can apply in any particular case to see if the Manhattan samples are sufficient, it is not clear how to analytically establish that one can remove all but the Manhattan samples in all cases. It is also not clear how the PMC approach would have led to the discovery that the union of the Nyquist regions of the bi-step lattices is a reconstructable spectra support region for Manhattan sampling, especially in dimensions three and above. However, once it is known that the Manhattan samples are sufficient for the spectral support region found in the present chapter, then the Papoulis approach will directly lead to a reconstruction algorithm.

As both the PMC and onion-peeling approaches involve partitioning frequency space, it is interesting to note that in dimension d the PMC approach requires a partition into $\prod_{i=1}^{d} k_i$ atoms, whereas the onion-peeling algorithm partitions into only 2^d atoms. The smaller size of the latter partition is due to its being closely tailored to the specific structure of Manhattan samples.

Similarly, in another line of work, Faridani [27] derived a sampling theorem and reconstruction formula for unions of shifts of one lattice. Given a spectral support region, the reconstruction involves partitioning this region in a certain way and setting up and solving a sizable number of systems of linear equations, assuming that the equations have a solution. Since a Manhattan set can be viewed as the union of shifts of a lattice (the coarse lattice) and since we know from the results of the present chapter that it is possible to reconstruct M-sampled images bandlimited to the Manhattan region, one could presumably solve the resulting equations to obtain a reconstruction formula. While this is interesting, finding the partition and setting up the equations can be difficult, especially in high dimensions. Thus, as before, the onion-peeling approach proposed in this chapter is more natural, intuitive and straightforward to implement.

While the PMC and Faridani approaches could be used to derive a reconstruction method for any Manhattan set, in their basic form, they do not provide direct closed form reconstruction methods, as given for example in this chapter. That is, given sets of k_i's and bi-step lattices, they outline a procedure that could be followed in order to derive a reconstruction

method. Then, when the k_i's or bi-step lattices are changed, the procedure must be followed again, essentially from scratch[1]. In contrast, the reconstruction methods given in the chapter are closed form, requiring just step-by-step following of the reconstruction formulas, which depend explicitly on the λ_i's, k_i's and bi-step lattices. While it is conceivable that with enough work this alternative approach could be made closed form, it would appear to take much additional work, especially to make it apply to arbitrary dimensions.

Behmard [33, 37] derived a sampling theorem and reconstruction formula for unions of shifts of more than one lattice, which includes M-sampling, as it is a more general setting than [27]. However, the *compatibility conditions* required to apply this approach are not satisfied by M-sampling and the Manhattan spectral support region.

Other work on sampling with unions of shifted lattices includes that of (a) Venkataramani and Bresler [30, 31], which considered unions of shifted lattices in one dimension, and (b) Unnikrishnan and Vetterli [34], which considered unions of shifted lattices in higher dimensions. The latter include M-sampling and a reconstruction procedure was proposed with similarities to our onion-peeling approach, but which requires the spectral support region to be convex, which rules out the Manhattan region. Indeed, one of their examples is a 2D Manhattan grid, from which images can be recovered provided their spectra are bandlimited to a circular subset of the Manhattan region. Consequently, a significantly smaller set of images is reconstructable with their procedure.

Finally, we mention that Manhattan-bandlimited spectra have been found to arise naturally in dynamic medical imaging applications, including both time-varying tomography [36] and dynamic MRI [38]. For example, Rilling et. al. [38, Fig. 1] give carotid blood velocity mapping as an example of a dynamic MRI application where a cross-shaped spectrum appears. Moreover, such spectra arise when temporal variation is localized to a small spatial area relative to the rest of the body, such as beating heart. With this motivation, Willis and Bresler [36] derived a single sampling lattice such that the cross-shaped spectral replicas did not overlap and the sampling rate was close to the Landau lower bound. In contrast, our sampling theorem also shows perfect reconstruction is possible. However, we sample with more than one lattice, the spectral replicas overlap, and the Landau bound is met exactly.

In summary, given that the present chapter shows that images bandlimited to the union of the Nyquist regions of the bi-step lattices of a Manhattan sampling set can be perfectly reconstructed from the Manhattan samples, there are probably a number of alternative ways to derive reconstruction algorithms. In the view of the authors, the onion-peeling method, whose development was guided by the specific structure of Manhattan samples, is a natural and efficient reconstruction method with a straightforward interpretation in frequency space.

[1]The method can be derived assuming unit λ_i's and then spatially scaled for the actual λ_i's.

It is also closed form in terms of the parameters of the Manhattan set. In addition, the union-of-bi-step-lattice viewpoint taken in this chapter leads naturally to the hypothesis that the union of Nyquist regions is a support region of images that are reconstructable from Manhattan samples. It is not known if other approaches would have lead investigators to this region.

The chapter is written so that the reader who is primarily interested in 2D images can focus on Sections 2.2, 2.3, and 2.5.

2.2 Preliminaries

This section provides background and notation for sampling and lattices that will be used throughout the chapter.

Let $\mathbb{R}$ denote the real numbers, let $\mathbb{R}^d$ denote d-dimensional Euclidean space, let $\mathbb{Z}$ denote the set of all integers, and let $\mathbb{Z}^d$ denote the set of all integer-valued d-dimensional vectors. In dimension d, an image is a mapping $x(\mathbf{t}) : \mathbb{R}^d \to \mathbb{R}$, where the spatial variable is $\mathbf{t} = (t_1, \ldots, t_d)$. We restrict attention to images $x(\mathbf{t})$ that contain no delta functions or other generalized functions, and have well defined Fourier transforms containing no delta functions or other generalized functions, where by Fourier transform we mean

$$X(\mathbf{u}) \;=\; \mathcal{F}\big\{x\big(\mathbf{t}\big)\big\} \;\triangleq\; \int x(\mathbf{t})\, e^{-j2\pi \mathbf{t}\cdot\mathbf{u}}\, d\mathbf{t}\,.$$

We will often refer to $X(\mathbf{u})$ as the *spectrum* of $x(\mathbf{t})$.

Sampling a d-dimensional image $x(\mathbf{t})$ means collecting its values on some countable *sampling set* $\mathbb{S}$. That is, it produces the set of values $\{x(\mathbf{t}) : \mathbf{t} \in \mathbb{S}\}$. As commonly done, one can model such sampling as multiplication of $x(\mathbf{t})$ by the *comb function* of the set $\mathbb{S}$, which produces the *sampled image*

$$x_{\mathbb{S}}(\mathbf{t}) \;\triangleq\; x(\mathbf{t})\, K_{\mathbb{S}} \sum_{\mathbf{t}' \in \mathbb{S}} \delta(\mathbf{t} - \mathbf{t}')\,, \tag{2.2}$$

where $K_{\mathbb{S}}$ is a normalizing constant and $\delta(\mathbf{t})$ denotes the Dirac delta function in d-space. The Fourier transform of $x_{\mathbb{S}}(\mathbf{t})$ is then called the *sampled spectrum*.

Rectangular sampling refers to sampling with a rectangular lattice. Given d and $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_d)$ with positive components, the d-dimensional *rectangular lattice* with *step vector* $\boldsymbol{\alpha}$ is a countably infinite set of points that are spaced by integer multiples of the *step size*

α_i in the ith dimension. Specifically,

$$\begin{aligned} L(\boldsymbol{\alpha}) &\triangleq \left\{\mathbf{t} : t_i \text{ is a multiple of } \alpha_i, i = 1, \ldots, d\right\} \\ &= \left\{\mathbf{t} = \mathbf{n} \odot \boldsymbol{\alpha} : \mathbf{n} \in \mathbb{Z}^d\right\}, \end{aligned}$$

where $\odot$ denotes element-wise product (also known as the Hadamard or Schur product). Alternatively, $L(\boldsymbol{\alpha})$ is the additive group generated by the basis $\alpha_1\mathbf{e}_1, \ldots, \alpha_d\mathbf{e}_d$, where $\mathbf{e}_1, \ldots, \mathbf{e}_d$ is the standard basis, i.e., $\mathbf{e}_i$ has a 1 in the ith place and 0's elsewhere. That is,

$$L(\boldsymbol{\alpha}) \triangleq \left\{\mathbf{t} = \sum_{i=1}^{d} n_i \alpha_i \mathbf{e}_i : \mathbf{n} \in \mathbb{Z}^d\right\}.$$

The *reciprocal lattice* corresponding to $L(\boldsymbol{\alpha})$ is

$$L^*(\boldsymbol{\alpha}) \triangleq L(\alpha_1^{-1}, \ldots, \alpha_d^{-1}).$$

When sampling with set $\mathbb{S} = L(\boldsymbol{\alpha})$, it is convenient to set the normalizing constant in (2.2) to be

$$K_{\mathbb{S}} = \prod_{i=1}^{d} \alpha_i. \tag{2.3}$$

With this, the sampled image, denoted $x_{\boldsymbol{\alpha}}(\mathbf{t})$, has spectrum

$$X_{\boldsymbol{\alpha}}(\mathbf{u}) = \sum_{\mathbf{v} \in L^*(\boldsymbol{\alpha})} X(\mathbf{u} - \mathbf{v}). \tag{2.4}$$

From this, one sees that the sampled spectrum $X_{\boldsymbol{\alpha}}(\mathbf{u})$ consists of replicas of the original image spectrum $X(\mathbf{u})$, translated to the sites in frequency domain of the reciprocal lattice. The usual d-dimensional sampling theorem follows from the fact that if the support of $X(\mathbf{u})$ lies entirely within the Nyquist region[2]

$$\mathcal{N}_{\boldsymbol{\alpha}} \triangleq \left\{\mathbf{u} : |u_i| < \frac{1}{2\alpha_i},\ i = 1, \ldots, d\right\},$$

then said replicas do not overlap, and consequently, the original spectrum can be recovered by extracting the portion of the sampled image spectrum in the Nyquist region.

[2]In this chapter, script variables such as $\mathcal{N}, \mathcal{B}$ or $\mathcal{A}$ will usually denote subsets of frequency space.

2.3 Two-Dimensional Manhattan Sampling

As introduced earlier and depicted in Fig. 2.1(a), Manhattan sampling (M-sampling) uses locations spaced closely along a grid of horizontal and vertical lines. In particular, we assume there is a sample at the origin, as well as samples spaced λ_1 apart on horizontal lines spaced $k_2\lambda_2$ apart, and samples spaced λ_2 apart on vertical lines spaced $k_1\lambda_1$ apart, where $\lambda_i > 0$ and k_1, k_2 are integers greater than one[3]. The issue, now, is to find an as large as possible set of images that can be perfectly reconstructed from these samples, as well as an efficient procedure for doing so.

A first thought is to model M-sampling as multiplying the given image $x(\mathbf{t})$ by a comb function having delta functions at the Manhattan sampling locations, and then to analyze the spectra of the resulting sampled image. Since this comb function has the same periodicity as a comb function for the *coarse lattice* $L_C \triangleq L(k_1\lambda_1, k_2\lambda_2)$, the replicas of the image spectrum lie at frequency sites in the reciprocal lattice L_C^*, or a subset thereof. Thus, perfect reconstruction is possible for images bandlimited to the Nyquist region $\mathcal{N}_C$ of the coarse lattice L_C. However, since such reconstructions need only use samples in the coarse lattice, it may be that a larger set of images is reconstructable from the full Manhattan grid. On the other hand, if images are bandlimited to a region larger than $\mathcal{N}_C$, e.g., a scaling of the Nyquist region such as $(1+\epsilon)\mathcal{N}_C$, $\epsilon > 0$, then the spectral replicas induced by an M-sampling comb may overlap. Even if this does not eliminate the possibility of perfect reconstruction, it will at least complicate the analysis.

Accordingly, we pursue an approach that does not rely on nonoverlapping replicas, but derives from the key observation that the Manhattan sampling set can be viewed as the *union* of two rectangular lattices. Let us initially focus on what can be recovered from the samples of each lattice by itself. The *horizontal lattice*, $L_H \triangleq L(\lambda_1, k_2\lambda_2)$, densely samples in the horizontal direction and coarsely samples in the vertical direction; the *vertical lattice*, $L_V \triangleq L(k_1\lambda_1, \lambda_2)$, coarsely samples in the horizontal direction and densely samples in the vertical direction; and the sampling set for M-sampling is

$$M(\boldsymbol{\lambda}; \mathbf{k}) \;=\; L_H \cup L_V \,.$$

Note also that the intersection of the two lattices is the coarse lattice L_C, whose comb function has the same periodicity as a comb function for the Manhattan grid.

Clearly, all images bandlimited to the Nyquist region $\mathcal{N}_H$ of the horizontal lattice L_H can be recovered from just the samples in this lattice. Likewise, all images bandlimited to the Nyquist region $\mathcal{N}_V$ of the vertical lattice L_V can be recovered from just the samples in

[3]We require $k_1, k_2 > 1$ since if $k_1 = 1$ or $k_2 = 1$, the sampling set reduces to a normal rectangular lattice.

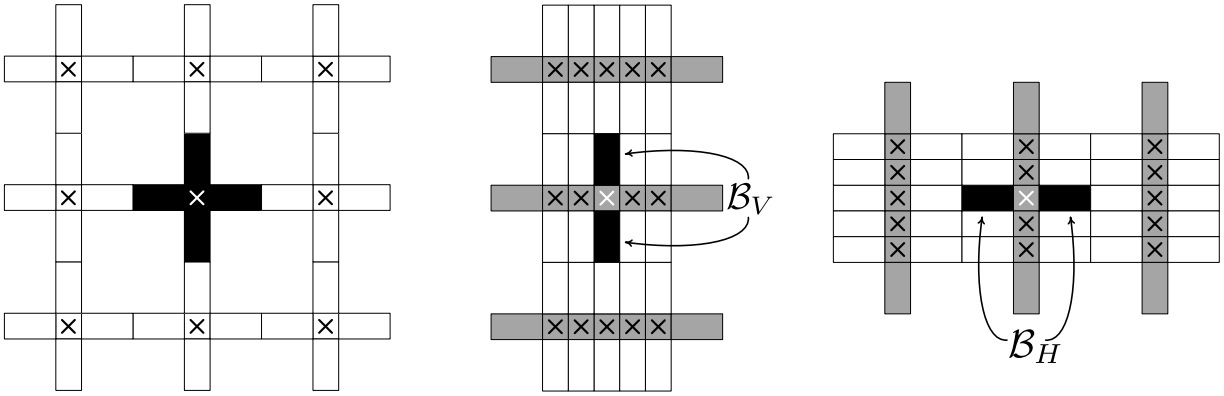


Figure 2.2: For Manhattan sampling with $\lambda_1 = \lambda_2$ and $k_1 = k_2 = 3$: (a) Support of the sampled spectrum for an image bandlimited to the Manhattan region, when sampled with the fine lattice L_{λ_1,λ_2}. The original spectrum is black with a white $\times$ in its center, whereas replicas are white with a black $\times$ in their centers. (b) Support of the sampled spectrum when sampled with the vertical lattice $L_{k_1\lambda_1,\lambda_2}$. Gray indicates regions where replicas overlap either the original spectrum or each other. (c) Same as (b), except that the sampling is with the horizontal lattice $L_{\lambda_1,k_2\lambda_2}$.

this lattice. Each of these by itself leads to a larger recoverable set of images than the set recoverable from sampling with the coarse lattice L_C. However, neither type of sampling and reconstruction uses all of the M-samples.

We now show how images bandlimited to the union of the Nyquist regions of the horizontal and vertical lattices can be recovered from the full set of M-samples. Specifically, suppose image $x(\mathbf{t})$ is bandlimited to $\mathcal{M}(\boldsymbol{\lambda};\mathbf{k}) = \mathcal{N}_H \cup \mathcal{N}_V$, which is the cross-shaped region shown in Fig. 2.1(c). First, consider only the samples of $x(\mathbf{t})$ taken on the vertical lattice. Since the cross-shaped region $\mathcal{M}(\boldsymbol{\lambda};\mathbf{k})$ is not contained in the Nyquist region $\mathcal{N}_V$, the replicas of $X(\mathbf{u})$ may overlap in the spectrum of the vertically sampled image, as illustrated in Fig. 2.2(b), However, certain portions of each cross-shaped replica cannot be overlapped, and thus these portions of the spectra of $x(\mathbf{t})$ can be immediately recovered.

Specifically, it is easy to see that with vertical sampling, the vertical highpass region $\mathcal{B}_V \triangleq \mathcal{N}_V - \mathcal{N}_C$ is not overlapped. Thus, with $I_D(\mathbf{u})$ denoting the indicator function of some set D and $X^V(\mathbf{u}) \triangleq X(\mathbf{u})I_{\mathcal{B}_V}(\mathbf{u})$ denoting the portion of $X(\mathbf{u})$ in $\mathcal{B}_V$, one sees that from the vertical samples and their spectrum $X_V(\mathbf{u})$, one can immediately recover $X^V(\mathbf{u})$ via $X^V(\mathbf{u}) = X_V(\mathbf{u})I_{\mathcal{B}_V}(\mathbf{u})$. Likewise from the horizontal samples and their spectrum $X_H(\mathbf{u})$, the horizontal highpass region $\mathcal{B}_H \triangleq \mathcal{N}_H - \mathcal{N}_C$ is not overlapped. Thus, one can immediately recover $X^H(\mathbf{u}) \triangleq X(\mathbf{u})I_{\mathcal{B}_H}(\mathbf{u}) = X_H(\mathbf{u})I_{\mathcal{B}_H}(\mathbf{u})$.[4]

[4]Throughout the chapter, a superscript on an image x or spectrum X will usually pertain to a frequency region, and a subscript will usually pertain to a sampling.

Since $X^V(\mathbf{u})$ and $X^H(\mathbf{u})$ are now known, and $X(\mathbf{u})$ is bandlimited to $\mathcal{M}(\boldsymbol{\lambda}, \mathbf{k}) = \mathcal{B}_H \cup \mathcal{B}_V \cup \mathcal{N}_C$, it remains only to find $X^C(\mathbf{u}) \triangleq X(\mathbf{u}) I_{\mathcal{N}_C}(\mathbf{u})$. It will then follow that $X(\mathbf{u}) = X^V(\mathbf{u}) + X^H(\mathbf{u}) + X^C(\mathbf{u})$. Inverse transforms will give $x(\mathbf{t}) = x^V(\mathbf{t}) + x^H(\mathbf{t}) + x^C(\mathbf{t})$.

To determine $X^C(\mathbf{u})$, consider the vertical sampling of $x(\mathbf{t})$, and observe in Fig. 2.2(b) that the overlap of the image spectrum $X(\mathbf{u})$ in $\mathcal{N}_C$ by the various spectral replicas is due only to replications of the horizontal highpass frequency components in $\mathcal{B}_H$. Since these have already been determined, it ought to be possible subtract their effects.

To see that this can be done, let us focus on $X_V(\mathbf{u})\, I_{\mathcal{N}_C}(\mathbf{u})$. From (2.4) and the fact that $X(\mathbf{u}) = 0$ for $\mathbf{u} \notin \mathcal{M}(\boldsymbol{\lambda}, \mathbf{k})$, we have

$$X_V(\mathbf{u})\, I_{\mathcal{N}_C}(\mathbf{u}) \;= \sum_{i=-n}^{n} X\Big(u_1 - \frac{i}{k_1\lambda_1}, u_2\Big)\, I_{\mathcal{N}_C}(\mathbf{u})\,,$$

where $n = \big\lfloor \frac{k_1}{2} \big\rfloor$. Now using $X(\mathbf{u}) = X^V(\mathbf{u}) + X^H(\mathbf{u}) + X^C(\mathbf{u})$ in the above along with the facts that

(a) $X^V\big(u_1 - \frac{i}{k_1\lambda_1}, u_2\big)\; I_{\mathcal{N}_C}(\mathbf{u}) = 0$ for all i,

(b) $X^H\big(u_1 - \frac{i}{k_1\lambda_1}, u_2\big)\, I_{\mathcal{N}_C}(\mathbf{u}) = 0$ for $i = 0$,

(c) $X^C\big(u_1 - \frac{i}{k_1\lambda_1}, u_2\big)\, I_{\mathcal{N}_C}(\mathbf{u}) = 0$ unless $i = 0$,

we find

$$X_V(\mathbf{u})\, I_{\mathcal{N}_C}(\mathbf{u}) \;=\; X^C(\mathbf{u}) \;+\; Y(\mathbf{u})\, I_{\mathcal{N}_C}(\mathbf{u})\,, \tag{2.5}$$

where

$$\begin{aligned} Y(\mathbf{u}) \;&\triangleq \sum_{\substack{i=-n \\ i\neq 0}}^{n} X^H\Big(u_1 - \frac{i}{k_1\lambda_1}, u_2\Big) \\ &= \sum_{\substack{i=-n \\ i\neq 0}}^{n} X_H\Big(u_1 - \frac{i}{k_1\lambda_1}, u_2\Big) I_{\mathcal{B}_H}\Big(u_1 - \frac{i}{k_1\lambda_1}, u_2\Big). \end{aligned} \tag{2.6}$$

where the last equality uses the fact, mentioned earlier, that $X^H(\mathbf{u}) = X_H(\mathbf{u}) I_{\mathcal{B}_H}(\mathbf{u})$. Notice that $Y(\mathbf{u})$ is the component of $X_V(\mathbf{u})$ due to aliasing by replicas of $X^H(\mathbf{u})$, and is directly computable from the horizontal samples. It follows from (2.5) that $X^C(\mathbf{u}) = (X_V(\mathbf{u}) - Y(\mathbf{u}))\, I_{\mathcal{N}_C}(\mathbf{u})$.

In summary, a procedure for recovering a cross bandlimited x from its M-samples is

1. Compute the spectra, $X_H(\mathbf{u})$ and $X_V(\mathbf{u})$, of the horizontally and vertically dense

samples, respectively.

2. From $X_H(\mathbf{u})$, compute $Y(\mathbf{u})$ for $\mathbf{u} \in \mathcal{N}_C$.

3. Let

$$\widehat{X}(\mathbf{u}) = \begin{cases} X_V(\mathbf{u}), & \mathbf{u} \in \mathcal{B}_V \\ X_H(\mathbf{u}), & \mathbf{u} \in \mathcal{B}_H \\ X_V(\mathbf{u}) - Y(\mathbf{u}), & \mathbf{u} \in \mathcal{N}_C \end{cases} . \tag{2.7}$$

4. Let $\widehat{x}(\mathbf{t})$ be the inverse Fourier transform of $\widehat{X}(\mathbf{u})$.

This result is summarized in the following.

Theorem 1. 2D Manhattan sampling theorem. *Given $\lambda_1, \lambda_2 > 0$ and integers k_1, k_2 greater than 1, any image $x(\mathbf{t})$ whose Fourier transform is bandlimited to the cross-shaped region $\mathcal{M}(\boldsymbol{\lambda}; \mathbf{k})$ can be recovered from its M-samples in $M(\boldsymbol{\lambda}; \mathbf{k})$ with the procedure given above.*

The following alternative expression for $X_V(\mathbf{u})\, I_{N_C}(\mathbf{u})$ will lead to an easier to implement procedure for discrete-space images with finite support (presented later). Using (2.4) and $X(\mathbf{u}) = X^V(\mathbf{u}) + X^H(\mathbf{u}) + X^C(\mathbf{u})$, we find

$$\begin{aligned} X_V(\mathbf{u})\, I_{\mathcal{N}_C}(\mathbf{u}) &= \sum_{\mathbf{v} \in L_V^*} \Big(X^V(\mathbf{u}-\mathbf{v}) + X^H(\mathbf{u}-\mathbf{v}) + X^C(\mathbf{u}-\mathbf{v}) \Big) I_{\mathcal{N}_C}(\mathbf{u}) \\ &= X^C(\mathbf{u}) + \sum_{\mathbf{v} \in L_V^*} X^H(\mathbf{u}-\mathbf{v})\; I_{\mathcal{N}_C}(\mathbf{u}) \\ &= X^C(\mathbf{u}) + Y'(\mathbf{u}) I_{\mathcal{N}_C}(\mathbf{u}) \,, \end{aligned}$$

where

$$Y'(\mathbf{u}) \triangleq \sum_{\mathbf{v} \in L_V^*} X^H(\mathbf{u}-\mathbf{v}) \,. \tag{2.8}$$

It follows that $Y(\mathbf{u})$ in the procedure given previously can be replaced by $Y'(\mathbf{u})$. The advantage is that, as shown below, $Y'(\mathbf{u})$ can be computed with Fourier transforms instead of a summation. To show this, let $\mathcal{S}_V$ denote the vertical sampling operator, which when applied to an image $z(\mathbf{t})$ produces $z_{L_V}(t)$ as defined by (2.2). We recognize the summation in (2.8) as the sampled spectrum when the image $x^H(\mathbf{t})$ is vertically sampled. Since $x^H(\mathbf{t})$,

and consequently $X^H(\mathbf{u})$, can be computed from the horizontal samples,

$$\begin{aligned} Y'(\mathbf{u}) &= \mathcal{F}\{\mathcal{S}_V\{x^H(\mathbf{t})\}\} = \mathcal{F}\{\mathcal{S}_V\{\mathcal{F}^{-1}\{X^H(\mathbf{u})\}\}\} \\ &= \mathcal{F}\Big\{\mathcal{S}_V\Big\{\mathcal{F}^{-1}\{I_{\mathcal{B}_H}(\mathbf{u})\mathcal{F}\{x_H(\mathbf{t})\}\}\Big\}\Big\}. \end{aligned}$$

While the above may initially appear complex[5], in the discrete-space, finite-support case discussed shortly, it will lead to a simple procedure that avoids the summations in (2.6) and (2.8).

Maximality, in the Landau sense, of the set of reconstructable images

The sampling density of an $M(\boldsymbol{\lambda};\mathbf{k})$ M-sampling set is

$$\rho = \frac{k_1 + k_2 - 1}{k_1 k_2 \lambda_1 \lambda_2},$$

since any $k_1\lambda_1 \times k_2\lambda_2$ rectangle in $\mathbb{R}^2$ contains $k_1 + k_2 - 1$ samples. In the frequency domain, the area of the Manhattan-bandlimited region, denoted $|\mathcal{M}(\boldsymbol{\lambda};\mathbf{k})|$, is the sum of the areas of $\mathcal{B}_H$, $\mathcal{B}_V$ and $\mathcal{N}_C$. Alternatively, it is sum of the areas of the Nyquist regions corresponding to the horizontal and vertical sampling lattices, minus the area of their intersection. Either way, this may be written as

$$|\mathcal{B}_H| + |\mathcal{B}_V| + |\mathcal{N}_C| = \frac{1}{k_1\lambda_1\lambda_2} + \frac{1}{k_2\lambda_1\lambda_2} - \frac{1}{k_1k_2\lambda_1\lambda_2},$$

which simplifies to the previous expression for sampling density ρ. Thus, the set of images bandlimited to the Manhattan region $\mathcal{M}(\boldsymbol{\lambda};\mathbf{k})$ is a maximal set of reconstructable images in the Landau sense for the M-sampling grid $M(\boldsymbol{\lambda};\mathbf{k})$.

Discrete-space images

In this section, we briefly consider M-sampling of discrete-space images. Such images might be created by rectangularly sampling a continuous-space image, or they might exist only as discrete-space objects. In any case, we consider an image to be a mapping $x[\mathbf{t}] : \mathbb{T} \to \mathbb{R}$ where $\mathbb{T}$ is either the (infinite) integer lattice $\mathbb{Z}^2$, or a finite subset of the form $\mathbb{T} = \{\mathbf{t} : 0 \le t_1 \le T_1 - 1,\ 0 \le t_2 \le T_2 - 1\}$ for some positive integers T_1, T_2. *Sampling* $x[\mathbf{t}]$ refers to collecting

[5]Note also that the expression (2.8) for $Y'(\mathbf{u})$ contains more terms in the sum than the corresponding expression (2.6) for $Y(\mathbf{u})$.

a subset of its values. In the infinite support case, $\mathbb{T} = \mathbb{Z}^2$, a Manhattan grid $M(\boldsymbol{\lambda}, \mathbf{k})$ is once again defined to be the union of a horizontal lattice $L_H \triangleq L(\lambda_1, k_2\lambda_2)$ and a vertical lattice $L_V \triangleq L(k_1\lambda_1, \lambda_2)$, except that now each lattice must be a sublattice of the integer lattice $\mathbb{Z}^d$, i.e., λ_1, λ_2 must be positive integers. In this case, we assume the discrete-space Fourier transform of $x[\mathbf{t}]$ is well defined and contains no delta functions or other generalized functions. In the finite support case, a Manhattan grid is formed in a similar way, namely, $M(\boldsymbol{\lambda}, \mathbf{k}) = L_H \cup L_V$, where now L_H and L_V are truncated to the finite $\mathbb{T}$.

(a) Infinite-support discrete-space images: In this case, Theorem 1 holds with only trivial changes, as does the reconstruction procedure. Specifically, the only required changes are: (i) replace the continuous-space Fourier transform as the formula for a spectrum with the discrete-space Fourier transform, and (ii) scale all specified frequencies by 2π, such as those defining Nyquist regions and $\mathcal{M}(\boldsymbol{\lambda}; \mathbf{k})$.

(b) Finite-support discrete-space images: In this case, as is customary, we use the Discrete Fourier Transform (DFT) as the formula for the spectrum of an image:

$$X[\mathbf{u}] \;= \sum_{\mathbf{t} \in \mathbb{T}} x[\mathbf{t}]\, e^{-j2\pi(\frac{u_1}{T_1}t_1 + \frac{u_2}{T_2}t_2)}, \quad \mathbf{u} \in \mathbb{T}\,.$$

The conventional sampling theorem (c.f. [33]) for discrete-space images with spatial support $\mathbb{T}$ (defined by T_1, T_2) sampled with a rectangular lattice $L(\alpha_1, \alpha_2)$ limited to $\mathbb{T}$ says that an image $x[\mathbf{t}]$ with support $\mathbb{T}$ can be recovered from its samples in this lattice if T_1 and T_2 are integer multiples of α_1 and α_2, respectively, and its DFT $X[\mathbf{u}]$ is zero outside the (discrete) Nyquist region

$$\widetilde{\mathcal{N}}_{\alpha_1,\alpha_2} \;\triangleq\; \Big\{\mathbf{u} \in \mathbb{T} :\; \text{for } i = 1\,\&\,2,\; 0 \le u_i < \frac{T_i}{2\alpha_i} \text{ or } T_i - \frac{T_i}{2\alpha_i} < u_i \le T_i - 1\Big\}\,.$$

Now suppose a finite-support discrete-space image $x[\mathbf{t}]$ is sampled on the Manhattan grid $M(\boldsymbol{\lambda}, \mathbf{k})$ and is bandlimited to the cross-shaped *Manhattan region*

$$\widetilde{\mathcal{M}}(\boldsymbol{\lambda}, \mathbf{k}) \;\triangleq\; \widetilde{\mathcal{N}}_H \cup \widetilde{\mathcal{N}}_V\,,$$

where $\widetilde{\mathcal{N}}_H$ and $\widetilde{\mathcal{N}}_V$ are the Nyquist regions of the horizontal and vertical lattices, respectively. Assuming T_1 and T_2 are integer multiples of $k_1\lambda_1$ and $k_2\lambda_2$, respectively, a straightforward adaptation of the analysis for continuous-space images shows that from the samples in the vertical lattice L_V, one can recover the spectrum $X[\mathbf{u}]$ in the highpass region $\widetilde{\mathcal{B}}_V \triangleq \widetilde{\mathcal{N}}_V - \widetilde{\mathcal{N}}_C$, where $\widetilde{\mathcal{N}}_V$ and $\widetilde{\mathcal{N}}_C$ are the Nyquist regions of the vertical and coarse lattices, respectively. Specifically, from the spectrum $X_V[\mathbf{u}]$ of the vertically sampled image $x_V[\mathbf{t}]$ (with scaling

as in (2.2)-(2.3)), one recovers $X^V[\mathbf{u}] \triangleq X[\mathbf{u}]I_{\widetilde{\mathcal{B}}_V}[\mathbf{u}] = X_V[\mathbf{u}]I_{\widetilde{\mathcal{B}}_V}[\mathbf{u}]$. Likewise, from the samples in the horizontal lattice L_H, one can recover the spectrum in the highpass region $\widetilde{\mathcal{B}}_H \triangleq \widetilde{\mathcal{N}}_H - \widetilde{\mathcal{N}}_C$ from the spectrum $X_H[\mathbf{u}]$ of the horizontally sampled image $x_H[\mathbf{t}]$ via $X^H[\mathbf{u}] \triangleq X[\mathbf{u}]I_{\widetilde{\mathcal{B}}_H}(\mathbf{u}) = X_H[\mathbf{u}]I_{\widetilde{\mathcal{B}}_H}[\mathbf{u}]$. Finally, the spectrum in the Nyquist region $\widetilde{\mathcal{N}}_C$ of the coarse lattice can be determined via $X^C[\mathbf{u}] = (X_V[\mathbf{u}] - Y[\mathbf{u}])\, I_{\widetilde{\mathcal{N}}_C}[\mathbf{u}]$, where

$$Y[\mathbf{u}] \;=\; k_1\lambda_1\lambda_2 \sum_{r=1}^{k_1\lambda_1-1} X^H\Big[\Big(u_1 - r\frac{T_1}{k_1\lambda_1}\Big) \bmod T_1, u_2\Big]\,.$$

This leads to the following.

Theorem 2. 2D discrete-space, finite-support Manhattan sampling theorem. *If T_1 and T_2 are integer multiples of $k_1\lambda_1$ and $k_2\lambda_2$, respectively, then an image $x[\mathbf{t}]$ with finite support $\mathbb{T}$ can be recovered from its M-samples in $M(\boldsymbol{\lambda}, \mathbf{k})$ if its DFT $X[\mathbf{u}]$ is zero outside the Manhattan region $\widetilde{\mathcal{M}}(\boldsymbol{\lambda}, \mathbf{k})$.*

Reconstruction procedure:

Given the samples in Manhattan grid $M(\boldsymbol{\lambda}, \mathbf{k})$ of an image $x[\mathbf{t}]$ bandlimited to $\widetilde{\mathcal{M}}(\boldsymbol{\lambda}, \mathbf{k})$, the following adaptation of the continuous-space procedure recovers the entire $x[\mathbf{t}]$.

1. Let $x_V[\mathbf{t}]$ equal $k_1\lambda_1\lambda_2\, x[\mathbf{t}]$ on the vertical lattice L_V and zero otherwise, and let $x_H[\mathbf{t}]$ equal $k_2\lambda_1\lambda_2\, x[\mathbf{t}]$ on the horizontal lattice L_H and zero otherwise. Compute $X_V[\mathbf{u}] = \mathrm{DFT}\big\{x_V[\mathbf{t}]\big\}$ and $X_H[\mathbf{u}] = \mathrm{DFT}\big\{x_H[\mathbf{t}]\big\}$.

2. Compute the "alias subtraction" term

$$Y'[\mathbf{u}] \;=\; \mathrm{DFT}\Big\{\widetilde{\mathcal{S}}_V\Big\{\mathrm{IDFT}\big\{I_{\widetilde{\mathcal{B}}_H}[\mathbf{u}]\,\mathrm{DFT}\{x_H[\mathbf{t}]\}\big\}\Big\}\Big\},$$

   where $\widetilde{\mathcal{S}}_V$ denotes the vertical sampling operator that, when applied to an image $z[\mathbf{t}]$, produces an image that is $k_1\lambda_1\lambda_2\, z[\mathbf{t}]$ on L_V, and zero elsewhere.

3. Compute the spectrum:

$$\widehat{X}[\mathbf{u}] \;=\; \begin{cases} X_V[\mathbf{u}], & \mathbf{u} \in \widetilde{\mathcal{B}}_V \\ X_H[\mathbf{u}], & \mathbf{u} \in \widetilde{\mathcal{B}}_H \\ X_V[\mathbf{u}] - Y'[\mathbf{u}], & \mathbf{u} \in \widetilde{N}_C \end{cases}.$$

4. Invert the spectrum:

$$x[\mathbf{t}] \;=\; \mathrm{IDFT}\big\{X[\mathbf{u}]\big\}\,.$$

This reconstruction procedure uses 5 DFT/IDFT operations, each requiring $\mathcal{O}(N \log N)$ arithmetic operations when implemented with an FFT, where $N = T_1 T_2$, plus three pairwise additions of $T_1 \times T_2$ matrices, each requiring $T_1 T_2$ additions, plus instances of setting matrix elements to zero. In summary, the complexity of reconstruction, which is dominated by the FFT's, is $\mathcal{O}(N \log N)$ operations per image.

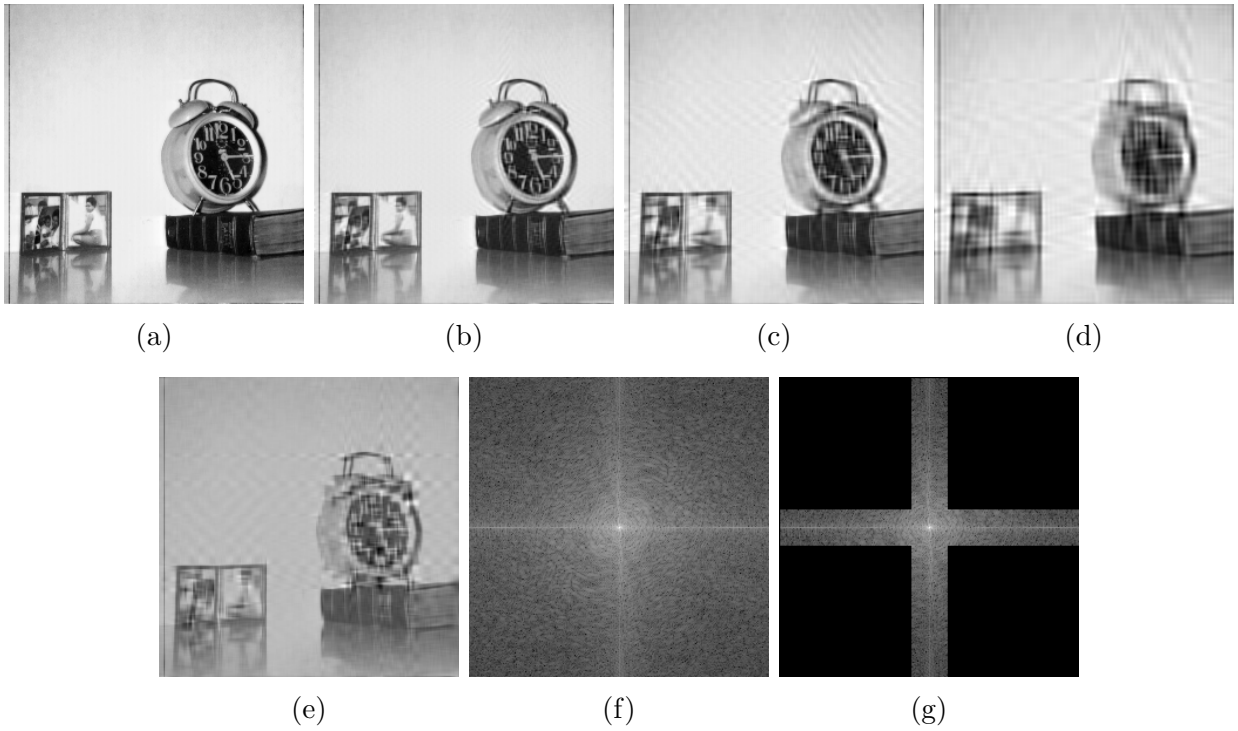


Figure 2.3: (a) Original 256×256 image. (b) Image bandlimited to Manhattan region $\widetilde{\mathcal{M}}(\boldsymbol{\lambda}, \mathbf{k})$, with $k_1 = k_2 = 4$ and $\lambda_1 = \lambda_2 = 1$. (c) Same as (b) except $k_1 = k_2 = 8$. (d) Same as (c) except $\lambda_1 = \lambda_2 = 2$. (Note: after spectra were zeroed outside $\widetilde{\mathcal{M}}(\boldsymbol{\lambda}, \mathbf{k})$, inverse transforms were applied, negligible imaginary parts were discarded, and images were quantized to $\{0, 1, \ldots, 255\}$.) (e) Image sampled with parameters of (c) and reconstructed without first bandlimiting to Manhattan region. Log magnitude spectra: (f) original image; (g) original image bandlimited with parameters of (c).

Note that few real world, finite-support images will satisfy the conditions of Theorem 2. As a result, to apply M-sampling to a real world image, the image can be pre-processed by zero-padding so that its dimensions are multiples of $k_1\lambda_1$ and $k_2\lambda_2$, respectively, and "Manhattan filtering" by taking the DFT and setting to zero all coefficients outside of $\widetilde{\mathcal{M}}(\boldsymbol{\lambda}, \mathbf{k})$. Such padded and filtered images can be recovered perfectly from their M-samples. To illustrate the effects of such filtering, which heavily suppresses diagonal frequencies, Fig. 2.3(a-d) shows a finite-support image and its filtering with several choices of parameters. Figures 2.3(f,g) show the spectra of the image before and after bandlimiting, and Figure 2.3(e)

shows the the effect of sampling and reconstruction without first pre-filtering. Note that the image was chosen to have sharp edges surrounded by a smooth background in order that one can easily see the ringing due to bandlimiting.

2.4 Higher-Dimensional Manhattan Sampling

2.4.1 Introduction

As mentioned earlier, in any dimension $d \geq 3$ there are a number of possible d-dimensional Manhattan sets. Each is a finite union of rectangular lattices, each defined by step sizes that in dimension i are constrained to be λ_i or $k_i\lambda_i$, where each λ_i is a positive constant called the *dense spacing* in dimension i, and each k_i is an integer greater than 1 called the *sampling factor* in dimension i. Such a rectangular lattice will be called a $(\boldsymbol{\lambda}, \mathbf{k})$-lattice, where $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_d)$ is its *dense spacing vector* and $\mathbf{k} = (k_1, \ldots, k_d)$ is its *sampling factor vector*. It will also be called a $(d, \boldsymbol{\lambda}, \mathbf{k})$-lattice when we wish to emphasize d, and a *bi-step lattice* when we do not wish to specify parameters. (The term "bi-step" emphasizes that each step size α_i can only take one of two values: λ_i or $k_i\lambda_i$). Accordingly, to specify a d-dimensional *Manhattan set*, one specifies a dense spacing vector $\boldsymbol{\lambda}$, a sampling factor vector $\mathbf{k}$, and a collection of $(\boldsymbol{\lambda}, \mathbf{k})$-lattices.

To efficiently characterize a $(\boldsymbol{\lambda}, \mathbf{k})$-lattice, we let $\mathbf{b} = (b_1, \ldots, b_d)$ denote a d-dimensional vector, called its *bi-step indicator vector*, or more concisely *bi-step vector*, that indicates the dimensions along which the bi-step lattice is dense, according to the convention $b_i = 1$ if the step size is λ_i in dimension i and 0 if the step size is $k_i\lambda_i$. Thus, the $(\boldsymbol{\lambda}, \mathbf{k})$-lattice specified by $\mathbf{b}$ is $L_{\boldsymbol{\lambda},\mathbf{k},\mathbf{b}} \triangleq L(\boldsymbol{\alpha}_\mathbf{b})$, with $\boldsymbol{\alpha}_\mathbf{b} = (\alpha_{\mathbf{b},1}, \ldots, \alpha_{\mathbf{b},d})$ defined by

$$\alpha_{\mathbf{b},i} \;=\; \begin{cases} \lambda_i, & b_i = 1 \\ k_i\lambda_i, & b_i = 0 \end{cases}. \tag{2.9}$$

or equivalently,

$$L_{\boldsymbol{\lambda},\mathbf{k},\mathbf{b}} \triangleq \left\{\mathbf{t} : t_i \text{ is a multiple of } k_i\lambda_i \text{ for } i \text{ s.t. } b_i = 0, \text{ and a multiple of } \lambda_i \text{ for other } i\right\}. \tag{2.10}$$

Note that we generally consider d, $\boldsymbol{\lambda}$ and $\mathbf{k}$ to be fixed, and so as an abbreviation and slight abuse of notation, we usually write $L_\mathbf{b}$ instead of $L_{\boldsymbol{\lambda},\mathbf{k},\mathbf{b}}$. It will also be useful to let $x_\mathbf{b}(\mathbf{t})$ and $X_\mathbf{b}(\mathbf{u})$ denote, respectively, the sampled image and the sampled spectrum due to sampling $x(\mathbf{t})$ with $L_\mathbf{b}$.

The following summarizes.

Definition 3. *Given dimension d, dense spacing vector $\boldsymbol{\lambda}$, sampling factor vector $\mathbf{k}$ (all of its components are integers greater than 1), and a finite collection of $(d, \boldsymbol{\lambda}, \mathbf{k})$-lattices specified by the bi-step vectors in $B = \{\mathbf{b}_1, \ldots \mathbf{b}_m\}$, the corresponding* $(d, \boldsymbol{\lambda}, \mathbf{k}, B)$-Manhattan (sampling) set *is*

$$M\big(d, \boldsymbol{\lambda}, \mathbf{k}, B\big) \triangleq \bigcup_{j=1}^{m} L_{\mathbf{b}_j} . \tag{2.11}$$

As $d, \boldsymbol{\lambda}$ and $\mathbf{k}$ will be considered fixed, we usually write $M\big(B\big)$ instead of $M\big(d, \boldsymbol{\lambda}, \mathbf{k}, B\big)$. B will be called a *Manhattan collection* or *M-collection* for short.

2.4.2 Examples and properties of bi-step lattices

It is useful to call attention to certain bi-step lattices. One is the *dense lattice* $L_{\mathbf{1}}$ corresponding to the bi-step vector $\mathbf{b} = \mathbf{1} \triangleq (1, \ldots, 1)$. It is a rectangular lattice with step vector $\boldsymbol{\lambda}$ that contains every other $(\boldsymbol{\lambda}, \mathbf{k})$-lattice. Another is the *coarse lattice* $L_{\mathbf{0}}$ corresponding to $\mathbf{b} = \mathbf{0} \triangleq (0, \ldots, 0)$, which is the rectangular lattice with step vector $\boldsymbol{\alpha} = \mathbf{k} \odot \boldsymbol{\lambda}$ and which is contained in every other $(\boldsymbol{\lambda}, \mathbf{k})$-lattice. As mentioned in the introduction for 3D Manhattan sets, it will be useful to consider the partition of $\mathbb{R}^d$ induced by the coarse lattice, whose cells are $k_1\lambda_1 \times \ldots \times k_d\lambda_d$ orthotopes (hyper-rectangles) with corners at lattice points. These orthotopes will be called *fundamental cells*. The coarse lattice contains just the corners of these fundamental cells; other $(\boldsymbol{\lambda}, \mathbf{k})$-lattices may contain points on their edges and faces, but only the dense lattice $L_{\mathbf{1}}$ contains points in their interiors.

A third lattice to consider is $L_{\mathbf{e}_i}$ corresponding to bi-step vector $\mathbf{b} = \mathbf{e}_i$, which can be viewed as a collection of points spaced densely on lines parallel to $\mathbf{e}_i$, with one line passing through each point of $(d-1)$-dimensional cubic lattice $L(k_1\lambda_1, \ldots, k_{i-1}\lambda_{i-1}, k_{i+1}\lambda_{i+1}, \ldots, k_d\lambda_d)$. Finally, we mention the lattice corresponding to $\mathbf{b} = \mathbf{1} - \mathbf{e}_i$, which can be viewed as sampling densely on shifts of the $d-1$ dimensional lattice $L(\lambda_1, \ldots, \lambda_{i-1}, \lambda_{i+1}, \ldots, \lambda_d)$ spaced $k_i\lambda_i$ apart.

Given two (binary) bi-step vectors $\mathbf{b}_1$ and $\mathbf{b}_2$, define their *union* $\mathbf{b}_1 \vee \mathbf{b}_2$ and *intersection* $\mathbf{b}_1 \wedge \mathbf{b}_2$ to be their element-wise 'OR' and 'AND', respectively, and define $\mathbf{b}_1 \subset \mathbf{b}_2$ to mean $\mathbf{b}_1 \wedge \mathbf{b}_2 = \mathbf{b}_1$. Define the complement to be $\mathbf{b}^c \triangleq \mathbf{1} - \mathbf{b}$, and the *Hamming weight* or simply *weight* $\|\mathbf{b}\|$ to be the number of ones contained in $\mathbf{b}$.

The following are useful properties of $\mathbf{b}$ representations of bi-step lattices.

Fact 4. *Considering $(d, \boldsymbol{\lambda}, \mathbf{k})$-lattices,*

(a) $L_{\mathbf{b}_1} \subset L_{\mathbf{b}_2}$ *if and only if* $\mathbf{b}_1 \subset \mathbf{b}_2$,

(b) $L_{\mathbf{b}_1} = L_{\mathbf{b}_2}$ *if and only if* $\mathbf{b}_1 = \mathbf{b}_2$,

(c) $L_{\mathbf{b}_1} \cap L_{\mathbf{b}_2} \;=\; L_{\mathbf{b}_1 \wedge \mathbf{b}_2}$,

(d) *If* $L_{\widetilde{\mathbf{b}}} \subset \bigcup_{j=1}^{m} L_{\mathbf{b}_j}$, *then for some* j, $L_{\widetilde{\mathbf{b}}} \subset L_{\mathbf{b}_j}$, *and consequently from (a),* $\widetilde{\mathbf{b}} \subset \mathbf{b}_j$.

Proof:

(a) and (b) are elementary.

(c) $L_{\mathbf{b}_1} \cap L_{\mathbf{b}_2}$

$= \{\mathbf{t} : t_i$ is multiple of $k_i\lambda_i \; \forall i$ s.t. $b_{1,i} = 0$ or $b_{2,i} = 0$, and t_i is multiple of λ_i for other $i\}$

$= \{\mathbf{t} : t_i$ is multiple of $k_i\lambda_i$ for all i s.t. $(\mathbf{b}_1 \wedge \mathbf{b}_2)_i = 0$ and t_i is multiple of λ_i for other $i\}$

$= L_{\mathbf{b}_1 \wedge \mathbf{b}_2}$.

(d) As is well known, for $m = 2$ this property derives from just the group nature of lattices, but not for larger values of m. Accordingly, to prove it for arbitrary m, we need to use properties of $(\boldsymbol{\lambda}, \mathbf{k})$ lattices. Specifically, we demonstrate the contrapositive. Suppose $L_{\widetilde{\mathbf{b}}} \not\subset L_{\mathbf{b}_j}$, $j = 1, \ldots, m$. Then from (a), for each $1 \le j \le m$, $\widetilde{\mathbf{b}} \not\subset \mathbf{b}_j$, and so there exists i_j such that $\widetilde{b}_{i_j} = 1$ and $b_{j,i_j} = 0$. Let I denote the set of all such i_j's, and let $\mathbf{t} = (t_1, \ldots, t_d)$ be defined by

$$t_i = \begin{cases} (k_i + 1)\lambda_i, & \text{if } i \in I \\ k_i\lambda_i, & \text{otherwise} \end{cases} .$$

Referring to (2.10), we see that $\mathbf{t} \in L_{\widetilde{\mathbf{b}}}$, because the only dimensions i for which t_i is not a multiple of $k_i\lambda_i$ are those in I, in which case $\widetilde{b}_i = 1$, i.e., the lattice $L_{\widetilde{\mathbf{b}}}$ is dense in dimension i. Moreover, again referring to (2.10), we see that for each $j \in \{1, \ldots, m\}$, $\mathbf{t} \notin L_{\mathbf{b}_j}$, because $t_{i_j} = (k_{i_j} + 1)\lambda_i$ is not a multiple of $k_{i_j}\lambda_i$, yet $b_{j,i_j} = 0$, i.e., the lattice $L_{\mathbf{b}_j}$ is coarse in dimension i_j. It follows that $\mathbf{t} \notin \cup_{j=1}^{m} L_{\mathbf{b}_j}$. Hence, $L_{\widetilde{\mathbf{b}}} \not\subset \cup_{j=1}^{m} L_{\mathbf{b}_j}$. □

Among other things, (b) shows there is a one-to-one correspondence between bi-step vectors $\mathbf{b}$ and $(d, \boldsymbol{\lambda}, \mathbf{k})$ lattices. which verifies that the $\mathbf{b}$'s are valid representations of the $(\boldsymbol{\lambda}, \mathbf{k})$-lattices. Also, since there are 2^d possible bi-step vectors $\mathbf{b}$, it follows that there are 2^d distinct $(d, \boldsymbol{\lambda}, \mathbf{k})$-lattices.

Note that while (c) shows that the intersection of two $(d, \boldsymbol{\lambda}, \mathbf{k})$-lattices is another $(d, \boldsymbol{\lambda}, \mathbf{k})$-lattice, such is not true for the union, which is why a Manhattan set is not ordinarily a lattice.

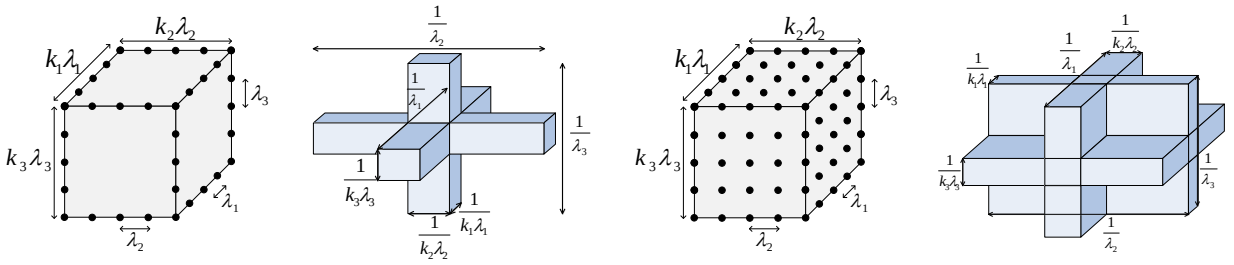


Figure 2.4: Examples of 3D Manhattan sampling $M(B)$ and their corresponding Manhattan regions $\mathcal{M}(B)$. (a) Manhattan lines $B = \{(1,0,0),(0,1,0),(0,0,1)\}$, and (b) its corresponding Manhattan region. (c) Manhattan facets $B = \{(1,1,0),(1,0,1),(0,1,1)\}$, and (d) its corresponding Manhattan region.

For example, when $d = 2$, $L_{\mathbf{e}_1} \cup L_{\mathbf{e}_2}$ is the Manhattan set $M\big(\{\mathbf{e}_1, \mathbf{e}_2\}\big)$ shown in Fig. 2.1(a), which is not a lattice and does not equal $L_{\mathbf{e}_1 \vee \mathbf{e}_2}$, which is the dense lattice $L_{(1,1)}$.

2.4.3 Examples of Manhattan sets

Several types of Manhattan sets deserve special attention.

1. *Manhattan lines* is a Manhattan set $M(B)$ specified by $B = \{\mathbf{e}_1, \ldots, \mathbf{e}_d\}$. In this case the samples are taken on the 1D edges of the fundamental cells, i.e., on d orthogonal sets of parallel lines in $\mathbb{R}^d$. See Figures 2.1(a) and 2.4(a) for illustrations of Manhattan lines in two and three dimensions, respectively.

2. *Manhattan facets* is a Manhattan set $M(B)$ specified by $B = \{\mathbf{e}_1^c, \ldots, \mathbf{e}_d^c\}$. Sampling on a set of Manhattan facets is analogous to sampling densely along d orthogonal sets of parallel hyperplanes in $\mathbb{R}^d$. See Figures 2.1(a) and 2.4(c) for illustrations of Manhattan facets in two and three dimensions, respectively. For $d = 2$, Manhattan facets and lines are identical.

3. Though technically any $(\boldsymbol{\lambda}, \mathbf{k})$-lattice, including the coarse and dense lattices, is a Manhattan set, we focus on Manhattan sets that are not lattices, which we call *proper*.

4. Video sampling: Let $d = 3$, let $i = 1, 2$ be spatial dimensions and let $i = 3$ be a temporal dimension. Whereas video is most commonly sampled with a rectangular lattice, say $L(\lambda_1, \lambda_2, \lambda_3)$, other samplings are possible, for example, the Manhattan set $M(3, \boldsymbol{\lambda}, \mathbf{k}, B)$ specified by $B = \{\mathbf{e}_3^c, \mathbf{e}_3\}$ uses fine spatial sampling every $k_3\lambda_3$ seconds and spatial subsampling with factors k_1 and k_2 at times that are other multiples of λ_3 seconds.

2.4.4 Alternate representations of Manhattan sets

By the definition of a Manhattan set (2.11) and Fact 4(a), augmenting an M-collection B by a subset $\mathbf{b}'$ of some $\mathbf{b}$ in B does not change the resulting Manhattan set. Thus many M-collections can generate the same Manhattan set. There are, however, unique largest and smallest M-collections that generate any given Manhattan set. To find these, we make use of the following.

Fact 5. *Let B and B' be M-collections. Then,*

(a) *$M(B) \subset M(B')$ if $B \subset B'$.*

(b) *$M(B) \subset M(B')$ if and only if for each $\mathbf{b} \in B$ there is $\mathbf{b}' \in B'$ such that $L_{\mathbf{b}} \subset L_{\mathbf{b}'}$, or equivalently by Fact 4(a), $\mathbf{b} \subset \mathbf{b}'$.*

(c) *It is not true that $M(B) = M(B')$ implies $B = B'$, or that $M(B) \subset M(B')$ implies $B \subset B'$.*

Proof:

(a) is obvious.

(b) If for each $\mathbf{b} \in B$ there is $\mathbf{b}' \in B'$ s.t. $L_{\mathbf{b}} \subset L_{\mathbf{b}'}$, then $M(B) = \cup_{\mathbf{b}\in B} L_{\mathbf{b}} \subset \cup_{\mathbf{b}'\in B'} L_{\mathbf{b}'} = M(B')$. Conversely, if $M(B) \subset M(B')$, then for each $\mathbf{b} \in B$, $L_{\mathbf{b}} \subset M(B') = \cup_{b'\in B'} L_{\mathbf{b}'}$, and Fact 4(d) implies $L_{\mathbf{b}} \subset L_{\mathbf{b}'}$ for some $\mathbf{b}' \in B'$.

(c) Part (a) shows that adding to B some subset of some $\mathbf{b} \in B$ that is not already in B yields $B' \neq B$ such that $M(B) = M(B')$. In this same case, $M(B') \subset M(B)$, but $B' \not\subset B$. □

The unique largest M-collection that generates $M(B)$ is

$$\overline{B} \triangleq \left\{\mathbf{b}' : \mathbf{b}' \subset \mathbf{b} \text{ for some } \mathbf{b} \in B\right\},$$

which we call the *closure* of B. Fact 5(a) implies $M(B) \subset M(\overline{B}) = M(B)$, and Fact 5(b) implies $M(\overline{B}) \subset M(B)$. Hence $M(\overline{B}) = M(B)$. $M(\overline{B})$ is the largest M-collection generating $M(B)$ because if $M(B') = M(B)$, and $\mathbf{b}' \in B'$, then Fact 5(b) implies there is a $\mathbf{b} \in B$ such that $L_{\mathbf{b}'} \subset L_{\mathbf{b}}$, and this implies $\mathbf{b}' \in \overline{B}$. Hence, $B' \subset \overline{B}$.

Removing all elements of an M-collection B that are subsets of another element results in the unique smallest M-collection generating $M(B)$, which we denote $\underline{B}$.

2.4.5 Manhattan sampling density

The *density* of a Manhattan set $M(B)$, i.e., the number of samples per unit area, is obviously less than the sum of the densities of the lattices of which it is the union, because each lattice contains all points in the coarse lattice $L_{\mathbf{0}}$. Accordingly, we partition the dense lattice $L_{\mathbf{1}}$ in such a way that for any B, $M(B)$ is the union of atoms of this partition, and its density can be computed by summing their densities.

To obtain a suitable partition, let us group into one atom all sites $\mathbf{t}$ of $L_{\mathbf{1}}$ having the same answers to the following d questions – "Is t_i not a multiple of $\lambda_i k_i$?" – for $i = 1, \ldots, d$. Specifically, with a binary vector $\mathbf{b} = (b_1, \ldots, b_d)$ indicating the set of i's for which the answers are "yes", consider the partition $\left\{V_{\mathbf{b}} : \mathbf{b} \subset \{0,1\}^d\right\}$, where the atom corresponding to $\mathbf{b}$ is

$$\begin{aligned} V_{\mathbf{b}} \triangleq \big\{\mathbf{t} :\ & t_i \text{ is a multiple of } k_i\lambda_i \text{ for } i \text{ s.t. } b_i = 0, \\ & \text{and } t_i \text{ is a multiple of } \lambda_i, \text{ but not } k_i\lambda_i, \text{ for } i \text{ s.t. } b_i = 1\big\}. \end{aligned}$$

Fig. 2.5 illustrates the partitioning of a 2D Manhattan grid. Essentially, it is a partition of the dense lattice into collections of cosets of the coarse lattice.

It is clear from the definition that no $\mathbf{t}$ can lie in both $V_{\mathbf{b}}$ and $V_{\mathbf{b}'}$ for $\mathbf{b} \neq \mathbf{b}'$. Hence, the $V_{\mathbf{b}}$'s are disjoint. By comparing the above to the definition (2.10) of $L_{\mathbf{b}}$, one sees that $V_{\mathbf{b}'} \subset L_{\mathbf{b}}$ if and only if $\mathbf{b}' \subset \mathbf{b}$. It follows that for any $\mathbf{b}$, $\cup_{\mathbf{b}' \subset b} V_{\mathbf{b}'} \subset L_{\mathbf{b}}$. Conversely, if $\mathbf{t} \in L_{\mathbf{b}}$, then it is easily seen that $\mathbf{t} \in L_{\mathbf{b}'}$ for $\mathbf{b}'$ defined by $b'_i = 0$ for i such that t_i is a multiple of $\lambda_i k_i$ and $b'_i = 1$ otherwise. It follows that $\cup_{\mathbf{b}' \subset b} V_{\mathbf{b}'} = L_{\mathbf{b}}$, i.e., $\{V_{\mathbf{b}}\}$ partitions any bi-step lattice, including $L_{\mathbf{1}}$. Moreover, since $M(B) = \cup_{\mathbf{b} \in B} L_{\mathbf{b}}$, one can also write $M(B) = \cup_{\mathbf{b} \in \overline{B}} V_{\mathbf{b}}$, i.e., $\{V_{\mathbf{b}}\}$ partitions any Manhattan grid.

With this partition in mind, the density of the Manhattan set $M(B)$ is now obtained by summing the densities of each $V_{\mathbf{b}}$, $\mathbf{b} \in \overline{B}$. Consider the points of $V_{\mathbf{b}}$ in the fundamental cell

$$F_{\mathbf{k},\boldsymbol{\lambda}} \triangleq \bigtimes_{i=1}^{d} [0, k_i\lambda_i)\,,$$

which has a corner at the origin, lies entirely in the positive hyper-quadrant, and has volume is $\prod_{i=1}^{d} k_i\lambda_i$. We see that $V_{\mathbf{b}} \cap F_{\mathbf{k},\boldsymbol{\lambda}}$ is the Cartesian product of d sets $A_1, \ldots, A_d$, where $A_i = \{\lambda_i, 2\lambda_i, \ldots, (k_i - 1)\lambda_i\}$ if $b_i = 1$ and $A_i = \{0\}$ if $b_i = 0$. Since $V_{\mathbf{b}} \cap F_{\mathbf{k},\boldsymbol{\lambda}}$ contains $\prod_{i:b_i=1}(k_i - 1)$ points, and since the density of $V_{\mathbf{b}}$ is this number divided by the volume of

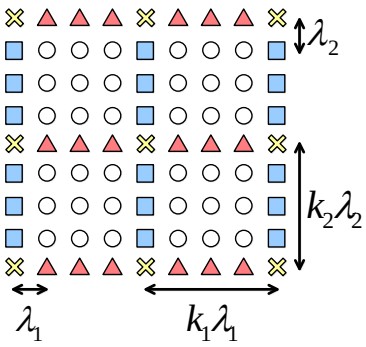


Figure 2.5: Partitioning of 2D Manhattan grid $M(\{\mathbf{e}_1, \mathbf{e}_2\})$ with sampling factors $k_1 = k_2 = 4$, which is the union of the yellow $\times$'s in $V_{(0,0)}$, the red $\triangle$'s in $V_{(1,0)}$, and the blue $\square$'s in $V_{(0,2)}$. The white $\circ$'s are in $V_{(1,1)}$, which is disjoint from this Manhattan grid.

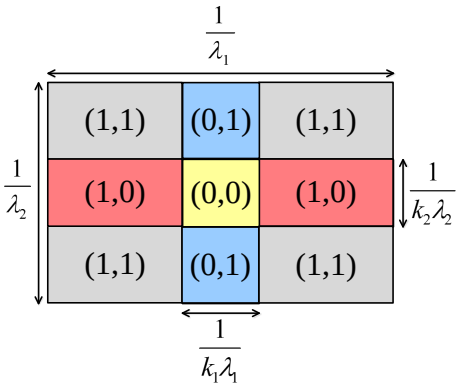


Figure 2.6: M-partition of $\mathcal{N}_{\mathbb{D}}$ for $d = 2$, $k_1 = 5$, $k_2 = 3$. Frequency $\mathbf{u} = 0$ lies at the center. Each M-atom $A^{\mathbf{b}}$ is identified by its $\mathbf{b}$. Note that the cross-shaped Manhattan region $\mathcal{M}(\{\mathbf{e}_1, \mathbf{e}_2\})$ is also partitioned by M-atoms; in particular, $\mathcal{M}(\{\mathbf{e}_1, \mathbf{e}_2\}) = A^{(0,0)} \cup A^{(1,0)} \cup A^{(0,1)}$.

$F_{\mathbf{k},\boldsymbol{\lambda}}$, the density of $M(B)$ is

$$\rho(B) \;=\; \frac{\sum_{\mathbf{b}\in\overline{B}} \prod_{i:b_i=1}(k_i - 1)}{\prod_{i=1}^{d} k_i} \times \frac{1}{\prod_{i=1}^{d} \lambda_i}\,. \tag{2.12}$$

For example, the densities of several Manhattan sets in three dimensions are given in Table 2.1.

Manhattan set	Γ	$\rho(\Gamma)$
Manhattan lines	$\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$	$(3k-2)/k^3$
Video sampling example	$\{\mathbf{e}_3^c, \mathbf{e}_3\}$	$(k^2+k-1)/k^3$
Manhattan facets	$\{\mathbf{e}_1^c, \mathbf{e}_2^c, \mathbf{e}_3^c\}$	$(3k^2-3k+1)/k^3$

Table 2.1: Density of several 3-dimensional Manhattan sets with $k_i = k$ and $\lambda_i = 1$ for all i.

2.4.6 Manhattan partition of frequency space

As mentioned earlier, our approach to reconstructing an appropriately bandlimited image $x(\mathbf{t})$ from M-samples $M(B)$ involves sequentially reconstructing regions of its spectrum $X(\mathbf{u})$. Specifically, each region will be recovered from the samples in some collection of bi-step lattices contained in the Manhattan set. This section describes a partition of frequency

space, some of whose atoms will be the reconstructable regions.

Let $\mathcal{N}_{\mathbf{b}}$ denote the Nyquist region for bi-step lattice $L_{\mathbf{b}}$, i.e.,

$$\mathcal{N}_{\mathbf{b}} \triangleq \left\{\mathbf{u} : |u_i| < \frac{1}{2\alpha_{\mathbf{b},i}},\ i = 1, \ldots, d\right\},$$

with step sizes $\alpha_{\mathbf{b},i}$ given by (2.9). For future reference we note that

$$\mathcal{N}_{\mathbf{b}'} \subset \mathcal{N}_{\mathbf{b}} \text{ if and only if } \mathbf{b}' \subset \mathbf{b}\,. \tag{2.13}$$

Since any $(d, \boldsymbol{\lambda}, \mathbf{k})$ Manhattan set is a subset of the dense lattice $L_{\mathbf{1}}$, it follows that the appropriate bandlimitation for images reconstructable from any such Manhattan set or any $(d, \boldsymbol{\lambda}, \mathbf{k})$ lattice is a subset of the Nyquist region of the dense lattice, namely,

$$\mathcal{N}_{\mathbf{1}} = \left\{\mathbf{u} : |\mathbf{u}_i| < \frac{1}{2\lambda_i},\ i = 1, \ldots, d\right\}.$$

Thus, we need only partition $\mathcal{N}_{\mathbf{1}}$.

Definition 6. *The* Manhattan partition *(*M-partition*) of* $\mathcal{N}_{\mathbf{1}}$ *is* $\left\{\mathcal{A}^{\mathbf{b}} : \mathbf{b} \in \{0,1\}^d\right\}$ *where* $\mathcal{A}^{\mathbf{b}}$ *is the* Manhattan atom[6] (M-atom*)*

$$\mathcal{A}^{\mathbf{b}} \triangleq a_1^{\mathbf{b}} \times \cdots \times a_d^{\mathbf{b}}\,,$$

and $a_i^{\mathbf{b}}$ *is the interval, or union of two intervals,*

$$a_i^{\mathbf{b}} \triangleq \begin{cases} \left\{u_i : \frac{1}{2k_i\lambda_i} \le |u_i| < \frac{1}{2\lambda_i}\right)\}, & b_i = 1 \\ \left\{u_i : |u_i| < \frac{1}{2k_i\lambda_i}\right\}, & b_i = 0 \end{cases}.$$

Thus, $\mathcal{A}^{\mathbf{b}}$ *is highpass for all dimensions such that* $b_i = 1$ *and lowpass for all other dimensions. The* weight *of atom* $\mathcal{A}^{\mathbf{b}}$ *is* $\|\mathbf{b}\|$*, the weight of* $\mathbf{b}$*.*

The M-partition is illustrated in Fig. 2.6 in the case of $d = 2$, $k_1 = 5$, $k_2 = 3$. We now make several easy to deduce, but important, observations.

1. M-atom $\mathcal{A}^{\mathbf{b}}$ is the Cartesian product of lowpass intervals $(-\frac{1}{2k_i\lambda_i}, \frac{1}{2k_i\lambda_i})$ along each dimension such that $b_i = 0$, and of the union of two highpass intervals $(-\frac{1}{2\lambda_i}, -\frac{1}{2k_i\lambda_i}] \cup$

[6]Consistent with previous conventions, $\mathbf{b}$ is a superscript because it determines a frequency region (an M-atom $\mathcal{A}^{\mathbf{b}}$), whereas it is a subscript when specifying the Nyquist region $\mathcal{N}_{\mathbf{b}}$ of a bi-step lattice $L_{\mathbf{b}}$, precisely because it prescribes a sampling. Except for $\mathbf{b} = \mathbf{0}$, the M-atoms are highpass regions, whereas Nyquist regions are lowpass.

$[\frac{1}{2k_i\lambda_i}, \frac{1}{2\lambda_i})$ along each dimension such that $b_i = 1$. Thus, $\mathcal{A}^{\mathbf{b}}$ is the union of $2^{\|\mathbf{b}\|}$ disjoint orthotopes in $\mathbb{R}^d$.

2. The M-atoms are disjoint, and their union is $\mathcal{N}_{\mathbf{1}}$. Hence, they comprise a partition of $\mathcal{N}_{\mathbf{1}}$.

3. The M-atom $\mathcal{A}^{\mathbf{b}}$ is a subset of the Nyquist region $\mathcal{N}_{\mathbf{b}}$. Equality holds only for $\mathbf{b} = \mathbf{0}$.

4. The weight $\|\mathbf{b}\|$ of an atom $\mathcal{A}^{\mathbf{b}}$ is a rough indicator of how highpass or lowpass is the atom.

5. The lowest weight M-atom, $\mathcal{A}^{\mathbf{0}}$, is lowpass in all dimensions and equals the Nyquist region $\mathcal{N}_{\mathbf{0}}$ of the coarse lattice $L_{\mathbf{0}}$, which is the bi-step lattice with smallest and lowest frequency Nyquist region.

6. The highest weight M-atom, $\mathcal{A}^{\mathbf{1}}$, is highpass in all dimensions and contains the high-pass "corners" of $\mathcal{N}_{\mathbf{1}}$. Its volume is at least large as that of any other atom, and usually larger. We will see later that no proper $(d, \boldsymbol{\lambda}, \mathbf{k})$ Manhattan set permits the reconstruction of these corners since they can only be recovered by sampling densely along every dimension, i.e., they are only recoverable if we sample on the dense lattice $L_{\mathbf{1}}$.

7. If an image $x(\mathbf{t})$ is bandlimited to $\mathcal{N}_{\mathbf{1}}$, then both $x(\mathbf{t})$ and its spectrum $X(\mathbf{u})$ can be decomposed into sums of M-atom components:

$$x(\mathbf{t}) = \sum_{\mathbf{b}\in\{0,1\}^d} x^{\mathbf{b}}(\mathbf{t}),$$
$$X(\mathbf{u}) = \sum_{\mathbf{b}\in\{0,1\}^d} X^{\mathbf{b}}(\mathbf{u}), \tag{2.14}$$

where $X^{\mathbf{b}}(\mathbf{u}) = X(\mathbf{u})$ for $\mathbf{u} \in \mathcal{A}^{\mathbf{b}}$, $X^{\mathbf{b}}(\mathbf{u}) = 0$ otherwise, and $x^{\mathbf{b}}(\mathbf{t})$ is the inverse transform of $X^{\mathbf{b}}(\mathbf{u})$. We shall refer to $x^b(\mathbf{t})$ and $X^{\mathbf{b}}(\mathbf{u})$ as *Manhattan atoms*, or simply *atoms*, of $x(\mathbf{t})$ and $X(\mathbf{u})$, respectively.

It will also be important that the M-atoms can partition the Nyquist region $\mathcal{N}_{\mathbf{b}}$ corresponding to any bi-step lattice $L_{\mathbf{b}}$, as shown below.

Fact 7.

(a) *For any* $\mathbf{b} \in \{0,1\}^d$*, the* $2^{\|\mathbf{b}\|}$ *M-atoms in* $\{\mathcal{A}^{\mathbf{b}'} : \mathbf{b}' \subset \mathbf{b}\}$ *partition* $\mathcal{N}_{\mathbf{b}}$ *in the sense that* $\mathcal{N}_{\mathbf{b}} = \cup_{\mathbf{b}'\subset\mathbf{b}}\mathcal{A}^{\mathbf{b}'}$.

(b) $\mathbf{b}' \subset \mathbf{b}$ *if and only if* $\mathcal{A}^{\mathbf{b}'} \subset \mathcal{N}_{\mathbf{b}}$.

Proof: (a) Since the elements of $\{\mathcal{A}^{\mathbf{b}'} : \mathbf{b}' \subset \mathbf{b}\}$ are disjoint and contained in $\mathcal{N}_{\mathbf{b}}$, it suffices to show

$$\mathcal{N}_{\mathbf{b}} \;\subset\; \bigcup_{\mathbf{b}' \subset \mathbf{b}} \mathcal{A}^{\mathbf{b}'} \,. \tag{2.15}$$

Accordingly, suppose $\mathbf{u} \in \mathcal{N}_{\mathbf{b}}$. It is then easy to see that $\mathbf{u} \in \mathcal{A}^{\mathbf{b}'}$, where

$$b'_i = \begin{cases} 1, & \frac{1}{2k_i\lambda_i} \le |u_i| < \frac{1}{2\lambda_i} \\ 0, & |u_i| \le \frac{1}{2k_i\lambda_i} \end{cases} ,$$

which demonstrates (2.15).

(b) First, if $\mathbf{b}' \subset \mathbf{b}$, then by (2.13) and the third observation after the definition of M-atom, $\mathcal{A}^{\mathbf{b}'} \subset \mathcal{N}_{\mathbf{b}'} \subset \mathcal{N}_{\mathbf{b}}$. Conversely, if $\mathcal{A}^{\mathbf{b}'} \subset \mathcal{N}_{\mathbf{b}}$, then by part (a) $\mathcal{A}^{\mathbf{b}'}$ must be one of the atoms whose union is $\mathcal{N}_{\mathbf{b}}$. Hence, $\mathbf{b}' \subset \mathbf{b}$. □

2.4.7 Spectral replication induced by bi-step lattice sampling

For a Manhattan set $M(B)$, the reconstruction algorithm to follow will reconstruct an image $x(\mathbf{t})$ by reconstructing its spectrum $X(\mathbf{u})$ one Manhattan atom at a time, in an order to be specified later. In particular, for each $\mathbf{b} \in B$, it will reconstruct atom $X^{\mathbf{b}}(\mathbf{u})$ from the subset of samples corresponding to the bi-step lattice $L_{\mathbf{b}}$, taking into account the aliasing due to previously reconstructed atoms. Using the more suggestive $\mathbf{s} = (s_1, \ldots, s_d)$, rather than $\mathbf{b}$, to denote a bi-step vector that characterizes a sampling, then as reviewed in Section 2.2, sampling with $L_{\mathbf{s}}$ replicates the spectrum $X(\mathbf{u})$ at all sites in the reciprocal lattice $L^*_{\mathbf{s}}$. Moreover, if $X(\mathbf{u})$ is bandlimited to $\mathcal{N}_{\mathbf{1}}$, substituting (2.14) into (2.4) gives the following decomposition of the sampled spectrum:

$$X_{\mathbf{s}}(\mathbf{u}) \;=\; \sum_{\mathbf{v} \in L^*_{\mathbf{s}}} \sum_{\mathbf{b} \in \{0,1\}^d} X^{\mathbf{b}}(\mathbf{u} - \mathbf{v}) \,. \tag{2.16}$$

We will refer to the term $X^{\mathbf{b}}(\mathbf{u}-\mathbf{v})$, and its spectral support $\mathcal{A}^{\mathbf{b}} + \mathbf{v} \triangleq \{\mathbf{u}+\mathbf{v} : \mathbf{u} \in \mathcal{A}^{\mathbf{b}}\}$, as the replica of atom $X^{\mathbf{b}}(\mathbf{u})$, respectively, $\mathcal{A}^{\mathbf{b}}$, at site $\mathbf{v}$. Using this terminology, one sees that reconstructing atom $X^{\mathbf{s}}(\mathbf{u})$ from the sampled spectrum $X_{\mathbf{s}}(\mathbf{u})$ requires accounting for the potential aliasing, i.e., overlap, of the replicas of the various atoms of $X(\mathbf{u})$ on $X^{\mathbf{s}}(\mathbf{u})$. This requires knowing which replicas of each atom will alias, i.e. overlap, $\mathcal{A}^{\mathbf{s}}$. More specifically, since the algorithm will only apply to images whose spectral support is limited to some subset of the Manhattan atoms, for any pair of bi-step vectors $\mathbf{b}$ and $\mathbf{b}'$, we will need to

know whether sampling with bi-step lattice $L_{\mathbf{s}}$ causes a replica of atom $\mathcal{A}^{\mathbf{b}'}$ (at some site $\mathbf{v} \in L_{\mathbf{s}}^*$) to overlap atom $\mathcal{A}^{\mathbf{b}}$ of the original spectrum.

Such overlap questions are answered by the following lemma and its corollary. Let $R_{\mathbf{s}}^{\mathbf{b}'}$ denote the union of the replicas of all $\mathbf{b}'$ atoms induced by sampling with $L_{\mathbf{s}}$. That is,

$$R_{\mathbf{s}}^{\mathbf{b}'} \triangleq \bigcup_{\mathbf{v} \in L_{\mathbf{s}}^* - \mathbf{o}} \left[\mathcal{A}^{\mathbf{b}'} + \mathbf{v}\right].$$

Lemma 8. *Consider sampling with $L_{\mathbf{s}}$.*

(a) *For all $\mathbf{b}, \mathbf{b}' \subset \mathbf{s}$, no replica of $\mathcal{A}^{\mathbf{b}'}$ overlaps $\mathcal{A}^{\mathbf{b}}$, i.e., $R_{\mathbf{s}}^{\mathbf{b}'} \cap \mathcal{A}^{\mathbf{b}} = \emptyset$.*

(b) *The replicas of $\mathcal{A}^{\mathbf{b}'}$ induced by sampling with $L_{\mathbf{s}}$ do not overlap $\mathcal{A}^{\mathbf{b}}$ if there exists at least one dimension i such that $s_i = 1$ and $b_i \neq b'_i$. That is, $R_{\mathbf{s}}^{\mathbf{b}'} \cap \mathcal{A}^{\mathbf{b}} = \emptyset$ if $(\mathbf{b} \oplus \mathbf{b}') \wedge \mathbf{s} \neq \mathbf{0}$, where $\mathbf{b} \oplus \mathbf{b}'$ denotes element-wise exclusive or (XOR).*

Proof:

(a) If $\mathbf{b}, \mathbf{b}' \subset \mathbf{s}$, then Fact 7(b) shows $\mathcal{A}^{\mathbf{b}}, \mathcal{A}^{\mathbf{b}'} \subset \mathcal{N}_{\mathbf{s}}$. Since the sampling theorem for conventional rectangular lattice sampling shows that replicas of $\mathcal{N}_{\mathbf{s}}$ do not overlap $\mathcal{N}_{\mathbf{s}}$, it follows that the replicas of $\mathcal{A}^{\mathbf{b}'}$ cannot overlap $\mathcal{A}^{\mathbf{b}}$.

(b) Let us compare the M-atom

$$\mathcal{A}^{\mathbf{b}} = a_1^{\mathbf{b}} \times \cdots \times a_d^{\mathbf{b}}$$

to an arbitrary replica in $R_{\mathbf{s}}^{\mathbf{b}'}$:

$$\mathcal{A}^{\mathbf{b}'} + \mathbf{v} = (a_1^{\mathbf{b}'} + v_1) \times \cdots \times (a_d^{\mathbf{b}'} + v_d),$$

for $\mathbf{v} \in L_{\mathbf{s}}^* - \{\mathbf{0}\}$. Note that $\mathcal{A}^{\mathbf{b}}$ and $\mathcal{A}^{\mathbf{b}'} + \mathbf{v}$ are disjoint if and only if $a_i^{\mathbf{b}} \cap (a_i^{\mathbf{b}'} + v_i) = \emptyset$, for some i.

If, as hypothesized in the lemma, $(\mathbf{b} \oplus \mathbf{b}') \wedge \mathbf{s} \neq \mathbf{0}$. Then there must exist i such that $s_i = 1$ and either $b_i = 1$, $b'_i = 0$ or $b_i = 0$, $b'_i = 1$. First, consider the case that $b_i = 1$, $b'_i = 0$. Then

$$a_i^{\mathbf{b}} = \left\{u_i : \frac{1}{2k_i\lambda_i} \leq |u_i| < \frac{1}{2\lambda_i}\right\}$$

and

$$a_i^{\mathbf{b}'} + v_i \;=\; \left\{u_i : |u_i| < \frac{1}{2k_i\lambda_i}\right\} + v_i\,.$$

Since $s_i = 1$, we have $v_i = \frac{n_i}{\lambda_i}$ for some n_i, and one sees from the above that no matter the value of n_i, $(a_i^{\mathbf{b}'} + v_i) \cap a_i^{\mathbf{b}} = \emptyset$. Hence, $(\mathcal{A}^{\mathbf{b}'} + \mathbf{v}) \cap \mathcal{A}^{\mathbf{b}} = \emptyset$, and so $R_{\mathbf{s}}^{\mathbf{b}'} \cap \mathcal{A}^{\mathbf{b}} = \emptyset$. A similar argument applies for the case that $b_i = 0$, $b_i' = 1$. □

The following will provide a key step in showing how to reconstruct appropriately bandlimited images.

Corollary 9. *If $\|\mathbf{b}'\| \le \|\mathbf{s}\|$, then replicas in $R_{\mathbf{s}}^{\mathbf{b}'}$ do not overlap $\mathcal{A}^{\mathbf{s}}$.*

Proof: We will apply Lemma 8 with $\mathbf{b} = \mathbf{s}$. If $\mathbf{b}' = \mathbf{s}$, then Part (a) of Lemma 8 shows $R_{\mathbf{s}}^{\mathbf{b}'} \cap \mathcal{A}^{\mathbf{s}} = \emptyset$. If $\mathbf{b}' \neq \mathbf{s}$ and $\|\mathbf{b}'\| \le \|\mathbf{s}\|$, then there must exist i such that $s_i = 1$ and $b_i' = 0$. Therefore, $(\mathbf{s} \oplus \mathbf{b}') \wedge \mathbf{s} \neq \mathbf{0}$, and Part (b) of Lemma 8 shows $R_{\mathbf{s}}^{\mathbf{b}'} \cap \mathcal{A}^{\mathbf{s}} = \emptyset$. □

2.4.8 The multidimensional Manhattan sampling theorem

Given a Manhattan set $M(B)$, consider its *Manhattan region*, which is defined to be the union of the Nyquist regions of the bi-step lattices of which it is the union:

$$\mathcal{M}(B) \;\triangleq\; \bigcup_{\mathbf{b}\in B} \mathcal{N}_{\mathbf{b}} \;=\; \bigcup_{\mathbf{b}\in B}\; \bigcup_{\mathbf{b}'\subset\mathbf{b}} \mathcal{A}^{\mathbf{b}'} \;=\; \bigcup_{\mathbf{b}'\in\overline{B}} A^{\mathbf{b}'},$$

where the second equality uses Fact 7(a). In this section we show that images bandlimited to $\mathcal{M}(B)$ can be recovered from their M-samples in $M(B)$; we give an explicit procedure for reconstructing such images from their samples in $M(B)$; and we show that the set of such bandlimited images is maximal in the Landau sense.

The key steps are the next two lemmas. The first shows that for any image $x(\mathbf{t})$ whose spectrum $X(\mathbf{u})$ is bandlimited to $\mathcal{M}(B)$, the portion of $X(\mathbf{u})$ in any highest weight M-atom $\mathcal{A}^{\mathbf{b}}$ can be easily recovered from the samples in $L_{\mathbf{b}}$, which are a subset of the M-samples. Specifically, $X^{\mathbf{b}}(\mathbf{u})$ can be recovered simply by extracting the $\mathcal{A}^{\mathbf{b}}$ portion of the sampled spectrum $X_{\mathbf{b}}(\mathbf{u})$ due to sampling with $L_{\mathbf{b}}$. Equivalently, the corresponding component $x^{\mathbf{b}}(\mathbf{t})$ of $x(\mathbf{t})$ can be recovered by filtering the sampled image $x_{\mathbf{b}}(\mathbf{t})$ with an ideal bandpass filter with frequency support $\mathcal{A}^{\mathbf{b}}$.

Lemma 10. *Suppose $M(B)$ is a Manhattan set and $x(\mathbf{t})$ is an image whose spectrum $X(\mathbf{u})$ is bandlimited to $\mathcal{M}(B)$. Then if $\mathbf{b}$ has maximal weight in $\overline{B}$,*

$$X^{\mathbf{b}}(\mathbf{u}) \;=\; H^{\mathbf{b}}(\mathbf{u})\, X_{\mathbf{b}}(\mathbf{u})\,, \tag{2.17}$$

where $H^{\mathbf{b}}(\mathbf{u}) = 1$ for $\mathbf{u} \in A^{\mathbf{b}}$, and 0 otherwise.

Proof: Consider any $\mathbf{b}$ of maximal weight. Since, according to (2.4), $X_{\mathbf{b}}(\mathbf{u})$ consists of replicas of $X(\mathbf{u})$ at the frequencies in $L_{\mathbf{b}}^*$, since $X(\mathbf{u})$ can be decomposed into its components on Nyquist atoms $\{\mathcal{A}^{\mathbf{b}'} : \mathbf{b}' \in \{0,1\}^d\}$, and since $X(\mathbf{u})$ is bandlimited to $\mathcal{M}(B) = \bigcup_{\mathbf{b}' \in \overline{B}} \mathcal{A}^{\mathbf{b}'}$, it suffices to argue that for all $\mathbf{b}' \in \overline{B}$, no replica of $\mathcal{A}^{\mathbf{b}'}$ intersects $\mathcal{A}^{\mathbf{b}}$. First, Lemma 8(a) applied with $\mathbf{s} = \mathbf{b}' = \mathbf{b}$ shows that no replica of $\mathcal{A}^{\mathbf{b}}$ with $\mathbf{v} \neq \mathbf{0}$ intersects $\mathcal{A}^{\mathbf{b}}$. Second, Corollary 9 and the fact that $\mathbf{b}$ has maximal weight in $\overline{B}$ imply that for any other $\mathbf{b}' \in \overline{B}$, no replica of $\mathcal{A}^{\mathbf{b}'}$ can overlap $\mathcal{A}^{\mathbf{b}}$. □

Once $X(\mathbf{u})$ has been recovered in all such highest weight (highest frequency) M-atoms, the next lemma shows that $X(\mathbf{u})$ can then be recovered in the next highest weight M-atoms by canceling the contributions due to atoms with larger weight. In effect, the aliasing of one atom comes only from atoms with larger weight, i.e., higher frequency. Specifically, for any such $\mathbf{b}$, it shows that $X^{\mathbf{b}}(\mathbf{u})$ can be recovered from the spectrum $X_{\mathbf{b}}(\mathbf{u})$ due to sampling with $L_{\mathbf{b}}$ simply by first subtracting each replica $X^{\mathbf{b}'}(\mathbf{u} - \mathbf{v})$, $\mathbf{v} \in L_{\mathbf{b}}^*$, of every M-atom $\mathbf{b}'$ with $\|\mathbf{b}'\| > \|\mathbf{b}\|$, and then extracting the $\mathcal{A}^{\mathbf{b}}$ portion of the resulting "de-aliased" spectrum.

Lemma 11. *Suppose $M(B)$ is a Manhattan set and $x(\mathbf{t})$ is an image whose spectrum $X(\mathbf{u})$ is bandlimited to $\mathcal{M}(B)$. If $X^{\mathbf{b}'}(\mathbf{u})$ is known for all $\mathbf{b}'$ larger than $\mathbf{b}$, then*

$$X^{\mathbf{b}}(\mathbf{u}) = H^{\mathbf{b}}(\mathbf{u})\Big[X_{\mathbf{b}}(\mathbf{u}) \;-\; \sum_{\mathbf{b}':\, \|\mathbf{b}'\| > \|\mathbf{b}\|} X_{\mathbf{b}}^{\mathbf{b}'}(\mathbf{u})\Big]\,, \tag{2.18}$$

where $H^{\mathbf{b}}(\mathbf{u})$ is defined in the previous lemma,

Proof:

$$\begin{aligned}
\text{RHS of (2.18)} &= H^{\mathbf{b}}(\mathbf{u})\Big[\sum_{\mathbf{v} \in L_{\mathbf{b}}^*} X(\mathbf{u} - \mathbf{v}) - \sum_{\mathbf{b}':\, \|\mathbf{b}'\| > \|\mathbf{b}\|} \sum_{\mathbf{v} \in L_{\mathbf{b}}^*} X^{\mathbf{b}'}(\mathbf{u} - \mathbf{v})\Big] \qquad (2.19)\\
&= H^{\mathbf{b}}(\mathbf{u})\Big[\sum_{\mathbf{b}':\, \|\mathbf{b}'\| \leq \|\mathbf{b}\|} \sum_{\mathbf{v} \in L_{\mathbf{b}}^*} X^{\mathbf{b}'}(\mathbf{u} - \mathbf{v})\Big]\\
&= X^{\mathbf{b}}(\mathbf{u})\,,
\end{aligned}$$

where the first equality uses (2.4), the second uses (2.14), and the last derives from Lemma 8 and its corollary. In particular, for the $\mathbf{b}' = \mathbf{b}$ term in the above sum, Part (a) of Lemma 8 applied with $\mathbf{s} = \mathbf{b}' = \mathbf{b}$ shows that all replicas of $\mathcal{A}^{\mathbf{b}}$ (with $\mathbf{v} \neq \mathbf{0}$) do not overlap $\mathcal{A}^{\mathbf{b}}$ and, consequently, are eliminated by the filter $H^{\mathbf{b}}(\mathbf{u})$. Corollary 9 implies every replica of $\mathcal{A}^{\mathbf{b}'}$ (with $\mathbf{v} \neq \mathbf{0}$) does not overlap $\mathcal{A}^{\mathbf{b}}$ and, consequently, is again eliminated by the filter. Since also $\mathcal{A}^{\mathbf{b}'}$ does not overlap $\mathcal{A}^{\mathbf{b}}$, the only term in the sum not eliminated by the filter is $X^{\mathbf{b}}(\mathbf{u})$, which establishes (2.18). □

Note that the sum in (2.18) can be limited to $\mathbf{b}' \in \overline{B}$. Note also that Lemma 11 implies Lemma 10, because when $\mathbf{b}$ is a largest weight bi-step vector in $\overline{B}$, the summation term in (2.18) is zero, and so (2.18) reduces to (2.17).

An alternative way to write (2.18) is

$$X^{\mathbf{b}}(\mathbf{u}) = H^{\mathbf{b}}(\mathbf{u})\Big[X_{\mathbf{b}}(\mathbf{u}) - \sum_{\mathbf{b}':\, \|\mathbf{b}'\| > \|\mathbf{b}\|} \sum_{\mathbf{n} \in C_{\mathbf{b}}} X^{\mathbf{b}'}(\mathbf{u} - \mathbf{n} \odot \boldsymbol{\beta}_{\mathbf{b}})\Big] \tag{2.20}$$

where $\boldsymbol{\beta}_{\mathbf{b}} = (\beta_{\mathbf{b},1}, \ldots, \beta_{\mathbf{b},d})$, with

$$\beta_{\mathbf{b},i} \triangleq \begin{cases} \frac{1}{\lambda_i}, & \text{if } b_i = 1 \\ \frac{1}{k_i \lambda_i}, & \text{if } b_i = 0 \end{cases}$$

and

$$C_{\mathbf{b}} \triangleq \big\{\mathbf{n} \in \mathbb{Z}^d : n_i = 0 \text{ for } i \text{ s.t. } b_i = 1, \text{ and} |n_i| \leq k_i - 1 \text{ for } i \text{ s.t. } b_i = 0\big\}.$$

To demonstrate (2.20), we note that since all atoms of the Manhattan partition are contained in $\mathcal{N}_{\mathbf{1}}$, one can eliminate from the last sum in (2.19) any $\mathbf{v}$ such that $(\mathcal{N}_{\mathbf{1}} + \mathbf{v}) \cap \mathcal{N}_{\mathbf{1}} = \emptyset$. This leads to limiting the sum to $\mathbf{v}$ such that $|v_i| < \frac{1}{\lambda_i}$ for each i. Taking into account what $\mathbf{v}$'s are in $L^*_{\mathbf{b}}$ leads to (2.20). For the usual 2D case, in which $B = \{(1,0),(0,1)\}$, (2.20) gives a different reconstruction formula than in Section 2.3 for the spectrum in the coarse Nyquist region, $X^C(\mathbf{u}) = X^{(0,0)}(\mathbf{u})$ (see (2.5)-(2.7)). Specifically, it subtracts terms involving both $X^V(\mathbf{u}) = X^{(0,1)}(\mathbf{u})$ and $X^H(\mathbf{u}) = X^{(1,0)}(\mathbf{u})$ from $X_C(\mathbf{u})$, whereas the formula in Section 2.3 subtracts terms involving $X^H(\mathbf{u})$ from $X_V(\mathbf{u})$. Moreover, the summation over $\mathbf{n}$ in (2.20) sums over approximately twice as many values of $\mathbf{v}$. This is because it conservatively includes all $\mathbf{v} \in L^*_{\mathbf{b}}$ such that $\mathcal{N}_1 + \mathbf{v} \cap \mathcal{N}_1 \neq \emptyset$, whereas the formula in Section 2.3 includes only $\mathbf{v}$'s such that $\mathcal{N}_V + \mathbf{v} \cap \mathcal{N}_C \neq \emptyset$. If desired $C_{\mathbf{b}}$, in (2.20) could be replaced by a smaller set $C_{\mathbf{b},\mathbf{b}'}$ that depends on $\mathbf{b}'$ as well as $\mathbf{b}$.

The basic idea behind following theorem, which is the main result of this section, is that

the process of finding $X^{\mathbf{b}}(\mathbf{u})$ for smaller and smaller weight $\mathbf{b}$'s can continue until $X^{\mathbf{0}}$, the spectrum in $\mathcal{A}^{\mathbf{0}}(\mathbf{u}) = \mathcal{N}_{\mathbf{0}}$, is found, and all of $X(\mathbf{u})$ is known. As a result, $x(\mathbf{t})$ will also be known.

Theorem 12. Multidimensional Manhattan Sampling Theorem. *Suppose we sample an image $x(\mathbf{t})$ with Manhattan set $M(B)$. If the image spectrum $X(\mathbf{u})$ is bandlimited to $\mathcal{M}(B)$, then for each $\mathbf{b} \in \overline{B}$, $X^{\mathbf{b}}(\mathbf{u})$ can be exactly recovered from the samples with the following "onion-peeling" approach — apply Lemma 10 for the largest $\mathbf{b}$'s in $\overline{B}$, and then repeatedly apply Lemma 11 for the next largest $\mathbf{b}$'s. Then, $x(\mathbf{t})$ can be exactly recovered from*

$$x(\mathbf{t}) \;=\; \mathcal{F}^{-1}\Big\{ \sum_{\mathbf{b}\in\overline{B}} X^{\mathbf{b}}(\mathbf{u}) \Big\}.$$

Proof: From (2.14) and the bandlimitation of $X(\mathbf{u})$, it is clear that $X(\mathbf{u})$ can be recovered if $X^{\mathbf{b}}(\mathbf{u})$ is recovered for each $\mathbf{b} \in \overline{B}$. First, $X^{\mathbf{b}}(\mathbf{u})$ can be recovered via Lemma 10 for the largest $\mathbf{b}$'s in $\overline{B}$, which correspond to the highest frequency M-atoms. Next, repeatedly applying Lemma 11 enables one to recover $X^{\mathbf{b}}(\mathbf{u})$ for the Nyquist atoms corresponding to the largest of the remaining $\mathbf{b}$'s, until $X^{\mathbf{0}}(\mathbf{u})$, corresponding to the lowpass atom, is recovered. □

While this theorem indicates a frequency domain reconstruction, followed by an inverse transform, a direct spatial domain reconstruction is also possible. As we now delineate, this involves reconstructing each $x^{\mathbf{b}}(\mathbf{t})$, $\mathbf{b} \in \overline{B}$, and then using

$$x(\mathbf{t}) \;=\; \sum_{\mathbf{b}\in\overline{B}} x^{\mathbf{b}}(\mathbf{t}).$$

Taking the inverse transform of (2.20) yields

$$\begin{aligned} x^{\mathbf{b}}(\mathbf{t}) \;&=\; h^{\mathbf{b}}(\mathbf{t}) \star \Big[x_{\mathbf{b}}(\mathbf{t}) \;-\; \sum_{\|\mathbf{b}'\|>\|\mathbf{b}\|} x_{\mathbf{b}}^{\mathbf{b}'}(\mathbf{t}) \Big] \\ &=\; h^{\mathbf{b}}(\mathbf{t}) \star \Big[K(\mathbf{b}) \sum_{\mathbf{t}'\in L_{\mathbf{b}}} \delta(\mathbf{t}-\mathbf{t}') \Big(x(\mathbf{t}) - \sum_{\|\mathbf{b}'\|>\|\mathbf{b}\|} x^{\mathbf{b}'}(\mathbf{t}) \Big) \Big] \\ &=\; K_{\mathbf{b}} \sum_{\mathbf{t}'\in L_{\mathbf{b}}} \Big(x(\mathbf{t}') \;-\; \sum_{\|\mathbf{b}'\|>\|\mathbf{b}\|} x^{\mathbf{b}'}(\mathbf{t}') \Big) \, h^{\mathbf{b}}(\mathbf{t}-\mathbf{t}'), \end{aligned}$$

where $h^{\mathbf{b}}(\mathbf{t}) = \mathcal{F}^{-1}\big\{H^{\mathbf{b}}(\mathbf{u})\big\}$, $\star$ denotes convolution, and $K_{\mathbf{b}} = \prod_{i:b_i=1} \lambda_i \times \prod_{i:b_i=0} k_i\lambda_i$. This shows how $x^{\mathbf{b}}(\mathbf{t})$ can be found — first for the largest $\mathbf{b}$'s from samples of $x(\mathbf{t})$ taken on $L_{\mathbf{b}}$,

then for the next largest $\mathbf{b}$'s from samples of $x(\mathbf{t})$ taken on $L_{\mathbf{b}}$, as well as samples of $x^{\mathbf{b}'}(\mathbf{t})$ taken on $L_{\mathbf{b}}$ for all larger $\mathbf{b}'$, and so on. It remains to find a formula for $h^{\mathbf{b}}(\mathbf{t})$.

To find a formula for $h^{\mathbf{b}}(\mathbf{t})$, which is the inverse transform of $H^{\mathbf{b}}(\mathbf{u})$, which in turn has support $\mathcal{A}^{\mathbf{b}}$, we begin by recalling that $\mathcal{A}^{\mathbf{b}}$ is the union of $2^{\|\mathbf{b}\|}$ orthotopes in frequency space. Along each dimension i, these orthotopes are centered at zero if $b_i = 0$, and at $\pm c_i$ if $b_i = 1$, where

$$c_i \triangleq \frac{1}{2}\Big(\frac{1}{2\lambda_i} + \frac{1}{2k_i\lambda_i}\Big).$$

Additionally, along the ith dimension, these orthotopes have length

$$w_i(\mathbf{b}) \triangleq \begin{cases} \frac{1}{2\lambda_i} - \frac{1}{2k_i\lambda_i}, & b_i = 1 \\ \frac{1}{k_i\lambda_i}, & b_i = 0. \end{cases}.$$

Using these quantities, we can write the filter $H^{\mathbf{b}}(\mathbf{u})$ as

$$H^{\mathbf{b}}(\mathbf{u}) = \Bigg[\prod_{i=1}^{d} \mathrm{rect}\Big(\frac{u_i}{w_i(\mathbf{b})}\Big)\Bigg] \star \Bigg[\prod_{i:b_i=0} \delta(u_i) \prod_{i:b_i=1} \Big[\delta(u_i - c_i) + \delta(u_i + c_i)\Big]\Bigg],$$

where $\mathrm{rect}(x) \triangleq 1$ for $|x| < \frac{1}{2}$, and 0 otherwise. Note that the first term is an orthotope centered at $\mathbf{0}$, and the convolution with delta functions shifts the orthotopes along all dimensions i such that $b_i = 1$. Taking the inverse transform yields

$$h_{\mathbf{b}}(\mathbf{t}) = \prod_{i=1}^{d} w_i(\mathbf{b}) \,\mathrm{sinc}(w_i(\mathbf{b})t_i) \cdot \prod_{i:b_i=1} 2\cos(2\pi c_i t_i),$$

where $\mathrm{sinc}(t) \triangleq \frac{\sin \pi t}{\pi t}$. Observe that, as mentioned in the introduction, these impulse responses depend on the k_i's and λ_i's, but not the choice of bi-step lattices that comprise the Manhattan set. Moreover, the λ_i's have only a simple spatial scaling effect on the filters.

2.4.9 Achievement of Landau lower bound on sampling density

We now show that the volume of $\mathcal{M}(B)$, denoted $|\mathcal{M}(B)|$, equals the sampling density of $M(B)$. As a result, the set of images bandlimited to $\mathcal{M}(B)$ is a maximal set of images that are reconstructable from sample set $M(B)$. Equivalently, $M(B)$ has the smallest density of any sampling set such that all images bandlimited to $\mathcal{M}(B)$ are reconstructable.

Since the M-atoms partition $\mathcal{M}(B)$, we can calculate $|\mathcal{M}(B)|$ simply by summing over

the volumes of the M-atoms $\mathcal{A}^{\mathbf{b}}$ for $\mathbf{b} \in \overline{B}$:

$$|\mathcal{M}(B)| = \sum_{\mathbf{b}\in\overline{B}} |\mathcal{A}^{\mathbf{b}}| = \sum_{\mathbf{b}\in\overline{B}} \Big(\prod_{i:b_i=1} 2\Big(\frac{1}{2\lambda_i} - \frac{1}{2k_i\lambda_i}\Big) \prod_{i:b_i=0} \frac{1}{k_i\lambda_i} \Big)$$
$$= \frac{\sum_{\mathbf{b}\in\overline{B}} \prod_{i:b_i=1}(k_i - 1)}{\prod_{i=1}^{d} k_i\lambda_i}.$$

Comparing the above to (2.12), we see that $|\mathcal{M}(B)|$ equals the sampling density $\rho(B)$.

2.4.10 Discrete-space images

The d-dimensional Manhattan sampling theorem and reconstruction procedures can be straightforwardly extended to discrete-space images in d dimensions in the same fashion as for two dimensions. For example, for infinite-support images, frequencies need to be scaled by 2π, and for finite-support images, each spatial resolution T_i must be a multiple of $k_i\lambda_i$ and the Manhattan atoms $\widetilde{A}_{\mathbf{b}}$ need to be redefined to be consistent with the DFT, as was done for the discrete Nyquist region $\widetilde{\mathcal{N}}_{\alpha_1,\alpha_2}$. Here, we simply give the main step of the frequency-space onion-peeling reconstruction algorithm for reconstructing a Manhattan bandlimited discrete-space image $x[\mathbf{t}]$ with finite support sampled with Manhattan set $M(B)$:

$$X^{\mathbf{b}}[\mathbf{t}] \;=\; \widetilde{H}^{\mathbf{b}}[\mathbf{u}] \;\mathrm{DFT}\Big\{ x_{\mathbf{b}}[\mathbf{t}] \;-\; \sum_{\mathbf{b}':\|\mathbf{b}'\|>\|\mathbf{b}\|} x_{\mathbf{b}}^{\mathbf{b}'}[\mathbf{t}] \Big\},$$

where $x_b[\mathbf{t}]$ and $x_{\mathbf{b}}^{\mathbf{b}'}[\mathbf{t}]$ denote, respectively, the $L_{\mathbf{b}}$ subsamplings of the Manhattan samples (scaled by $K_{\mathbf{b}}$), and the previously reconstructed atom $x^{\mathbf{b}'}[\mathbf{t}]$, and $\widetilde{H}^{\mathbf{b}}[\mathbf{u}]$ denotes an ideal bandpass filter for atom $\widetilde{\mathcal{A}}_{\mathbf{b}}$.

2.5 Concluding Remarks

In two dimensions, this chapter has shown that from samples of a Manhattan set one can perfectly reconstruct any image that is bandlimited to the union of the Nyquist regions of the horizontal and vertical rectangular lattices comprising the Manhattan set. It also prescribed a straightforward linear reconstruction procedure, for continuous- and discrete-space images.

For three and higher dimensions, this chapter has identified Manhattan sets as the union of a finite number of bi-step rectangular lattices, with the result that many Manhattan geometries are possible. It introduced an efficient binary-vector representation of bi-step

lattices, and consequently Manhattan sets, which enabled the specification of a partition of the dense rectangular lattice into collections of cosets of the coarse lattice. This, in turn, enabled the density of a Manhattan to be computed. The representation of bi-step lattices also enabled a partition of the Nyquist region of the dense rectangular lattice, which in turn enabled a precise analysis of the aliasing, i.e., spectral overlaps, by the atoms of any particular type in the spectral replicas induced by any particular bi-step lattice subsampling. With this, it was shown that images bandlimited to the union of the Nyquist regions of the bi-step lattices comprising the Manhattan set can be perfectly reconstructed using an an efficient closed-form onion-peeling type reconstruction algorithm that reconstructs the image spectrum working from higher to lower frequency atoms of the partition. At each step, the algorithm works with samples of one particular bi-step lattice (of the Manhattan set), and obtains the spectrum of the image in the corresponding atom of the frequency partition by subtracting contributions due to aliasing of previously determined atoms of the partition. Both frequency- and time-domain versions of the algorithm were given. It was also shown that the set of Manhattan bandlimited images is maximal in the Landau sense. To the best of our knowledge, this is the first demonstration that images bandlimited to the union of Nyquist regions can be recovered from the union of the corresponding lattices.

There are several avenues for future research. One could seek to extend the results to continuous-space images whose spectra contain delta functions, e.g. periodic images. Second, instead of a recursive onion-peeling reconstruction, one could seek direct closed-form linear reconstructions, as in [27], which might be useful for implementations, though they might have less intuitive appeal. This is not difficult in 2D, but is more challenging in higher dimensions. Finally, whereas M-sampling can be viewed as sampling (in various ways) on the boundaries of a rectangular (hyper-rectangular) lattice tessellation, one could seek sampling theorems and reconstruction procedures for images sampled on the boundaries of other lattice tessellations, such as a hexagonal tessellation.

CHAPTER 3

Manhattan Image Reconstruction

In Chapter 2, we provided a method for perfectly reconstructing an image from its Manhattan samples, provided that the image satisfied certain bandlimitation conditions. However, most natural images are not bandlimited, and perfect reconstruction cannot be guaranteed. This motivates the need for general algorithms that reconstruct non-bandlimited images from their Manhattan-grid samples with as little error as possible.

In Section 3.1, we discuss previous work on such algorithms, which includes a method called the Cutset-MRF method [1]. The remaining three sections propose three new algorithms which improvement on this method.

Section 3.2 describes the "Piecewise Planar" method that contains two main contributions: the *K-planes algorithm* and the *interior labeling algorithm*. The K-planes algorithm is a generalization of the K-means algorithm, and is used to segment Manhattan-grid samples into piecewise-planar regions. The interior labeling algorithm is used to extend these segmentations into the interior of the Manhattan blocks. The final estimation step of the Piecewise Planar method is similar to the final estimation step of the Cutset MRF method, but it uses the piecewise-planar approximations of each image segment as the segment mean, instead of using a constant value across each segment.

In Section 3.3, the Orthogonal Gradient (OG) Algorithm is presented, which is an algorithm that alternates between minimizing a convex optimization problem, and updating the parameters of that optimization problem based on the gradient of the previous image. Specifically, parameters are chosen in order to penalize adjacent pixel differences most heavily in directions orthogonal to the spatial gradient of the previous image. One downside of the OG method is that it only considers the direction of the gradient when calculating these parameters, which leads to some unwanted artifacts in smooth regions of the image. These artifacts are reduced by the Locally Orthogonal Orientation Penalization (LOOP) Algorithm which is presented in Section 3.4. This is also an alternating algorithm that calculates parameters based on the previous image gradients, but it considers both gradient magnitudes

and directions when calculating parameters. In particular, it calculates a local singular value decomposition (SVD) of neighboring gradient vectors at each pixel. The resulting singular values and singular vectors are then used to carefully choose tradeoff parameters in both the isotropic and anisotropic terms of the cost function of the convex optimization problem. Finally, Section 3.5 compares the OG and LOOP algorithm to two competing interpolation/inpainting algorithms for non-lattice sampled data. Overall, we find that the LOOP algorithm outperforms all other methods both qualitatively, as well as in mean-squared error measurements. Furthermore, we also apply the LOOP algorithm to the task of lattice interpolation; we find that the LOOP algorithm outperforms bicubic interpolation, and is competitive with a recent interpolation algorithm [11].

We note to the reader that the first two presented methods (Cutset MRF and Piecewise Planar) are much more similar to each other than the last two presented methods (the OG and LOOP algorithms). The Cutset MRF and Piecewise-Planar Methods should be viewed as exploratory methods, whereas the OG and LOOP algorithms should be viewed as the core contributions to this chapter.

We mention that the Cutset-MRF method was originally presented at ICIP 2011 [1], the piecewise-planar method was presented at ICIP 2012 [13], and the OG algorithm was presented at ICIP 2014 [18].

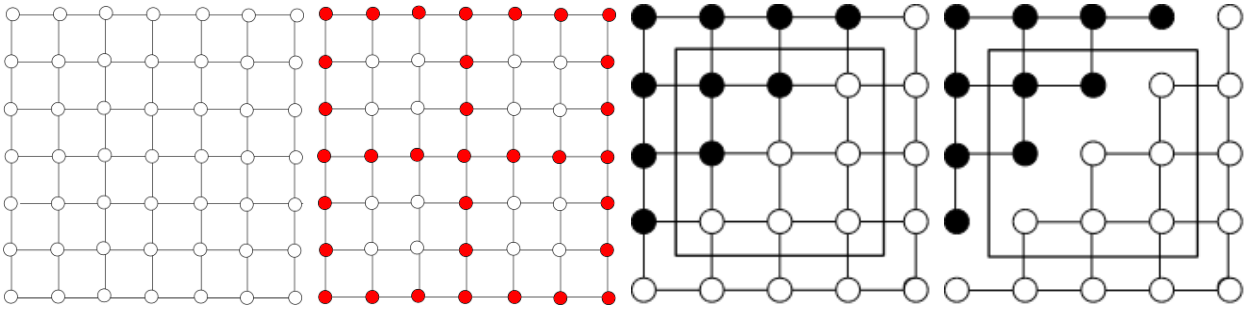
3.1 Background: Cutset-MRF Reconstruction Method

We begin by describing the notation to be used in this section and the next. We will then formally define our problem of image reconstruction from Manhattan samples.

3.1.1 Definitions and Notation

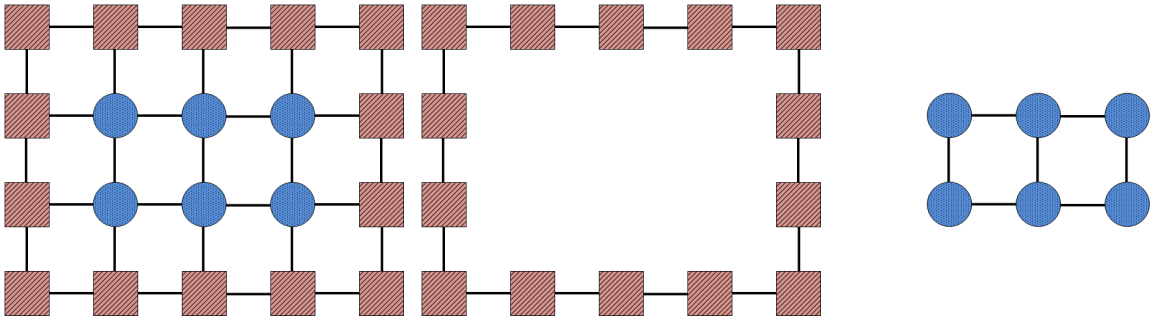
The following notation is used in this section as well as in Section 3.2. Let $\mathbf{f} = \{f_i : i \in I_{global}\}$ be the matrix of pixel intensities in a finite discrete image, where the index set I_{global} indexes the image pixels. We define a graph on the entire image $G_{global} = (I_{global}, E_{global})$, where the arc[1] set E_{global} is formed using a 4-point neighborhood (Figures 3.1(a)). Let $C \subset I_{global}$ be an $M \times N$ Manhattan-grid cutset (red nodes in Figure 3.1(b)). It should be clear that the removal of all nodes in C from the graph separates the graph into non-connected *blocks*. Thus, suppose we sample $\mathbf{f}$ on an $M \times N$ Manhattan grid cutset C by recording every Mth row and Nth column of pixel intensities of $\mathbf{f}$. Since we have sampled the image on a cutset

[1] An *arc* is more commonly known as an *edge*. However, in this document, we will use *edge* to refer to an *image* edge and *arc* to refer to a *graphical* edge.



(a) 4-point graph (b) 3×3 Manh. Cutset (c) Segmented block (d) Modified graph

Figure 3.1: Cutsets



(a) Set of nodes indexed by $I = \overline{I} \cup \widetilde{I}$ (b) Set of border nodes indexed by $\overline{I}$ (c) Set of interior nodes indexed by $\widetilde{I}$

Figure 3.2: Index sets for the graph of a 3×4 block.

with respect to graph G_{global}, we have separated the image into $(M-1) \times (N-1)$ chunks of unknown pixels. The union of an unknown pixel chunk with its known boundary pixels will be called a *block*, and each block is of size $(M+1) \times (N+1)$ pixels.

It will be useful to define some block-wise notation. Let $\mathbf{x} = \{x_i : i \in I\}$ be the matrix of pixel intensities in a block, where I indexes the $(M+1)(N+1)$ block pixels. We define a graph on an image block $G = (I, E)$, where again the arc set E is a 4-point neighborhood. Two pixels i and j are said to be *neighbors* if the arc (i, j) is contained in E. We let ∂i and ∂x_i denote the set of pixels and pixel intensities neighboring i, respectively. We denote the border and interior pixel intensities by $\overline{\mathbf{x}} = \{x_i : i \in \overline{I}\}$ and $\widetilde{\mathbf{x}} = \{x_i : i \in \widetilde{I}\}$, respectively, where $\overline{I}$ and $\widetilde{I}$ are the corresponding index sets, as shown in Figure 3.2. For use in Section 3.2, let $\mathbf{u}_i = [u_{i1}, u_{i2}]^T$ denote the 2D coordinates of the ith pixel in $\mathbb{R}^2$.

A key step in both the Cutset MRF method and Piecewise-Planar method involves further separating the underlying graph G of each image block into non-connected regions, with goal of obtaining a final graph G' so that no significant image edges pass through any connected

component of the graph. This is done in two steps. First, in the *segmentation step*, each pixel $i \in I$ is *labeled* with an integer. Second, in the *edge removal step*, edges in the graph G are removed wherever neighboring pixels have different labels. Specifically, let $\mathbf{v} = \{v_i : i \in I\}$ denote a block *segmentation*, where each v_i can take on an integer value[2]. Let ∂v_i denote the labels of the pixels adjacent to v_i. When pixels i and j are adjacent pixels with different labels, i.e., $v_i \neq v_j$, we say that the arc (i, j) is an *odd bond*. The arc set E is then modified by removing all odd bonds to obtain a new graph G'. After arc removal, a group of connected pixels in G' with the same label are together called a *segment*. Note that there can be multiple segments with the same label, and that pixels with the same label will not always be part of the same connected segment.

3.1.2 Problem Background and Algorithm Overview

Cutset sampling was first investigated as a method for lossy and lossless bilevel image compression [4–7]. Since then, [1] contains a first investigation into reconstructing grayscale images from their Manhattan grid samples. In this section, we will give a brief discussion of the Markov Random Field (MRF) based method from [1]. Section 3.2 will described the improved "Piecewise Planar" method.

Recall that a *Markov random field* (MRF) is a set of random variables $\mathbf{X} = \{X_i : i \in I_{global}\}$ defined on a graph $G_{global} = (I_{global}, E_{global})$ such that given the boundary ∂B of a set of nodes B, the random variables defined on B are conditionally independent of all other nodes. If $\mathbf{X}$ is modeled as an MRF on G_{global}, each block of image intensities $\mathbf{x}$ will be conditionally independent of one another given the intensities on the cutset. Thus, if we make an MRF assumption about $\mathbf{f}$, then any estimate of the block interior values can be obtained based only on the border of each block, thereby allowing each block to be processed separately. Furthermore, we would like to use another field to model image edges. Specifically, we would like to segment the image (or blocks of the image) such that no image edges pass through any resulting segment, and then remove graph arcs wherever there is an odd bond.

The preceding discussions suggest the following three-step algorithm:

1. Segment the entire cutset C so that no significant edges pass through any segment. From this segmentation of the entire Manhattan grid, one can obtain a border segmentation $\overline{\mathbf{v}}$ for each block.

[2]The Cutset MRF method of Section 3.1 uses a binary segmentation, whereas the Piecewise-Planar method of Section 3.2 will allow each v_i to take one of K different values.

2. For each block, segment the interior, producing $\widetilde{\mathbf{v}}$, based only on the border segmentation $\overline{\mathbf{v}}$.

3. For each block, estimate the interior pixel intensities $\widetilde{\mathbf{x}}$ using only the border pixel intensities $\overline{\mathbf{x}}$ and the entire block segmentation $\mathbf{v} = \overline{\mathbf{v}} \cup \widetilde{\mathbf{v}}$.

In general, the combined goal of steps 1 and 2 is to segment the image into smooth regions. In step 3, before estimation, the graph arcs E_{global} are modified by removing arcs between indices wherever there is an odd bond in the segmentation. As will be seen, these steps prevent separate regions from influencing each other during the interior estimation step, thereby ensuring the preservation of sharp edges. See Figure 3.3 for a visual representation of this three-step process. We will briefly describe each of the three steps in slightly more detail; the curious reader is directed to [1] for additional information.

(a) Original image (b) Segmented cutset (c) Complete segm. (d) Reconstruction

Figure 3.3: Original three-step honest algorithm found in [1]

3.1.2.1 Step 1: Cutset segmentation

The goal of this step is to segment the Manhattan grid cutset so that no significant image edges pass through any resulting segments. In order to do so, we use the the adaptive clustering algorithm (ACA) [39], which is a generalization of K-means clustering. It is a general-purpose iterative algorithm that segments an image into K regions by adapting to local pixel intensities. In Step 1 of the algorithm, ACA with $K = 2$, i.e., binary segmentation, is applied only to the sampled values of the image on the Manhattan cutset; the result is a segmentation defined only on the Manhattan cutset. Thus, each element of of the block border label set $\overline{\mathbf{v}}$ is assigned either a 0 or a 1.

3.1.2.2 Step 2: Block interior segmentation

This is a block-wise step where the interior segmentation of a block $\widetilde{\mathbf{v}}$ is performed using the border segmentation $\overline{\mathbf{v}}$ obtained in Step 1. In particular, this is viewed as a MAP estimation

problem where $\mathbf{v}$ is a binary MRF, known as the Ising model [7,40]. It can be easily seen that a block MAP solution for this problem is the solution with minimum *odd bonds*, regardless of the choice of temperature parameter for the Ising model. The procedure for determining these solutions for the most common boundaries is found in [4, 7]. Thus, this step produces a full labeling $\mathbf{v} = \overline{\mathbf{v}} \cup \widetilde{\mathbf{v}}$.

3.1.2.3 Step 3: Block reconstruction

Recall that each block of pixel intensities $\mathbf{x}$ is defined on a graph $G = (I, E)$, where I indexes the pixels and E is an edge set defined by the 4-point neighborhood show in Figure 3.1(a). Given the full block segmentation produced by the previous step $\mathbf{v}$, we obtain the modified edge set E' by removing all edges in E corresponding to odd bonds in $\mathbf{v}$. This process is depicted in Figures 3.1(c) and 3.1(d). Removing these arcs will produce sharp edge transitions in the reconstruction by only allowing interior pixels to be estimated from border pixels with the same label.

We now model the block $\mathbf{x}$ as a Gauss Markov random field on the modified graph $G' = (I, E')$ having probability density

$$p(\mathbf{x}; \mathbf{v}) = \frac{1}{Z} \prod_{i \in I} \phi_i(x_i) \prod_{\{i,j\} \in E'} \psi_{i,j}(x_i, x_j) \tag{3.1}$$

where Z is a normalizing constant, ϕ_i and $\psi_{i,j}$ are the *node* and *edge potential functions*

$$\phi_i(x_i) = \exp\left\{ -\frac{1}{2} d(x_i - \mu_i)^2 \right\}$$

and

$$\psi_{i,j}(x_i, x_j) = \exp\left\{ -cd(x_i - \mu_i)(x_j - \mu_j) \right\}.$$

Note that the inverse covariance matrix R^{-1} corresponding to this Gaussian model has diagonal entries $R^{-1}_{ii} = d$, and off-diagonal entries $R^{-1}_{ij} = -cd$ or 0, when $\{i, j\} \in E'$ or not, respectively.

We assume that pixels sharing the same label v_i have the same mean. Thus, the mean of each pixel is estimated as the mean of the border pixels that share the same label. For example, if $v_j = 0$, then μ_j is determined by averaging all $\{x_i : i \in \overline{I},\, v_i = 0\}$.

We chose to estimate the interior pixels $\widetilde{\mathbf{x}}$ using the minimum mean-squared error (MMSE) estimate of $\widetilde{\mathbf{x}}$ given $\overline{\mathbf{x}}$, which is exactly the MAP estimate under the Gaussian assumption. Furthermore, the MAP estimation rule for the interior pixels $\widetilde{\mathbf{x}}$ given the the border pixels $\overline{\mathbf{x}}$ reduces to the usual linear MMSE method based on R^{-1} and μ.

When estimating $\widetilde{\mathbf{x}}$, we have a modeling choice of parameters d and c. However, the choice of d has no effect on the MAP estimation step, so we arbitrarily chose $d = 1$. We chose $c = 0.26$, which was empirically found to be the largest value of c such that the inverse covariance matrix R^{-1} of the resulting Gaussian MRF was positive definite.

Finally, we mention that the Gaussian model was one of several models tested in [1] for doing MMSE estimation, and there was another approach that was not based on MMSE/MAP estimation. We presented the Gaussian model here due to its similarity to the Piecewise-Planar method, which will be introduced in Section 3.2.

3.1.2.4 Results

The method presented here is compared to the new Piecewise Planar method at the end of Section 3.2. The anxious reader is encouraged to consider Figure 3.6 and Table 3.1 to see experimental results.

3.2 Piecewise-Planar Reconstruction Method

In the previous section, for each image block, we assumed that pixels sharing the same label v_i had the same constant mean value μ_j. In this section, we modify that assumption by attempting to model the segment means with a linear function instead of a constant value, and develop a new algorithm called the Piecewise-Planar Reconstruction Method.

3.2.1 The Piecewise Planar Assumption

For the Piecewise-Planar algorithm, we assume that an image can be well approximated as piecewise planar, plus some noise. Furthermore, we assume that most planar regions are reasonably large, so that typically only a small number overlap any one block of the Manhattan grid. We also assume that most planar regions have smooth boundaries.

Specifically, we fix a small integer K, e.g., $K = 3$, and for each block, the first step segments the border into K regions, and K planes that well-approximate the border pixel intensities of each region. We introduce the *K-planes algorithm* for simultaneously performing the segmentation and choosing the planes. Then, as before, the second step extends the segmentation into the interior. This step exploits the assumption that most region boundaries are smooth. These two steps form a piecewise planar approximation for the entire block. Finally, the third step uses the piecewise planar approximation and the border pixels to estimate the block interior pixels using MMSE estimation based on a Gaussian random field model for the block interior whose mean is the piecewise planar approximation.

The main idea behind this piecewise-planer assumption is that although our previous method preserved sharp edges, it also oversharpened "soft" gradual edges. The resulting reconstructions thus had a "painted" effect. Our goal here is to continue preserving sharp edges while avoiding oversharpening gradual edges. By modeling an image as piecewise planar, we can model sharp edges as discontinuities between planes while modeling gradual edges as planes with nonzero slopes.

3.2.2 The Piecewise-Planar Method

Unlike our previous method, this algorithm operates in a true blockwise fashion, since the first step is no longer global. Specifically, for each block $\mathbf{x}$, it operates in three main steps.

1. Using only the border pixels, estimate the planes that will be used to approximate the block border, and label each border pixel with the plane to which it is associated.
2. Segment (label) the interior of the block by associating a plane to each interior pixel, creating a piecewise planar approximation to the block.
3. Estimate the interior pixel intensities using the planes and labelings obtained in Steps 1 and 2.

3.2.3 Step 1: The K-planes algorithm

For some positive integer K, a parameter of the algorithm, and each block $\mathbf{x}$ of the image $\mathbf{f}$, we seek a set of K planes $\{y_1, \cdots, y_K\}$ and a segmentation $\mathbf{v}$ that can be used to approximate the block border $\overline{\mathbf{x}}$. This is challenging because the planes and segmentation for $\mathbf{x}$ must be determined from only its border $\overline{\mathbf{x}}$. However, the fact that the boundary turns corners helps, especially for small blocks, e.g., $N, M \leq 8$.

For $k = 1, \ldots, K$, the equation of the kth plane is

$$y_k(\mathbf{u}) = \mathbf{w}_k^T \mathbf{u} + a_k, \quad k = 1, \ldots, K, \tag{3.2}$$

where $\mathbf{u} \in \mathbb{R}^2$ is a coordinate vector, $\mathbf{w}_k \in \mathbb{R}^2$ is a vector of slope coefficients, and $a_k \in \mathbb{R}$ is an offset parameter. We let $y_k = (\mathbf{w}_k, a_k)$ denote the kth plane and $\mathcal{Y} = \{y_1, \cdots, y_K\}$ denote the set of K planes. Given such a set for block $\mathbf{x}$, each pixel i on its border $\overline{\mathbf{x}}$ is associated with the plane whose value $y_k(\mathbf{u}_i)$ is nearest to x_i. Thus, a label vector $\overline{\mathbf{v}}$ is assigned to the border according to $\overline{v}_i = \text{argmin}_k \, |x_i - y_k(\mathbf{u}_i)|$. Overall, we seek the set of planes $\mathcal{Y}$ that

minimizes the objective function

$$\Lambda(\mathcal{Y}) = \Lambda(y_1, \ldots, y_K) = \sum_{i \in \overline{I}} \min_k \left(x_i - y_k(\mathbf{u}_i)\right)^2, \tag{3.3}$$

which is simply the sum of squared differences between each observed border pixel value and the value of the closest plane.

To minimize Λ for a given K and block border $\overline{\mathbf{x}}$, we introduce the following *K-planes algorithm*, which is an alternating minimization, reminiscent of K-means. It begins with some initial choice of K planes $\mathcal{Y} = \{(\mathbf{w}_1, a_1), \ldots, (\mathbf{w}_K, a_K)\}$, and then iterates the following two steps until a stopping criteria is met.

1. Label each border pixel according to $\overline{v}_i = \text{argmin}_k \ |x_i - y_k(\mathbf{u}_i)|$ and the current set of K planes.

2. For $k = 1 \ldots K$, choose a new plane $(\mathbf{w}_k, a_k)$ that minimizes the sum of squared errors between that plane and all border pixels that are currently labeled k. If no border pixels are currently labeled k, then $(\mathbf{w}_k, a_k)$ remains unchanged.

Finally, after iterations have ceased, the labeling is filtered according to

3. $$\widehat{\overline{v}}_i = \text{mode}\{\overline{v}_i, \overline{\partial}\overline{v}_i\}, \tag{3.4}$$

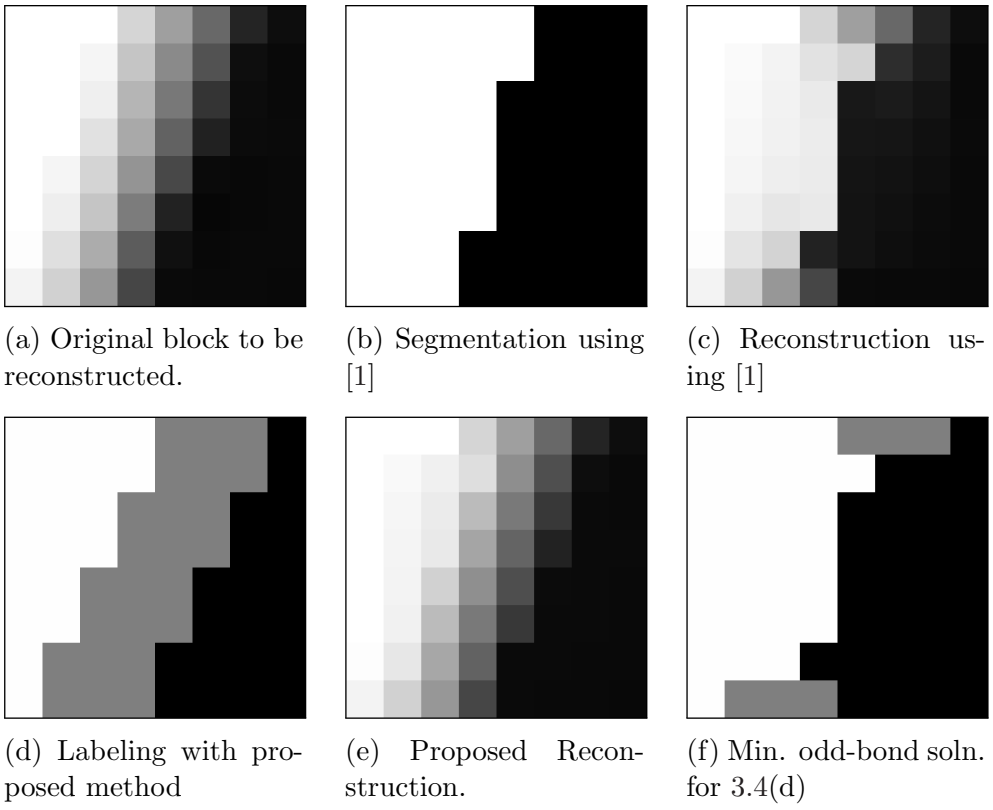
where $\overline{\partial}\overline{v}_i$ denotes the neighbors of $\overline{v}_i$ that are contained in the border and mode$\{\overline{v}_i, \overline{\partial}\overline{v}_i\}$ equals the most frequently occurring element of the set $\{\overline{v}_i, \overline{\partial}\overline{v}_i\}$, if there is one, and equals $\overline{v}_i$ if no element occurs more frequently than the others. This *label filtering* helps avoid having "noisy" border regions. It is possible to incorporate the label filtering into our iterative steps, but we found that we do not lose anything by doing a single filtering at the end.

Step 2 is a 2-D planar regression problem that is solved using the usual matrix pseudoinverse method. It works best when pixels currently labeled k are not all contained in one side of the block.

Note that there is no attempt to make the planes approximating one block match those approximating a neighboring block, but this is clearly something to attempt in future work.

One can stop the algorithm when all parameters of all planes change by less than some small percentage, or one can simply iterate a pre-specified number of times.

For typical images sampled with Manhattan grids of size 8×8 or less, we found $K = 3$ works well. For larger line separations and resulting larger blocks, a larger value of K might be needed. We used 50 iterations as our stopping criteria. For the initial set of planes, we chose $\mathbf{w}_k = (0, 0)$ for each k, making the planes flat, along with offsets $a_1 = \frac{1}{2}\min\{x_i\}$,



(a) Original block to be reconstructed.

(b) Segmentation using [1]

(c) Reconstruction using [1]

(d) Labeling with proposed method

(e) Proposed Reconstruction.

(f) Min. odd-bond soln. for 3.4(d)

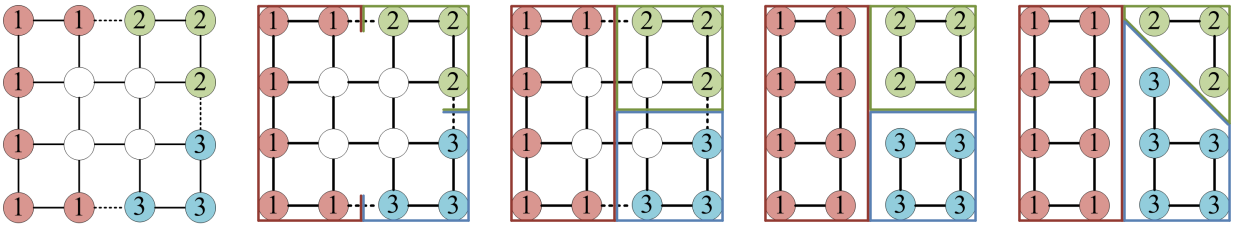
Figure 3.4: Comparison of the method in [1] and the proposed method for a soft edge passing through a 7×7 Manhattan grid sampling of the image "Al", shown in Figure 3.3(a). The block was taken from a soft edge between the books in the upper-right region of the image.

$a_3 = \frac{3}{2} \max \{x_i\}$, and $a_2 = \frac{1}{2}(a_1 + a_3)$. This initialization has several advantages, including that it enables the algorithm to find one plane that fits the entire border, when this is in fact feasible, rather than subdividing the border into three different regions.

3.2.4 Step 2: The interior labeling algorithm

In this step, given the border labeling $\overline{\mathbf{v}}$, we label each interior pixel i with the index $v_i = k$ of one of the K planes, so that this pixel is then planar approximated as $\widehat{x}_i = y_k(\mathbf{u}_i)$. Equivalently, we partition the pixel set I into K regions $\{I_1, \ldots, I_K\}$ and assign a distinct plane to use on each region. We focus the labeling algorithm on choosing such a partition. We experimented with a number of possible approaches and here describe one that worked well.

As background, recall that we model an image as piecewise planar over regions with smooth boundaries. For this reason we place restrictions on the admissible segmentations, and then design an algorithm that gives preferences to segmentations with desirable proper-



(a) Observed border. (b) Initial polygon faces. (c) Optimum polygons. (d) Final partitioning. (e) Suboptimum.

Figure 3.5: Estimation of $\widetilde{\mathbf{v}}$ in Step 2 for a block taken from a 3×3 Manhattan grid. Nodes and polygons associated with plane 1 are colored red, plane 2 are colored green, and plane 3 are colored blue.

ties.

We view the segmentation task as a partition problem, where our goal is to partition a rectangular-shaped subset of $\mathbb{R}^2$. Recall that the pixels of block $\mathbf{x}$ are considered to have a graph structure characterized by (I, E). To describe the admissible partitions we also take into account the Euclidean structure of the pixels. In particular, let $B \subset \mathbb{R}^2$ denote the smallest rectangle in $\mathbb{R}^2$ that contains all block pixel locations, i.e., $B \supset \{\mathbf{u}_i : i \in I\}$. We restrict attention to partitions of I generated by a partition of B into K regions $\{B_1, \ldots, B_K\}$, via $I_k = \{i : \mathbf{u}_i \in B_k\}$, $k = 1, \ldots, K$. Furthermore, we add the following requirements:

- Each partition region B_k must be bounded by straight line segments, i.e., it is either a polygon or the union of polygons.
- If there is an odd bond between adjacent border pixels i and j, then regions B_{v_i} and B_{v_j} must each have a vertex on the line segment connecting $\mathbf{u}_i$ and $\mathbf{u}_j$. Additionally, no other region can have a vertex on this line segment.

Now, among admissible partitions B we choose the one that satisfies the following conditions.

1. The number of polygon faces that cross the block from one side to another is as large as possible, where if two polygons have coincident faces, then both are counted.
2. Among partitions satisfying 1, the sum of the lengths of the cross-the-block faces is as large as possible.
3. Among partitions satisfying 1 and 2, the total number of all faces is as small as possible.
4. Among partitions satisfying 1-3, the sum of the lengths of all faces is as small as possible.

The algorithm for interior labeling described in Step 2 of Section 3.1 required that an admissible partition must minimize the total number of odd bonds. However, we found that such partitions often lead to poor interior labelings that did not satisfy our requirement of smooth boundaries between planar regions. For example the block in Figure 3.4(f) is a minimum odd-bond segmentation that fails to connect two segments with the same label on opposite sides of the block. Conditions 1-4, however, emphasize odd bond configurations that form long, smooth boundaries between regions, as shown in Figure 3.4(d)

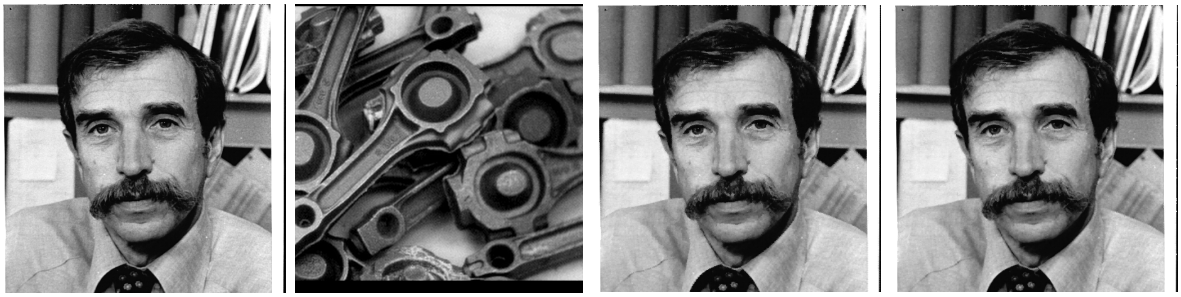
As an example of how an algorithm might find an interior that satisfies these conditions, consider Figure 3.5(a) for a 3×3 block whose border has been labeled. Note that there are three odd bonds. As a first step, it is necessary to to ensure that each observed odd bond on the boundary contains vertices of the two adjacent regions. To illustrate this, in Figure 3.5(b), polygon faces have been drawn through each odd bond on the boundary, with each face (denoted with color) corresponding to a particular region type. Figure 3.5(c) shows the optimum configuration of polygons satisfying conditions 1-4. Figure 3.5(d) shows the removal of edges corresponding to odd bonds in the labeling, as specified in the next subsection. Figure 3.5(e) shows a suboptimum configuration; the total length of its faces can be reduced by making the "2-3" pair of faces orthogonal to the "1"-face as in Figure 3.5(d).

Step 2 of our algorithm is implemented by finding configurations that satisfy Conditions 1-4 when there are four or fewer observed odd bonds on the border. The case-by-case algorithm is listed in Appendix A. When there are more than four odd bonds on the boundary, bonds are removed until they total four or fewer. This removal processed involves finding two odd bonds of the same type in close proximity and reclassifying the pixels in between. For example, if a "2-3" bond and another "2-3" bond are separated by two pixels labeled "3", then those pixels enclosed by the bonds are relabeled as "2." If no such pairs are found, then mode filtering (as in K planes algorithm) with an increasing window is applied repeatedly to the border until such a bond is found, or the total bonds were sufficiently reduced. This suboptimum procedure is not needed frequently, but is necessary to reduce the complexity of Step 2.

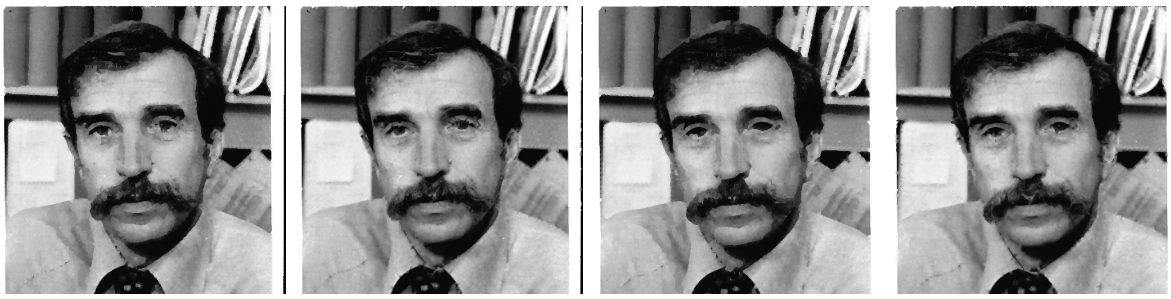
3.2.5 Step 3: Block Reconstruction

This step is identical to that of Section 3.1.2.3 with a few subtle adjustments. The conditional Gaussian MRF is now conditional on our set of K planes $\mathcal{Y}$:

$$p(\mathbf{x}|\mathbf{v}, \mathcal{Y}) = \frac{1}{Z} \prod_{i \in I} \phi_i(x_i) \prod_{\{i,j\} \in E'} \psi_{i,j}(x_i, x_j). \tag{3.5}$$



(a) Original image "Al" (b) Original image "Tools" (c) 4×3, Cutset-MRF [1] (d) 4×3, Piecewise-Planar



(e) 7×7, Cutset-MRF [1] (f) 7×7, Piecewise-Planar (g) 8×8, Cutset-MRF [1] (h) 8×8, Piecewise-Planar

Figure 3.6: Comparison of the proposed Piecewise-Planar method to the previous "MRF model with cutset segmentation" method described in [1].

In particular, the mean at interior pixel x_i is $\mu_i = y_{v_i}(\mathbf{u}_i)$ and comes from the piecewise planar approximations i.e. it is simply value of the plane associated with pixel i. Additionally, $c = 0.25$ is chosen instead of $c = 0.26$ to ensure that R^{-1} is invertible[3]. Once again, the usual linear MMSE method is used to estimate the interior pixels $\widetilde{\mathbf{x}}$ from the border pixels $\overline{\mathbf{x}}$.

3.2.6 Results

Figure 3.6 shows the results of the proposed algorithm for three Manhattan grid sizes. It also contains images reconstructed using the Cutset-MRF method described in [1] for comparison. When reading this thesis in electronic form, the reader is urged to expand the images to see the differences. There is significant visual improvement in the proposed method; the most noticeable improvement is seen in the edges between regions. Consider the books in the

[3]As with the Cutset-MRF method, we empirically chose c to be large enough such that R^{-1} is invertible. The careful reader may realize that this choice of $c = 0.25$ would lead to a singular inverse covariance matrix R^{-1} under the assumption of circulant boundary conditions on the graph. However, we did not use circulant boundary conditions; pixels on boundaries instead had fewer than 4 neighbors.

top part of the image "Al." As the example in Figure 3.4 shows, these books contain many gradual "soft" edges. Although the method in [1] reconstructs sharp edges very well, it tends to oversharpen these soft edges. The proposed method does a much better job at preserving these soft edges, while still maintaining sharp edges like the collar of the shirt in "Al." Additionally, the method in [1] produces regions with a "painted" look to them, especially at coarser sampling densities, whereas the same regions produced by the new method do not look artificial. This difference is easily seen in the region directly above the shoulder in the right side of the image "Al." The reason for this difference lies in the way that each method models the mean of each region. The method in [1] models the mean as a constant value (i.e. a flat plane), which leads to oversharpening of edges; thus, it can only model edges in the image as discontinuities. The proposed method overcomes this issue by modeling the means of each pixel as the labeled plane value at that coordinate. This enables "soft" edges to be modeled by planes instead of discontinuities. Finally, it is important to note that the proposed model tends to create crevasse-like artifacts. These can be seen in the eyes and along the collar of Figure 3.6(f). These can occur when two border segments of a block have similar intensities, but the pixels inbetween are very different. For example, the eyebrow and eyes of "Al" are both dark, but light skin region inbetween the eyes and eyebrows is incorrectly modeled as a dark region.

In addition to these visual improvements, there is an overall improvement in the Peak Signal-to-Noise Ratio (PSNR) for reconstructions obtained using the proposed method over the method in [1], where PSNR (in dB) is defined to be

$$PSNR = 20 \log_{10} \left(\frac{x_{max}}{\sqrt{MSE}} \right), \tag{3.6}$$

where MSE is the mean-squared error between the original image and its reconstruction. For all experiments in this chapter, the images intensities are restricted to the range of $[0, 255]$, so we have that $x_{max} = 255$ in (3.6). The resulting PSNR values for our experiments are shown in Table 3.1. The images were truncated on the bottom and right edges to ensure the outside border was fully sampled using the given Manhattan sampling scheme (for example, for 4×3 Manhattan sampling, the number of pixel rows must equal $4m + 1$ for some integer m in order to fully sample the last row of blocks). Furthermore, the calculated PSNR values ignore a 20 pixel-wide border around the image to avoid edge effects caused by noisy pixels in the upper part of "Al" and the unnatural black lines on the right side of "Al" and the top/bottom of "Tools." In all cases, the Proposed Piecewise-Planar method outperforms the MRF Cutset method in PSNR.

Grid	Al		Tools	
	MRF Cutset	Piecewise-Planar	MRF Cutset	Piecewise-Planar
4×3	32.2	34.3	32.7	36.1
7×7	27.2	29.1	26.4	28.0
8×8	25.9	28.0	25.4	26.6

Table 3.1: Comparison of PSNRs in dB between the "MRF model with cutset segmentation" method in [1] and new Piecewise-Planar method for $c = 0.25$ and $d = 1$. Values in a 20 pixel-wide border around the image were not used in these calculations.

3.3 Orthogonal Gradient (OG) Algorithm

This section presents another new method for interpolating pixels from their Manhattan samples called the *orthogonal gradient (OG) algorithm*, which exploits the fact that pixels tend to be highly correlated along the direction orthogonal to the image gradient. Such an approach is enhanced by Manhattan sampling, where dense sampling along straight lines allows for better reconstruction of both sharp and soft image edges. In particular, the OG algorithm alternates between solving a constrained optimization problem, and changing the weights of the optimization problem according to the direction of the gradient of each new image estimate. The proposed method improves upon previously Manhattan interpolation algorithms, both qualitatively as well as in mean-squared error.

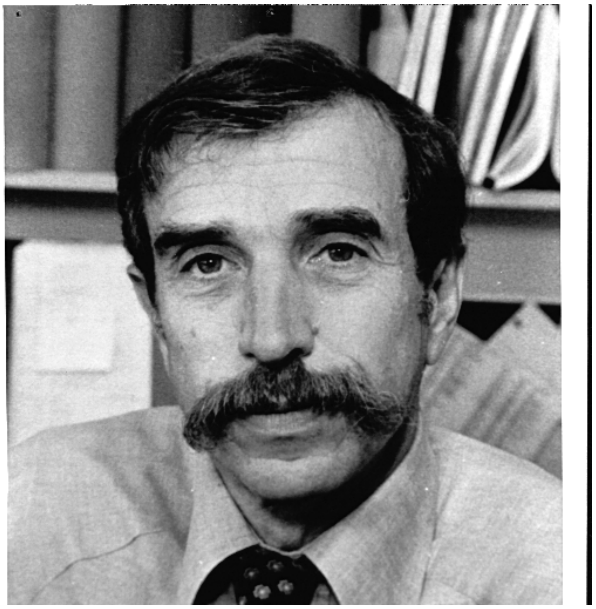
3.3.1 Constrained Optimization Problem Formulation and Solution

Suppose we lexicographically order an $M \times N$ image into a vector $\mathbf{x} \in \mathbb{R}^{n_p}$, where $n_p = MN$ is the number of pixels. Let $x_i = [\mathbf{x}]_i$ be the intensity of pixel i. Let x_i^N, x_i^S, x_i^E, x_i^W, x_i^{NE}, x_i^{NW}, x_i^{SE}, and x_i^{SW} denote intensities of the north, south, east, west, northeast, northwest, southeast, and southwest neighbors of pixel i, respectively, as shown in Fig. 3.7(a). Let $S \in \mathbb{R}^{n_s \times n_p}$ be the sampling matrix formed by deleting the ith row from the identity matrix $I_{n_p \times n_p}$ wherever pixel i is not sampled, where $n_s < n_p$ is the number of samples. Let $\mathbf{y} = S\mathbf{x}$ be the vector of samples. Our goal is to estimate $\mathbf{x}$ from $\mathbf{y}$, particularly in the case where S corresponds to $k_1 \times k_2$ Manhattan sampling, which denotes sampling every k_1th row and k_2th column of pixels, as shown in Fig. 3.7(c), We formulate our interpolation problem as the constrained optimization problem

$$\underset{\mathbf{x}}{\text{minimize}} \quad \Psi(\mathbf{x}) \quad \text{subject to} \quad S\mathbf{x} = \mathbf{y}, \quad 0 \leq x_i \leq 255, \tag{3.7}$$

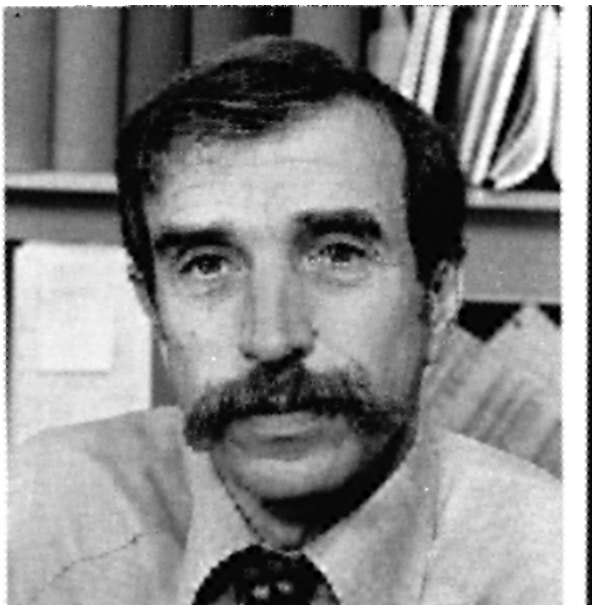
x_i^{NW}	x_i^N	x_i^{NE}
x_i^W	x_i	x_i^E
x_i^{SW}	x_i^S	x_i^{SE}

(a) 8-point neighbors of pixel i.



(b) Original image 'Al'

(c) Sampled on 7×7 M-Grid



(d) Minimizes Ψ_{iso}, 7×7 M-Grid

Figure 3.7: Neighborhood definitions and a Manhattan interpolation obtained by solving problem (3.7) with isotropic objective $\Psi_{iso}(\mathbf{x})$.

where $\Psi(\mathbf{x})$ is an *objective function* that captures some prior information about $\mathbf{x}$. One possible choice of an objective function is

$$\Psi_{iso}(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{n_p}[(2\sqrt{2}(1+\sqrt{2})x_i - (x_i^N + x_i^S + x_i^W + x_i^E) - \tfrac{1}{\sqrt{2}}(x_i^{NW} + x_i^{NE} + x_i^{SW} + x_i^{SE})]^2.$$

This function encourages smoothness by summing differences between pixel i and its eight neighbors, and then squaring this quantity. The $1/\sqrt{2}$ factor encourages isotropy, since diagonal pixels are $\sqrt{2}$ further away than their north/south/east/west counterparts. Equivalently, $\Psi_{iso}(\mathbf{x})$ is the summed energy of the output of the following Laplacian filter applied to $\mathbf{x}$:

$$\mathbf{H} = \begin{bmatrix} -1/\sqrt{2} & -1 & -1/\sqrt{2} \\ -1 & 2\sqrt{2}(1+\sqrt{2}) & -1 \\ -1/\sqrt{2} & -1 & -1/\sqrt{2} \end{bmatrix}.$$

Thus, the objective function Ψ_{iso} penalizes images the least that have a small 2nd derivative, i.e., images that are piecewise linear. This is similar to the piecewise-planar plus noise model used previously for Manhattan interpolation in Section 3.2. An example of an image that minimizes $\Psi_{iso}(\mathbf{x})$ while satisfying the equality constraint $S\mathbf{x} = \mathbf{y}$ for a 7×7 Manhattan grid is shown in Figure 3.7(d). This image is clearly unsatisfactory, as the blurry edges are very distracting.

Instead of Ψ_{iso}, we propose the anisotropic objective function

$$\begin{aligned} \Psi(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{n_p} \big[& w_i^{N,S}(2x_i - x_i^N - x_i^S) \\ & + w_i^{W,E}(2x_i - x_i^W - x_i^E) \\ & + w_i^{NE,SW}\tfrac{1}{\sqrt{2}}(2x_i - x_i^{NE} - x_i^{SW}) \\ & + w_i^{NW,SE}\tfrac{1}{\sqrt{2}}(2x_i - x_i^{NW} - x_i^{SE})\big]^2, \end{aligned} \tag{3.8}$$

where the $w_i^{N,S}, w_i^{W,E}, w_i^{NE,SW}, w_i^{NW,SE}$'s are positive weights that take values in the unit interval $[0, 1]$ and sum to one. The key idea here is that we would like an intelligent choice of weights that prevents blurring across edges. One way to do this is to use the spatial gradient of the image[4]. Images tend to have similar pixel intensities in directions *orthogonal* to the spatial gradient direction. For example, if there is a sharp image transition (i.e. an edge) going north-to-south near pixel i, we would like $w_i^{N,S}$, $w_i^{NE,SW}$, and $w_i^{NW,SE}$ to be small to discourage blurring across the edge. Furthermore, we would like $w_i^{W,E}$ to be large to encourage similarities in the direction orthogonal to the edge.

Let us write this function more compactly in matrix-vector notation. First, each quantity in (3.8) before squaring is the output of a linear combination of the following four filters

[4]Since the image $\mathbf{x}$ is unknown, we instead use the spatial gradient of a previous image reconstruction $\widehat{\mathbf{x}}$. Thus, our weights W will depend not on the original image $\mathbf{x}$, but on some "previous" reconstruction $\widehat{\mathbf{x}}$.

applied to $\mathbf{x}$:

$$\mathbf{H}_{N,S} = \begin{bmatrix} 0 & -1 & 0 \\ 0 & 2 & 0 \\ 0 & -1 & 0 \end{bmatrix}, \mathbf{H}_{NW,SE} = \frac{1}{\sqrt{2}} \begin{bmatrix} -1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & -1 \end{bmatrix},$$
$$\mathbf{H}_{W,E} = \begin{bmatrix} 0 & 0 & 0 \\ -1 & 2 & -1 \\ 0 & 0 & 0 \end{bmatrix}, \mathbf{H}_{NE,SW} = \frac{1}{\sqrt{2}} \begin{bmatrix} 0 & 0 & -1 \\ 0 & 2 & 0 \\ -1 & 0 & 0 \end{bmatrix}.$$

Note that these filters decompose $\mathbf{H}$ into four directions offset by $\pi/4$, since $\mathbf{H} = \mathbf{H}_{N,S} + \mathbf{H}_{W,E} + \mathbf{H}_{NW,SE} + \mathbf{H}_{NE,SW}$. Thus, choosing $w_i = 1$ for all weights reduces $\Psi(\mathbf{x})$ to the isotropic objective function $\Psi_{iso}(\mathbf{x})$. Denote the lexicographically-ordered vectorized output of filter $\mathbf{H}_{N,S}$ applied to $\mathbf{x}$ as the matrix-vector operation $C_{N,S}\mathbf{x} = [(2x_1 - x_1^N - x_1^S), \cdots, (2x_{n_p} - x_{n_p}^N - x_{n_p}^S)]^T$, where $C_{N,S}$ is an $n_p \times n_p$ matrix. Similarly, define $C_{W,E}$, $C_{NE,SW}$, and $C_{NW,SE}$ as linear filtering operations according to the kernels $\mathbf{H}_{W,E}$, $\mathbf{H}_{NE,SW}$ and $\mathbf{H}_{NW,SE}$, respectively. Stack these matrices to define the $4n_p \times n_p$ matrix $C = \left[C_{N,S}^T, C_{W,E}^T, C_{NE,SW}^T, C_{NW,SE}^T\right]^T$. Furthermore, define the north/south diagonal weighting matrix to be $W_{N,S} = \text{diag}(\{w_i^{N,S}\}_{i=1}^{n_p})$; similarly define the diagonal matrices $W_{W,E}$, $W_{NE,SW}$, and $W_{NW,SE}$. Again, stack these weighting matrices to form the $4n_p \times n_p$ matrix $W = \left[W_{N,S}^T, W_{W,E}^T, W_{NE,SW}^T, W_{NW,SE}^T\right]^T$. We can now compactly rewrite the objective function as

$$\Psi(\mathbf{x}) = \frac{1}{2}\|W^T C\mathbf{x}\|_2^2.$$

Note that our objective function $\Psi(\mathbf{x})$ is convex[5], since $C^T W W^T C$ is positive semidefinite.

Since one of our constraints in (3.7) is a linear equality, it can be eliminated by following the process in [41, p. 523]. Our linear equality constraint requires that our solution lie in the set $\{\mathbf{x} : S\mathbf{x} = \mathbf{y}\}$. Let F be an $n_p \times (n_p - n_s)$ matrix whose range is the nullspace of S. In particular, for this sampling and interpolation problem, we choose F to be the matrix obtained by taking the identity matrix $I_{n_p \times n_p}$ and deleting column i if pixel i is sampled. Let $\widetilde{\mathbf{x}}$ be any vector whose Manhattan samples match our observations $S\widetilde{\mathbf{x}} = \mathbf{y}$. Clearly then, $\{\mathbf{x} : S\mathbf{x} = \mathbf{y}\} = \{F\mathbf{z} + \widetilde{\mathbf{x}} : \mathbf{z} \in \mathbb{R}^{n_p - n_s}\}$. We rewrite our optimization problem as

$$\begin{aligned} \underset{\mathbf{z}}{\text{minimize}} \quad & \widetilde{\Psi}(\mathbf{z}) \triangleq \Psi(F\mathbf{z} + \widetilde{\mathbf{x}}) \\ \text{subject to} \quad & a \leq [F\mathbf{z} + \widetilde{\mathbf{x}}]_i \leq b, \ i = 1, \ldots, n_p, \end{aligned} \tag{3.9}$$

[5]Again, due to the fact that W is fixed and chosen using some previous image reconstruction $\widehat{\mathbf{x}}$; W does not depend on the variable $\mathbf{x}$.

where our modified objective function is a function of $\mathbf{z} \in \mathbb{R}^{n_p - n_s}$. If $\mathbf{z}^*$ solves this problem, then our final image estimate is

$$\mathbf{x}^* = F\mathbf{z}^* + \widetilde{\mathbf{x}}. \tag{3.10}$$

Although not true for general F, our choice of F with the inequality constraints in (3.7) form box constraints that can be solved with the gradient projection method [42]. This is a special case of the proximal gradient method [43] where the prox function is hard thresholding to $[0, 255]$.

For reference, the objective function has gradient

$$\nabla_{\mathbf{z}} \widetilde{\Psi}(\mathbf{z}) = F^T C^T W W^T C F \mathbf{z} + F^T C^T W W^T C \widetilde{\mathbf{x}}.$$

The spectral radius of the Hessian of $\widetilde{\Psi}(\mathbf{z})$ is upper bounded by $32(3 + 2\sqrt{2})$. Choosing a step size of $(32(3 + 2\sqrt{2}))^{-1}$ guarantees convergence of our objective function. In practice, gradient projection is very slow. In our implementation, we first ignored the box constraints and minimized the objective function in (3.9) using conjugate gradient descent. The solution to this unconstrained problem is used as an initial estimate to "warm-start" the gradient projection method. Note that all gradient descent steps can implemented efficiently because W, W^T, C, and C^T are linear filtering operations, and the F and F^T are sampling/zero insertion operations. Thus, we avoid storing or inverting any large matrices. We note that filtering operations were not performed on the border to avoid edge effects. Our stopping criterion for the gradient projection method was $|\widetilde{\Psi}(\mathbf{z}^{n+1}) - \widetilde{\Psi}(\mathbf{z}^n)/\widetilde{\Psi}(\mathbf{z}^{n+1})| < \epsilon_1 = 10^{-4}$.

3.3.2 The Orthogonal Gradient (OG) Algorithm

Thus far, we have not described how to choose the weighting matrix W. In this section, we describe the *Orthogonal Gradient* (OG) algorithm, which alternates solving an optimization problem of the form (3.9) and choosing a new weighting matrix W. Earlier, we posited that smooth images tend to have pixel similarities in directions orthogonal to the image gradient.

Suppose we have an image reconstruction $\widehat{\mathbf{x}}$, and let $\theta_i = \angle([\nabla \widehat{\mathbf{x}}]_i)$ denote the angle of the reconstruction spatial gradient at pixel i. Using the convention that "south" has angle

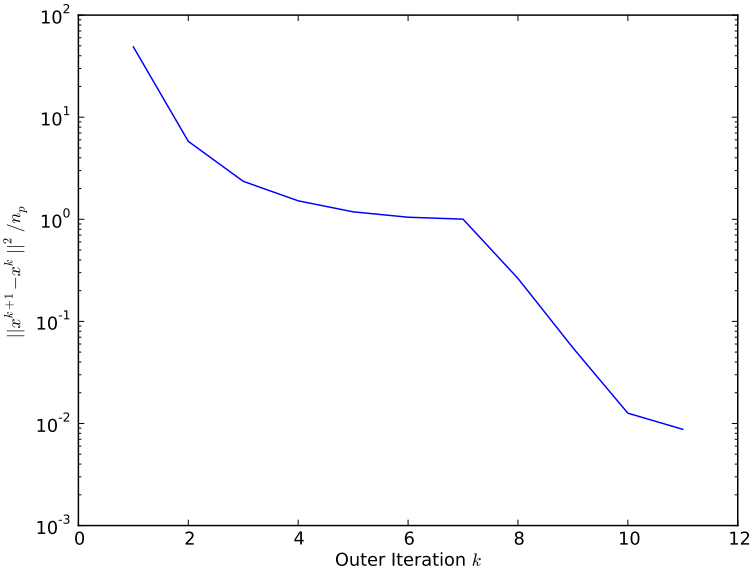


Figure 3.8: Plot of $\|\mathbf{x}^{k+1} - \mathbf{x}^k\|^2/n_p$ for reconstructing "Al" sampled on a 7×7 Manhattan grid using the OG Algorithm. Note the image enters a limit cycle, which is eliminated by decreasing the λ^k sequence at $k \geq 7$.

0 and "east" has angle $\pi/2$, one choice of weights is

$$\begin{aligned}
\alpha_i^{N,S}(\theta_i) &= \left(1 - \tfrac{4}{\pi}\left|\theta_i - \tfrac{\pi}{2}\right|\right)\mathbf{1}\left(\theta_i \in \left[\tfrac{\pi}{4}, \tfrac{3\pi}{4}\right]\right) \\
&\quad + \left(1 - \tfrac{4}{\pi}\left|\theta_i + \tfrac{\pi}{2}\right|\right)\mathbf{1}\left(\theta_i \in \left[-\tfrac{3\pi}{4}, -\tfrac{\pi}{4}\right]\right), \\
\alpha_i^{W,E}(\theta_i) &= \left(1 - \tfrac{4}{\pi}\left|\theta_i - \pi\right|\right)\mathbf{1}\left(\theta_i \in \left[\tfrac{3\pi}{4}, \pi\right]\right) \\
&\quad + \left(1 - \tfrac{4}{\pi}\left|\theta_i\right|\right)\mathbf{1}\left(\theta_i \in \left[-\tfrac{\pi}{4}, \tfrac{\pi}{4}\right]\right) \\
&\quad + \left(1 - \tfrac{4}{\pi}\left|\theta_i + \pi\right|\right)\mathbf{1}\left(\theta_i \in \left[-\pi, -\tfrac{3\pi}{4}\right]\right), \\
\alpha_i^{NE,SW}(\theta_i) &= \left(1 - \tfrac{4}{\pi}\left|\theta_i - \tfrac{\pi}{4}\right|\right)\mathbf{1}\left(\theta_i \in \left[0, \tfrac{\pi}{2}\right]\right) \\
&\quad + \left(1 - \tfrac{4}{\pi}\left|\theta_i + \tfrac{3\pi}{4}\right|\right)\mathbf{1}\left(\theta_i \in \left[-\pi, -\tfrac{\pi}{2}\right]\right), \\
\alpha_i^{NW,SE}(\theta_i) &= \left(1 - \tfrac{4}{\pi}\left|\theta_i - \tfrac{3\pi}{4}\right|\right)\mathbf{1}\left(\theta_i \in \left[\tfrac{\pi}{2}, \pi\right]\right) \\
&\quad + \left(1 - \tfrac{4}{\pi}\left|\theta_i + \tfrac{\pi}{4}\right|\right)\mathbf{1}\left(\theta_i \in \left[-\tfrac{\pi}{2}, 0\right]\right),
\end{aligned}$$

where $\mathbf{1}(E)$ is the indicator function for event E. That is, we choose $w_i^{N,S} = \alpha_i^{N,S}(\theta_i)$, and similarly for $w_i^{W,E}$, $w_i^{NE,SW}$, and $w_i^{NW,SE}$. For succinctness, we summarize this choice of weights with the notation

$$W = A(\nabla\widehat{\mathbf{x}}).$$

where

$$A(\nabla \mathbf{x}) = \Big[\text{diag}\{\alpha_i^{N,S}(\theta_i)\}\ \ \text{diag}\{\alpha_i^{W,E}(\theta_i)\},\ \text{diag}\{\alpha_i^{NE,SW}(\theta_i)\},\ \text{diag}\{\alpha_i^{NW,SE}(\theta_i)\}\Big]^T.$$

Note that the α_i functions are equal to unity when θ_i lies orthogonal to the given direction and decrease linearly to zero with slope $4/\pi$. Thus, these are "triangle functions" of width $\pi/2$ centered at the orthogonal directions. For example, $\alpha_i^{N,S}(\theta_i)$ equals 1 when the gradient points in the "west" or "east" direction. Note that we have chosen these functions so that $\alpha_i^{N,S} + \alpha_i^{W,E} + \alpha_i^{NW,SE} + \alpha_i^{NE,SW} = 1$.

To calculate the spatial gradient, we filtered the image with the well known *Sobel operators* [44, p. 131] to obtain the $x-$ and $y-$gradient values. The initial weights W^0 are chosen isotropically by setting all w_i's to one, which is equivalent to choosing $\Psi(\mathbf{x}) = \Psi_{iso}(\mathbf{x})$ for the first pass. If we continually update the weights according to $W^{k+1} = A(\nabla \mathbf{x}^{k+1})$, the sequence W^k may enter a limit cycle. To avoid this, we instead update the weights according to $W^{k+1} = W^k + \lambda^k(A(\nabla \mathbf{x}^{k+1}) - W^k)$, where λ_k is a sequence of relaxation parameters satisfying $0 < \lambda^k \leq 1$, $\lambda^{k+1} \leq \lambda^k$, and $\lambda^k \to 0$. Using an induction argument it can be shown that the change in successive weights is bounded by $|[W^{k+1}]_{ij} - [W^k]_{ij}| \leq \lambda^k$, and thus the sequence of weighting matrices W^k will converge. In our experiments, we chose $\lambda^k = 1$ for $0 \leq k \leq 6$, and then decreased them according to $\lambda^k = (k-6)^{-1}$ for $k \geq 7$. The algorithm stops when average square differences between image estimates is less than a threshold. Specifically, we used the stopping criterion $\|\mathbf{x}^{k+1} - \mathbf{x}^k\|^2/n_p < \epsilon_2 = 10^{-2}$. The final OG algorithm is summarized above.

Finally, it would be desirable to claim that the sequence of image estimates $\|\mathbf{x}^{k+1} - \mathbf{x}^k\|$ converges. Although we are unable to present an analytic proof of such, we observed this empirically, as shown in Figure 3.8.

The reader who is interested in experimental results for the OG algorithm is encouraged to skip ahead to Section 3.5.

3.4 Local Orthogonal Orientation Penalization (LOOP) Algorithm

One drawback to the OG algorithm is that is uses an anisotropic penalization at every pixel, namely, it only penalizes differences in the directions orthogonal to the image gradient. However, in smooth (flat) regions of the image, it will be desirable to penalize isotropically in all directions to avoid "false contour" artifacts. The Local Orthogonal Orientation Penalization (LOOP) algorithm attempts to avoid this by calculating a local "dominant" gradient

1: Set $k = 0$, choose initial $\mathbf{x}^0$, W^0, and relaxation parameter sequence λ^k satisfying conditions in Section 3.3.2.
2: **repeat**
3: Set $\mathbf{z}^{k+1}$ to solution of optimization problem (3.9) using
4: current estimate $\widetilde{\mathbf{x}} = \mathbf{x}^k$ and current weights $W = W^k$.
5: $\mathbf{x}^{k+1} \leftarrow F\mathbf{z}^{k+1} + \mathbf{x}^k$
6: $W^{k+1} \leftarrow W^k + \lambda^k\big(A(\nabla \mathbf{x}^{k+1}) - W^k\big)$
7: $k \leftarrow k + 1$
8: **until** stopping criterion is satisfied.

Figure 3.9: Orthogonal Gradient (OG) Algorithm

strength and direction at each pixel using a Singular Value Decomposition (SVD) of the local gradient vectors from the previous image estimate. The difference of the two singular values at each pixel can be used as a measure of gradient strength in the region surrounding each pixel, and is used to locally adapt the cost function to be more isotropic, or to be more anisotropic. Note that this allows us to take into account gradient magnitudes in addition to gradient directions, whereas the OG algorithm only uses gradient directions to calculate its parameters.

3.4.1 The Dominant Gradient Strength and Direction

Recall that the OG algorithm uses the gradient directions of the previous image estimate to determine the parameters of an optimization problem. These gradient directions are highly susceptible to noise, especially in flat areas of the image. In particular, since the OG algorithm only penalizes pixel differences in two directions (the directions orthogonal to the previous image gradient), it tends to create "false contour" artifacts in flat regions of the image (see discussion and images in Section 3.5). If we were to take into account the gradient magnitude as well as direction, we could instead use an isotropic cost function in flat regions of the image; this isotropic cost function penalizes differences in *all* directions, and thereby eliminates or reduces these artifacts.

Additionally, instead of using a single gradient vector at each pixel to calculate our parameters, it is possible to use several gradient vectors located in a neighborhood around each pixel. One may be tempted to use the average gradient vector in a window surrounding each pixel as a statistic of interest. However, in regions with sharp edges, typically the gradient field is sparse, since the gradient is strong along the edge, and zero otherwise. If we were to use the average gradient vector to calculate our parameters near a sharp edge, the average gradient would be very small relative to the strong gradient vectors along the edge.

This would cause our algorithm to use an isotropic cost function near edge regions, which would in turn cause edges to become blurred in the final image estimate.

Instead of calculating the average gradient in a window around each pixel, we propose finding the unit vector that maximizes the average inner product between itself and each vector in a pixel neighborhood; this is called the *dominant gradient*. It is well known that the solution to this problem is the first left singular vector of the data matrix formed from the neighborhood of gradient vectors, and this singular vector corresponds to the largest singular value of the data matrix [45].

However, in addition to using the first left singular vector of this "local SVD" computation, one can also use the singular values as a measure of the "strength" of the dominant gradient direction. In particular, if both singular values are similar, then the region likely contains no sharp edges or strong gradients. However, if the first singular value is much larger than the second singular value, then it is likely that the region contains an edge or a large gradient, i.e. the dominant gradient direction is "strong." Thus, when designing our algorithm, we would like to use an isotropic penalty in regions where the singular values are the same, and an anisotropic penalty in region where the singular values are different.

We begin by describing how to calculate the dominant gradient direction at each pixel. Suppose we are given the gradient g_i at each pixel i of an image. Let $\mathcal{N}_i$ be a neighborhood of pixel i, for example, a $\sqrt{n} \times \sqrt{n}$ window centered at i. For each pixel i, we generate a matrix of n gradient vectors by arranging all neighboring gradient vectors column-wise in a matrix $A_i = [g_j]_{j \in \mathcal{N}_i} \in \mathbb{R}^{2 \times n}$. We define the *dominant gradient direction* to be the unit vector v_i that maximizes the average inner product of v_i with the columns of A_i, i.e.

$$v_i = \underset{v:\|v\|=1}{\operatorname{argmax}} \, v^T A_i A_i^T v. \tag{3.11}$$

As noted earlier, the solution to this problem is the first left singular value of A_i, or the eigenvector of $A_i A_i^T$ corresponding to its largest eigenvalue.

For some brief insight into why this is useful, suppose that the true gradient of the image is piecewise constant, but due to noise and/or discretization, the observed vectors $[g_j]_{j \in \mathcal{N}_i}$ are i.i.d. Gaussian $g_j \sim \mathcal{N}(\gamma u, \sigma^2 I_{2\times 2})$. where $\gamma > 0$, $u \in \mathbb{R}^2$, and $\|u\| = 1$. As mentioned in [45] and shown in [46], the maximum likelihood estimate of u given $[g_j]_{j \in \mathcal{N}_i}$ is exactly v_i. For some additional intuition as to why this is the case, let $C = \frac{1}{n} A_i A_i^T \in \mathbb{R}^{2 \times n}$. It can be shown (see Appendix B) that the expected value of the matrix C has the eigenvalue decomposition

$$\mathrm{E}\left[\frac{1}{n} A_i A_i^T\right] = \left[u, \; u^\perp\right] \operatorname{diag}\{\gamma^2 + \sigma^2, \sigma^2\} \left[u, \; u^\perp\right]^T,$$

and we see that the eigenvector corresponding to the largest eigenvalue is exactly u.

We can modify the OG algorithm to make use of the dominant gradient direction. Instead of using the angle of the gradient at each pixel i, we can instead use the direction of the first left singular vector

$$\theta_i = \angle v_i \tag{3.12}$$

and then design our cost function to penalize differences in the two directions orthogonal to θ_i. However, as noted earlier, we do not always want to only penalize differences in two directions at each pixel, as this will lead to "false contour" artifacts. Fortunately, the SVD of A_i provides us with a nice measure of the "strength" of the dominant gradient in the form of singular values (equivalently, eigenvalues of $A_i A_i^T$).

In order to determine whether to use the isotropic or anisotropic penalty function, we propose using the *dominant gradient strength*, which is the normalized quantity

$$\alpha_i = \begin{cases} \frac{\lambda_{i,1} - \lambda_{i,2}}{\lambda_{i,1}}, & \lambda_{i,1} > 0, \\ 0, & \lambda_{i,1} = 0, \end{cases} \tag{3.13}$$

where $\lambda_{i,1}$ and $\lambda_{i,2}$ are the largest and smallest eigenvalues of $A_i A_i^T$, respectively, and α_i is guaranteed to be between 0 and 1. This is similar to the parameter R chosen in [45]. To motivate this choice of parameter, let us return to the case where $[g_j]_{j \in \mathcal{N}_i}$ are i.i.d. Gaussian assumption. If $\gamma \gg \sigma$, we expect the eigenvalues of $A_i A_i^T$ to be very different, and $\alpha_i \approx 1$. However, when $\gamma \ll \sigma$, we expect the eigenvalues of $A_i A_i^T$ to be very close to one another, and $\alpha_i \approx 0$. We can thus use α_i as a way to trade-off between using an isotropic and anisotropic cost function; when $\alpha_i \approx 0$, we will use an isotropic cost function that penalizes pixel differences in all directions, and when $\alpha_i \approx 1$, we will use an anisotropic cost function that only penalizes pixel differences in the two directions orthogonal to v_i. We also note that α_i is exactly one minus the inverse condition number of the matrix $A_i A_i^T$.

Finally, in Section 3.4.4 we describe how v_i and α_i can be calculated very efficiently using filtering operations and basic array arithmetic. This avoids the naive method of iterating through every pixel i, building a local gradient matrix A_i, and using a built-in SVD solver (i.e. Matlab's solver or a Python library) to calculate v_i and α_i.

3.4.2 LOOP Algorithm

In this section, we give a high-level overview of the LOOP algorithm and its individual steps. For a final, exact summary of the LOOP algorithm, the reader can consult Figure 3.10. Similar to the OG algorithm, the Local Orthogonal Orientation Penalization (LOOP)

algorithm is an alternating algorithm that can be summarized with the following steps:

1. Initialize

$$\alpha_i^{(0)} \leftarrow 0, \quad \forall i = 1, \cdots, n,$$
$$\theta_i^{(0)} \leftarrow 0, \quad \forall i = 1, \cdots, n,$$

   to ensure a pure isotropic cost function at step 0.

2. Set $\mathbf{x}$ equal to minimizer of a convex objective function minimize $\Psi(\mathbf{x}; \boldsymbol{\alpha}, \theta)$ subject to equality and inequality constraints. This cost function is formally defined in Section 3.4.3, equation (3.14), and it consists of a convex combination of an isotropic term and an anisotropic term at each pixel i. The process of minimizing this cost function follows the same process as the OG Algorithm, as described in Section 3.3.1.

3. For each pixel i, calculate the "dominant gradient direction" using a singular value decomposition of the local gradient vector matrix $A_i = [g_j]_{j \in \mathcal{N}_i}$. This produces singular values $\sigma_{i,1}, \sigma_{i,2}$ and left singular vectors $[\mathbf{v}_{i,1}, \mathbf{v}_{i,2}]$. Alternatively, this can also be viewed as taking an eigendecomposition of the matrix $A_i A_i^T$ of local gradient vectors, yielding eigenvalues $\lambda_{i,1} = \sigma_{i,1}^2$ and $\lambda_{i,2} = \sigma_{i,2}^2$. The eigenvectors are exactly the same left singular vectors obtained using the SVD.

4. Set the dominant gradient strengths to be

$$\alpha_i \leftarrow \begin{cases} \frac{\sigma_{i,1}^2 - \sigma_{i,2}^2}{\sigma_{i,1}^2}, & \sigma_{i,1} > 0 \\ 0, & \sigma_{i,1} = 0 \end{cases}$$

   and the dominant gradient directions to be

$$\theta_i \leftarrow \angle \mathbf{v}_{i,1}.$$

The initialization in Step 1 forces the cost function to be purely isotropic during the first iteration of the algorithm. Once this initial estimate is obtained, a dominant gradient direction θ_i and measure of neighboring gradient strength α_i can be calculated at every pixel i using operations on the spatial image gradient, producing singular values and singular vectors at each pixel (alternatively, eigenvalues and eigenvectors). We empirically observed that using gradients from *all previous image gradient estimates* when calculating the local gradient SVD (Step 3) caused the image estimates to converge, i.e., our convergence criterion $\|\mathbf{x}^{(k+1)} - \mathbf{x}^k\|/n_p < 10^{-2}$ was satisfied in all of our experiments. We describe how this is

done efficiently in Section 3.4.4. We were not able to prove this convergence analytically, and we cannot provide any theoretical guarantees of convergence. We briefly note that we could have forced convergence of $\boldsymbol{\alpha}$ and $\boldsymbol{\theta}$ if we followed the same process described in Section 3.3.2 for the OG Algorithm. Specifically, a "relaxation sequence" could have been used when updating $\boldsymbol{\alpha}$ and $\boldsymbol{\theta}$, which would have forced convergence of these parameters, and this would likely have further helped convergence of the image estimates, as we observed for the OG algorithm. Again, we found that we did not need to do this, since convergence was observed empirically without it.

As described in Section 3.4.1, the update for $\boldsymbol{\alpha}$ measures the strength of the dominant gradient direction at each pixel i. Dividing by σ_1^2 ensures that the weights α_i and $1-\alpha_i$ will sum to one. Thus, α_i gives a measure of how "anisotropic" the image gradient is near pixel i. If α_i is close to one, then there is a dominant gradient direction around pixel i. If α_i is close to zero, then the region around i is mostly smooth, or extremely noisy. The update on θ_i is simply the angle of the dominant gradient direction.

3.4.3 Cost Function for the LOOP Algorithm

For some fixed choice of $\boldsymbol{\alpha}$ and $\boldsymbol{\theta}$, satisfying

$$\begin{aligned} 0 \leq \alpha_i \leq 1, \quad \forall i = 1, \cdots, n, \\ -\pi < \theta_i \leq \pi, \quad \forall i = 1, \cdots, n, \end{aligned}$$

the second step of the LOOP algorithm solves the optimization problem

$$\underset{\mathbf{x}}{\text{minimize}} \quad \Psi(\mathbf{x}; \boldsymbol{\alpha}, \boldsymbol{\theta}) \quad \text{subject to} \quad S\mathbf{x} = \mathbf{y}, \quad x_{i,min} \leq x_i \leq x_{i,max}, \tag{3.14}$$

where S is a sampling matrix, $x_{i,min}$ and $x_{i,max}$ define box constraints on the max/min values attainable by the reconstruction, and Ψ is a linear combination of two convex cost functions

$$\Psi(\mathbf{x}; \boldsymbol{\alpha}, \boldsymbol{\theta}) = \Psi_{ISO}(\mathbf{x}; \boldsymbol{\alpha}) + \Psi_{ANISO}(\mathbf{x}; \boldsymbol{\alpha}, \boldsymbol{\theta}), \tag{3.15}$$

where $\Psi_{ISO}(\mathbf{x}; \boldsymbol{\alpha})$ and $\Psi_{ANISO}(\mathbf{x}; \boldsymbol{\alpha}, \boldsymbol{\theta})$ are isotropic and anisotropic cost functions, respectively. The anisotropic weighting vector $\boldsymbol{\alpha}$ forces Ψ to be a convex combination of Ψ_{ISO} and Ψ_{ANISO} at every pixel, and $\boldsymbol{\theta}$ determines the penalization directions of the anisotropic cost function Ψ_{ISO}. The box constraints $x_{i,min}$ and $x_{i,max}$ encourage the solution to have similar values to observed (nearby) samples. Specifically, for Manhattan sampling, we chose $x_{i,min}$ and $x_{i,max}$ to be the minimum and maximum observed image values on the "block border"

surrounding the unsampled data (referring to Figure 3.2b, these would be the sampled pixels with indices in $\overline{I}$).

We use the isotropic cost function

$$\Psi_{ISO}(\mathbf{x};\boldsymbol{\alpha}) = \frac{1}{2}\sum_{i=1}^{n}\frac{1-\alpha_i}{|\mathcal{N}_i|}\sum_{j\in\mathcal{N}_i}\frac{(x_i-x_j)^2}{d_{ij}^2}, \tag{3.16}$$

where $\mathcal{N}_i$ is the 8-point pixel neighborhood at pixel i, and d_{ij} is the distance in pixels between pixels i and j (either 1 or $\sqrt{2}$). This is a straightforward convex cost function that penalizes the average squared difference between neighboring pixels in the image.

For the anisotropic term, recall that the cost function in the OG algorithm was purely anisotropic because it only penalized pixel differences in the directions orthogonal to the image gradient. In a similar manner, we propose an anisotropic cost function that selectively penalizes orthogonal pixel differences using bicubic interpolation. Specifically, at every pixel i, we interpolate $\mathbf{x}$ on the unit circle centered at pixel i at angles $\theta_i + \frac{\pi}{2}$ and $\theta_i - \frac{\pi}{2}$ to obtain interpolated values $f_i\left(\mathbf{x}, \theta_i - \frac{\pi}{2}\right)$ and $f_i\left(\mathbf{x}, \theta_i + \frac{\pi}{2}\right)$. We then penalize the average squared pixel differences between x_i and these values. The motivation for this is that the intensity of each pixel j neighboring i will be weighted according to how close it is to the orthogonal dominant gradient directions $\theta_i \pm \frac{\pi}{2}$. This cost function can be written as

$$\Psi_{ANISO}(\mathbf{x};\boldsymbol{\alpha},\boldsymbol{\theta}) = \frac{1}{2}\sum_{i=1}^{n}\frac{\alpha_i}{2}\left[\left(x_i - f_i\left(\mathbf{x},\theta_i-\frac{\pi}{2}\right)\right)^2 + \left(x_i - f_i\left(\mathbf{x},\theta_i+\frac{\pi}{2}\right)\right)^2\right], \tag{3.17}$$

where $f_i(\mathbf{x},\theta_i)$ is the value obtained by interpolating $\mathbf{x}$ using bicubic interpolation on the unit circle centered at pixel i at angle θ_i, and the extra factor of $\frac{1}{2}$ averages the two values. Note that $f_i(\mathbf{x},\theta_i)$ can be calculated using a linear filter operation.

The cost function can be rewritten in matrix-vector form as

$$\Psi(\mathbf{x};\boldsymbol{\alpha},\theta) = \frac{1}{2}\left\|D\mathbf{x}\right\|^2_{W_{ISO}(\boldsymbol{\alpha})} + \frac{1}{2}\left\|C_{\boldsymbol{\theta}}\mathbf{x}\right\|^2_{W_{ANISO}(\boldsymbol{\alpha})},$$

where for some matrix W, we define the weighted squared 2-norm to be

$$\|\mathbf{x}\|^2_W = \mathbf{x}^T W\mathbf{x},$$

the isotropic weighting matrix is defined to be

$$W_{ISO}(\boldsymbol{\alpha}) = I_{8n_p \times 8n_p} - \underbrace{\begin{bmatrix} \text{diag}\{\alpha_i\}) & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \text{diag}\{\alpha_i\}) \end{bmatrix}}_{\text{Block diagonal, 8 copies of diag}\{\alpha_i\}},$$

the anisotropic weighting matrix is defined to be

$$W_{ANISO}(\boldsymbol{\alpha}) = \begin{bmatrix} \text{diag}\{\alpha_i\} & \mathbf{0} \\ \mathbf{0} & \text{diag}\{\alpha_i\} \end{bmatrix},$$

D is an isotropic differencing matrix, and $C_{\boldsymbol{\theta}}$ is an anisotropic differencing matrix that implements differences between each x_i and its bicubic interpolated values $f_i(\mathbf{x}, \theta_i - \frac{\pi}{2})$ and $f_i(\mathbf{x}, \theta_i + \frac{\pi}{2})$. Similar to the OG algorithm, it can be shown that this objective function is convex and can be minimized with gradient descent methods. This objective function is solved by following the process in Section 3.3.1 for the OG algorithm.

3.4.4 Efficient Computation of the Dominant Gradient Direction

In this section, we expand on Step 3 of the LOOP algorithm. Suppose we have an image $\mathbf{x}$ that is very high dimensional e.g. $\mathbf{x} \in \mathbb{R}^{512^2}$. Let $\mathbf{g}_x$ and $\mathbf{g}_y$ denote the spatial image gradient in the x- and y- directions, respectively (we abuse notation here by talking about the x- and y- directions). These spatial gradients $\mathbf{g}_x$ and $\mathbf{g}_y$ can be efficiently computed using classical filtering operations with a set of gradient filter kernels (e.g. the Sobel operators), either by spatial convolution or using FFTs. For each pixel i let $A_i \in \mathbb{R}^{2 \times m_1 m_2}$ denote the matrix where each column is a gradient vector in a rectangular $m_1 \times m_2$ window centered at pixel i. Recall that we defined the *dominant gradient direction at pixel i* [45] to be the unit vector $\mathbf{u}$ that maximizes $\|A_i^T \mathbf{u}\|^2$. As is well known, the solution to this problem is the eigenvector $\mathbf{u}$ corresponding to the largest eigenvalue of the (scaled) matrix $A_i A_i^T$. If we rebuild the gradient matrix for each pixel and use a built-in SVD solver, then it will take n_p^2 individual SVD computations. If this step is done naively using, say, Python's or Matlab's built-in SVD solver, the program could be very slow, especially for large images. In this section, we discuss using linear filtering operations and element-by-element array operations to reduce the amount of computation to a single, closed-form operation.

We begin by noting that $A_i A_i^T$ is a symmetric 2×2 matrix with entries

$$\begin{aligned} A_i A_i^T &= \begin{bmatrix} a_{i,xx} & a_{i,xy} \\ a_{i,xy} & a_{i,yy} \end{bmatrix} \\ &= \begin{bmatrix} \sum_{j \in \mathcal{N}_i} g_{x,j}^2 & \sum_{j \in \mathcal{N}_i} \sum_{k \in \mathcal{N}_i} g_{x,j} g_{y,k} \\ \sum_{j \in \mathcal{N}_i} \sum_{k \in \mathcal{N}_i} g_{x,j} g_{y,k} & \sum_{j \in \mathcal{N}_i} g_{y,j}^2 \end{bmatrix}, \end{aligned}$$

where $\mathcal{N}_i$ is an $m_1 \times m_2$ rectangular window centered at pixel i. Observe that entries of the matrix are simply the result of convolving $\mathbf{g}_x$ and $\mathbf{g}_y$ with an $m_1 \times m_2$ box kernel (e.g. "all ones"). Thus, we can calculate the entries of these matrices for all i using basic element-wise array operations and linear filtering operations. Let $\mathbf{1}_{m_1 \times m_2}$ be an $m_1 \times m_2$ matrix of all ones. Define the signals

$$\begin{aligned} \mathbf{a}_{xx} &= \mathbf{1}_{m_1 \times m_2} * (\mathbf{g}_x \odot \mathbf{g}_x), \\ \mathbf{a}_{xy} &= \mathbf{1}_{m_1 \times m_2} * (\mathbf{g}_x \odot \mathbf{g}_y), \\ \mathbf{a}_{yy} &= \mathbf{1}_{m_1 \times m_2} * (\mathbf{g}_y \odot \mathbf{g}_y), \end{aligned}$$

where $\odot$ denotes element-wise multiplication and $*$ denotes convolution.

We now show how Step 3 uses all previous image gradient estimates in its calculation of the dominant gradient direction. If we have completed k iterations of our algorithm, we can form $A_i A_i^T$ using a $k \times m_1 \times m_1$ neighborhood of all past image estimates by initializing $a_{xx}^{(0)} = a_{xy}^{(0)} = a_{yy}^{(0)} = \mathbf{0}$ and updating these variables according to

$$\begin{aligned} \mathbf{a}_{xx}^{(k)} &= \mathbf{a}_{xx}^{(k-1)} + \mathbf{1}_{m_1 \times m_2} * (\mathbf{g}_x^{(k)} \odot \mathbf{g}_x^{(k)}), \\ \mathbf{a}_{xy}^{(k)} &= \mathbf{a}_{xy}^{(k-1)} + \mathbf{1}_{m_1 \times m_2} * (\mathbf{g}_x^{(k)} \odot \mathbf{g}_y^{(k)}), \\ \mathbf{a}_{yy}^{(k)} &= \mathbf{a}_{yy}^{(k-1)} + \mathbf{1}_{m_1 \times m_2} * (\mathbf{g}_y^{(k)} \odot \mathbf{g}_y^{(k)}). \end{aligned}$$

For convenience, we will drop the dependence on k for the remainder of this section. Recall that a 2×2 eigenvalue problem can be solved in closed form using the quadratic formula. Following the usual procedure, but substituting element-wise operations for scalar operations, it can be shown that the largest and smallest eigenvalues of $A_i A_i^T$ can be calcu-

lated at every i according to

$$\boldsymbol{\lambda}_1 = \frac{\mathbf{a}_{xx} + \mathbf{a}_{yy} + \sqrt{(\mathbf{a}_{xx} + \mathbf{a}_{yy})^2 - 4(\mathbf{a}_{xx} \odot \mathbf{a}_{yy} - \mathbf{a}_{xy}^2)}}{2},$$
$$\boldsymbol{\lambda}_2 = \frac{\mathbf{a}_{xx} + \mathbf{a}_{yy} - \sqrt{(\mathbf{a}_{xx} + \mathbf{a}_{yy})^2 - 4(\mathbf{a}_{xx} \odot \mathbf{a}_{yy} - \mathbf{a}_{xy}^2)}}{2},$$

where $\boldsymbol{\lambda}_1$ is the image of largest eigenvalues and $\boldsymbol{\lambda}_2$ is the image of smallest eigenvalues. Furthermore, the x- and y- components of the largest eigenvector v_i at each pixel i can then be calculated according to

$$\mathbf{v}_x = \frac{\mathbf{a}_{xy}}{\sqrt{\mathbf{a}_{xy}^2 + (\boldsymbol{\lambda}_1 - \mathbf{a}_{xx})^2}},$$
$$\mathbf{v}_y = \frac{\boldsymbol{\lambda}_1 - \mathbf{a}_{xx}}{\sqrt{\mathbf{a}_{xy}^2 + (\boldsymbol{\lambda}_1 - \mathbf{a}_{xx})^2}}.$$

We also note that if the eigenvectors corresponding to the smallest eigenvalues at every pixel i are needed, then they can be calculated from $\mathbf{v}_x$ and $\mathbf{v}_y$ using a simple rotation operation. We thus can estimate the direction of the dominant gradient direction by taking

$$\boldsymbol{\theta} = \arctan 2(\mathbf{v}_y, \mathbf{v}_x),$$

where $\text{arctan2}(y, x)$ is the usual two-argument arctan function applied element-wise to $\mathbf{v}_y$ and $\mathbf{v}_x$ (unlike the single-argument arctan function, the double-argument arctan2 function retains sign information about an angle in four quadrants by returning an angle in $[-\pi, \pi)$).

The entire LOOP algorithm is summarized step-by-step in Figure 3.10. Note that for an image with n_p pixels, this algorithm only requires storage of $6n_p$ parameters: The current image estimate, the matrix entry variables $\mathbf{a}_{xx}$, $\mathbf{a}_{xy}$ and $\mathbf{a}_{yy}$, and the cost function parameters $\boldsymbol{\theta}$ and $\boldsymbol{\alpha}$.

3.5 Reconstruction Method Comparisons

In this section, we compare the performance of four algorithms in the task of estimating natural images from their Manhattan-grid samples. Specifically, we compare the performance of the OG Algorithm, the LOOP Algorithm, a kernel regression algorithm for inpainting [10], and a constrained Total Variation (TV) minimization algorithm [9].

The OG and LOOP algorithms were implemented in Python using standard scientific

computing libraries, i.e., `numpy` and `scipy`. The OG algorithm took 104 seconds to interpolate the image "Al" at 7×7 sampling on an Intel Core i5 CPU at 3.40 GHz. Both algorithms take longer for larger grid sizes, as fewer samples are available, and it takes time for information to propagate into the interior of each unsampled block of pixels. Similar performance was observed for the LOOP algorithm.

The TV inpainting "Missing Pixels: Image Inpainting" approach in [9, Sec. III.C.3] was used as another baseline algorithm using MATLAB code provided on the authors' website [47]. Specifically, we modified the file `demos/Constrained/demo_inpainting.m` by changing the random sampling pattern to the appropriate Manhattan grid size. This TV method took about 8 seconds to reconstruct "Al" at 7×7 sampling.

The kernel regression inpainting algorithm used was the "Iterative Steering Kernel Regression" approach in [10, Sec. III.C.] was implemented using code on the authors' website [48]. Specifically, we modified the file `Examples/Lena_irregular.m` by changing the random sampling pattern to the appropriate Manhattan grid size. We also added a single thresholding step where the intensity of the output image was truncated the interval $[0, 255]$. This kernel regression method took about 30 seconds to reconstruct "Al" at 7×7 sampling.

We ran our interpolation experiments on six images for two different sampling patterns. The images included "Al," "baboon," "Barbara," "boat," "peppers," and "tools," and the images were sampled on 4×3 and 7×8 Manhattan grids. These sampling patterns correspond to keeping 50% and 25% of the original data.

Once again, we used mean-squared error (specifically, PSNR) as our object performance metric. Our PSNR results are shown in Table 3.2. Note that the LOOP algorithm outperforms all other algorithms in each experiment, and the OG algorithm outperforms the TV and Kernel Regression methods for most experiments.

For subjective quality comparisons, Figures 3.11, 3.12, 3.13, 3.14, 3.15, and 3.16 show the results of interpolation for 7×8 Manhattan sampling on the six test images. Specifically, they show the original image, the sampled image, and the resulting reconstruction for each of the four algorithms. We do not show the results for 4×3 sampling since all four methods performed reasonably well, and visual differences are difficult to notice.

The outputs of the OG algorithm had very smooth and crisp edges. For example, in "Al", note the nice reconstructions of his collar and the books in the upper right corner. By contrast, periodic blocking artifacts are generated by all the competing algorithms in these two regions. Although the OG algorithm performs very well at reconstructing edges, it can also generate false contouring artifacts in regions of uniform intensity. Examples of such false contours are the "swirls" in the forehead and cheeks of "Al." These artifacts are reduced greatly by the LOOP algorithm, which attempts to use an isotropic cost function

in smoother regions.

The kernel regression algorithm struggled greatly with the Manhattan sampling pattern. Typically, traditional inpainting methods rely on sampling patterns that are "locally dense," or even regions of the image that are completely sampled in order to perform well. "Gaps" can be seen in the kernel regression reconstructions where the kernels failed to estimate the image intensity, most likely due to having "small" kernels at samples in the area. An example of these gaps can be seen in the reconstruction of "Al" in Figure 3.11(d) on the left and right sides of the image, where the edge of the image meets the white empty space. These are also seen along the long pepper in peppers (Figure 3.15(d)).

The "boat" results (Figure 3.14) are very striking; the LOOP algorithm manages to reconstruct the boat "poles," whereas all other methods do not come close.

Nearly all the methods struggle with high-frequency texture, like the fur in "baboon" (Figure 3.12 and the cloth in "Barbara" (Figure 3.13).

Finally, in Figures 3.17 and 3.18, we can see the final values of $\boldsymbol{\alpha}$ for the 7×8 LOOP reconstructions. These are a good visual tool to see when the LOOP algorithm used an isotropic objective function, and when it used an anisotropic objective function. Specifically, in the regions where $\boldsymbol{\alpha}$ is black (zero), Ψ_{iso} was weighted more heavily in the cost function. In the regions where $\boldsymbol{\alpha}$ is white (one), Ψ_{aniso} was weighted more heavily in the cost function. Note that $\boldsymbol{\alpha} \approx 1$ near regions with texture or strong image edges, which shows that our algorithm is working as intended.

Finally, it is easy to visually check that the subjective quality of the LOOP algorithm is much better than the OG, Piecewise Planar, or Cutset MRF methods for reconstruction of images from their Manhattan-grid samples. The "crevasse" artifacts from the Piecewise-Planar method have been eliminated, as well as the "oversharpening" caused by the segmentation step of the Cutset MRF method. It is easy to see this by comparing "Al" in Figure 3.11 with "Al" in Figure 3.6. In fact, even at a slightly lower density, the LOOP algorithm results at 7×8 sampling outperform the 7×7 Piecewise-Planar and Cutset MRF reconstructions.

3.5.1 Traditional Lattice Sampling Experiments

A natural question is to ask how the LOOP algorithm performs when used when interpolating from lattice samples. Thus, we ran a standard 2-factor downsample/upsample scenario where 25% of the pixels are saved on an equispaced lattice (every other pixel is sampled in each direction).

We compared the LOOP algorithm to traditional bicubic kernel interpolation, as well as a

4x3 Manhattan Grid	TV [9]	Kern. Reg. [10]	OG	LOOP
Al	34.5	31.4	35.2	**35.9**
Baboon	24.4	22.0	24.5	**26.3**
Barbara	25.0	24.0	24.7	**26.1**
Boat	30.6	28.1	31.3	**32.9**
Peppers	34.9	31.6	35.5	**36.1**
Tools	37.1	34.2	38.9	**39.3**
7x8 Manhattan Grid	TV [9]	Kern. Reg. [10]	OG	LOOP
Al	27.1	24.4	29.7	**30.6**
Baboon	21.0	20.4	20.6	**22.1**
Barbara	23.7	22.7	23.0	**24.8**
Boat	25.0	25.9	26.8	**27.7**
Peppers	28.7	28.4	30.7	**31.3**
Tools	27.6	28.5	31.2	**31.7**

Table 3.2: Comparison of PSNR values (dB) for various methods. Highest PSNR for each row is in bold.

recent interpolation algorithm [11] that was state-of-the-art in 2008. Bicubic results were obtained using Matlab's `imresize` command, and the results were thresholded to $[0, 255]$. The SAI algorithm [11] (**S**oft-decision estimation technique for **A**daptive image **I**nterpolation) also attempts to interpolate orthogonal to image edges. The experiment was performed using code available on the authors' website [49]. SAI ran in less than one second for our images.

Implementation details for the LOOP algorithm include the following: First $\mathbf{x}_{min}$ and $\mathbf{x}_{max}$ were defined at each pixel i using the minimum and maximum of nearest-neighbor sampled values. Specifically, a "middle" pixel used the minimum and maximum of the four surrounding sampled values, located northeast, northwest, southeast, and southwest of the pixel. Similarly, a "side" unsampled pixel used the minimum and maximum of two flanking sampled values (i.e. either the samples to the east and west, or the samples to the north and south). A window of $m_1 \times m_2 = 7 \times 7$ was used for the local SVD computations. The initial estimate $\mathbf{x}^{(0)}$ was initialized to mean of $\mathbf{x}^{(0)} = \frac{1}{2}(\mathbf{x}_{min} + \mathbf{x}_{max})$.

PSNR results are shown in Table 3.3. Note that the LOOP algorithm performs much better than bicubic, and is competitive with SAI. Subjectively, the images look very similar. However, one difference is that the LOOP algorithm denoises slightly better than SAI. For example, in "Al" (Figure 3.19(d)), the forehead of Al and the books in the upper-left corner have some noisy "ripples" in the SAI reconstruction. These are also present in the LOOP reconstruction (Figure 3.19(f)), but are much less noticeable. This is likely due to using an isotropic objective function in these regions, as supported by the dark values of $\boldsymbol{\alpha}$ in Figure

2×2 Lattice Sampling	bicubic	SAI [11]	LOOP
Al	27.2	**30.3**	29.1
Baboon	21.3	22.7	**23.0**
Barbara	23.3	23.6	**24.3**
Boat	26.9	**29.7**	29.4
Peppers	28.1	**31.1**	**31.1**
Tools	30.2	**33.5**	30.7

Table 3.3: Comparison of PSNR values (dB) for lattice interpolation methods. Highest PSNR for each row is in bold.

3.19(e).

3.5.2 Manhattan vs. Lattice Comparison

Observe that 7×8 Manhattan sampling and 2×2 lattice both have a sampling density of 25%. In this section, we compare the results of reconstructing an image from its 7×8 Manhattan samples to those obtained from 2×2 lattice samples. In particular, we determine if there is a difference in PSNR, as well as in subjective quality. We pay particular attention to how well edges are reconstructed by each method, including both hard and soft image edges. Recall that we hypothesized in Chapter 1 that edges may be better reconstructed from Manhattan cutset samples.

3.5.2.1 PSNR Comparison

Let us compare the PSNR results for 7×8 Manhattan sampling in Table 3.2 to the PSNR results for 2×2 lattice sampling in Table 3.3. Surprisingly, the 7×8 Manhattan reconstructions for "Al," "Barbara," and "Peppers" using the LOOP algorithm have higher PSNR than any reconstructions obtained using 2×2 lattice samples. We do note, however, that the differences between these PSNRs and the best results for their 2×2 lattice counterparts were only less than 1 dB. For the other images, the 7×8 Manhattan LOOP reconstructions had PSNRs within 2 dB of the best 2×2 lattice reconstructions.

3.5.2.2 Subjective Comparison

We now subjectively compare bicubic and SAI reconstructions from samples on a 2×2 lattice to OG and LOOP reconstructions from samples on a 7×8 Manhattan grid. In particular, we will consider regions of the image that contain edges to see if cutset sampling provides an advantage to reconstructing edges over lattice sampling.

Figure 3.25 shows a zoomed-in region of Al's collar in "Al" for the original image and the four reconstructions. All four methods do very well in the smooth regions of the collar, but the LOOP algorithm reconstructs the least amount of texture, i.e., it denoises slightly better in these regions. Along the edge of the collar, the bicubic reconstruction fails, leaving a jaggy edge. However, the three other reconstruction methods reproduce the edge of the collar very well, except for near the boundary of the image, where the OG and LOOP algorithms blur due to boundary effects. Between these three algorithms, the reconstructed collars look very similar.

Figure 3.26 shows a zoomed-in region of the books in the upper-right corner of "Al." All four algorithms reconstruct the books well. The OG and LOOP algorithms produce smoother reconstructions, whereas the bicubic and SAI reconstructions introduce some textured noise between the books.

Figure 3.27 shows a zoomed-in region of some poles on the top of the boat in "Boat." All four algorithms reconstructed the poles differently: Bicubic interpolation created "jaggy" artifacts, SAI did fairly well overall, but introduced some "streaky" artifacts, the OG algorithm looks "painted", and the LOOP algorithm introduced some periodic blurring along the diagonal poles. Note that the SAI algorithm reconstructed the two large diagonal poles the best, whereas the LOOP algorithm reconstructed the vertical pole in the middle the best. This behavior is not surprising, since we found in Chapter 2 that Manhattan sampling is well-suited for recovering vertical and horizontal edge (high frequency) information from the original image, but not for recovering diagonal edge information.

3.5.2.3 Manhattan vs. Lattice Conclusions

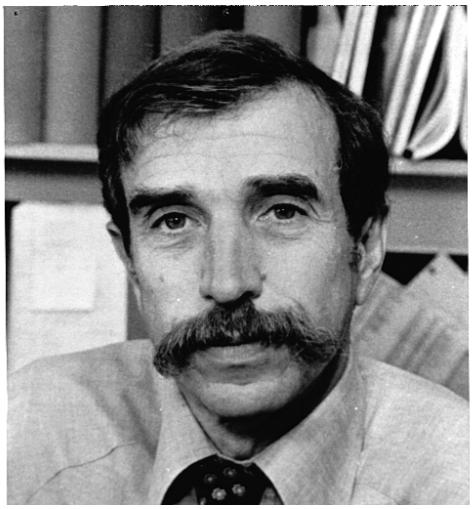
When comparing reconstruction methods from 2×2 lattice sampling to reconstruction methods from 7×8 Manhattan sampling, we found that the two sampling methods were competitive in PSNR values. In particular, we found that the best reconstructions from each sampling pattern were within 2 dB of one another. In terms of subjective quality, all four methods reconstruct smooth images well, but the LOOP algorithm tends to remove texture from smooth regions more than the other methods. When reconstructing edges, bicubic interpolation fails as expected, but the SAI, OG, and LOOP algorithms perform similarly. Thus, our conjecture that we can better reconstruct edges from Manhattan samples is false in this case. However, the similar performance of the two sampling/reconstruction methods is encouraging, and warrants more research into cutset sampling.

3.5.3 Conclusions

In this chapter, we presented a previous method and three new methods for reconstructing images from their Manhattan-grid samples. However, the LOOP algorithm outperforms them all, both in PSNR and visual quality. The LOOP algorithm also outperforms a TV minimization algorithm, and a kernel regression inpainting algorithm. It also had competitive results with the SAI algorithm for lattice-sampled images, which is an algorithm specifically designed for lattice interpolation. Overall, we believe that the LOOP algorithm is a promising new method for reconstructing image data from arbitrary sampling patterns. Its strength lies in its ability to adapt to the edges in the image using the output of the local gradient SVD computation.

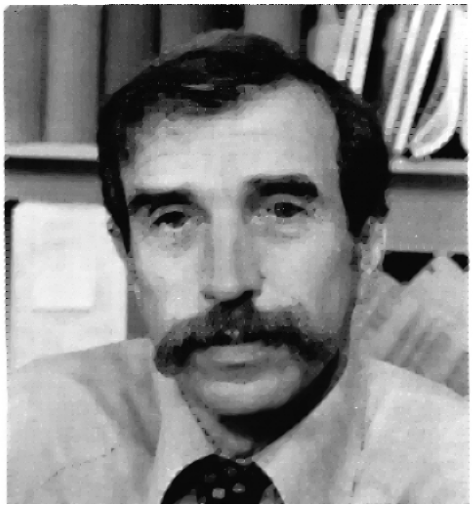
Input: Sampling matrix S, samples $\mathbf{y}$, min/max constraints $\mathbf{x}_{min}$ and $\mathbf{x}_{max}$,
initial condition $\mathbf{x}^{(0)}$, gradient filter kernels $\mathbf{h}_x$ and $\mathbf{h}_y$, local window size $m_1 \times m_2$.
Initialize: $k = 1$, $\boldsymbol{\alpha}^{(0)} = \mathbf{0}$, $\boldsymbol{\theta}^{(0)} = \mathbf{0}$, and $\mathbf{a}_{xx}^{(0)} = \mathbf{a}_{xy}^{(0)} = \mathbf{a}_{yy}^{(0)} = \mathbf{0}$.
repeat
  Set $\mathbf{x}^{(k)}$ to solution of optimization problem (3.14) using
    previous weights $\boldsymbol{\alpha}^{(k-1)}$ and previous dominant gradient directions $\boldsymbol{\theta}^{(k-1)}$,
    under constraints determined by S, $\mathbf{y}$, $\mathbf{x}_{min}$, and $\mathbf{x}_{max}$.
  $\mathbf{g}_x^{(k)} = \mathbf{x}^{(k)} * \mathbf{h}_x$
  $\mathbf{g}_y^{(k)} = \mathbf{x}^{(k)} * \mathbf{h}_y$
  $\mathbf{a}_{xx}^{(k)} \leftarrow \mathbf{a}_{xx}^{(k-1)} + \mathbf{1}_{m_1 \times m_2} * \left(\mathbf{g}_x^{(k)} \odot \mathbf{g}_x^{(k)} \right)$
  $\mathbf{a}_{xy}^{(k)} \leftarrow \mathbf{a}_{xy}^{(k-1)} + \mathbf{1}_{m_1 \times m_2} * \left(\mathbf{g}_x^{(k)} \odot \mathbf{g}_y^{(k)} \right)$
  $\mathbf{a}_{yy}^{(k)} \leftarrow \mathbf{a}_{yy}^{(k-1)} + \mathbf{1}_{m_1 \times m_2} * \left(\mathbf{g}_y^{(k)} \odot \mathbf{g}_y^{(k)} \right)$
  $\boldsymbol{\lambda}_1^{(k)} \leftarrow \frac{1}{2} \left[\mathbf{a}_{xx}^{(k)} + \mathbf{a}_{yy}^{(k)} + \sqrt{\left(\mathbf{a}_{xx}^{(k)} + \mathbf{a}_{yy}^{(k)} \right)^2 - 4 \left(\mathbf{a}_{xx}^{(k)} \odot \mathbf{a}_{yy}^{(k)} - {\mathbf{a}_{xy}^{(k)}}^2 \right)} \right]$
  $\boldsymbol{\lambda}_2^{(k)} \leftarrow \frac{1}{2} \left[\mathbf{a}_{xx}^{(k)} + \mathbf{a}_{yy}^{(k)} - \sqrt{\left(\mathbf{a}_{xx}^{(k)} + \mathbf{a}_{yy}^{(k)} \right)^2 - 4 \left(\mathbf{a}_{xx}^{(k)} \odot \mathbf{a}_{yy}^{(k)} - {\mathbf{a}_{xy}^{(k)}}^2 \right)} \right]$
  $\mathbf{v}_x^{(k)} \leftarrow \frac{\mathbf{a}_{xy}^{(k)}}{\sqrt{\left(\mathbf{a}_{xy}^{(k)} \right)^2 + \left(\boldsymbol{\lambda}_1^{(k)} - \mathbf{a}_{xx}^{(k)} \right)^2}}$
  $\mathbf{v}_y^{(k)} \leftarrow \frac{\boldsymbol{\lambda}_1^{(k)} - \mathbf{a}_{xx}^{(k)}}{\sqrt{\left(\mathbf{a}_{xy}^{(k)} \right)^2 + \left(\boldsymbol{\lambda}_1^{(k)} - \mathbf{a}_{xx}^{(k)} \right)^2}}$
  $\boldsymbol{\theta}^{(k)} \leftarrow \arctan 2 \left(\mathbf{v}_y^{(k)}, \mathbf{v}_x^{(k)} \right)$
  $[\boldsymbol{\alpha}^{(k)}]_i \leftarrow \begin{cases} \frac{\lambda_{1,i}^{(k)} - \lambda_{2,i}^{(k)}}{\lambda_{1,i}^{(k)}}, & \lambda_{1,i} > 0 \\ 0, & \lambda_{1,i} = 0 \end{cases}$
  $k \leftarrow k + 1$
until stopping criterion is satisfied.

Figure 3.10: Local Orthogonal Orientation Penalization (LOOP) Algorithm

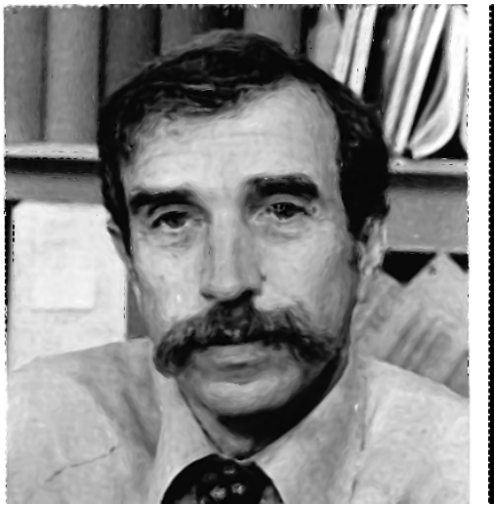


(a) "Al"

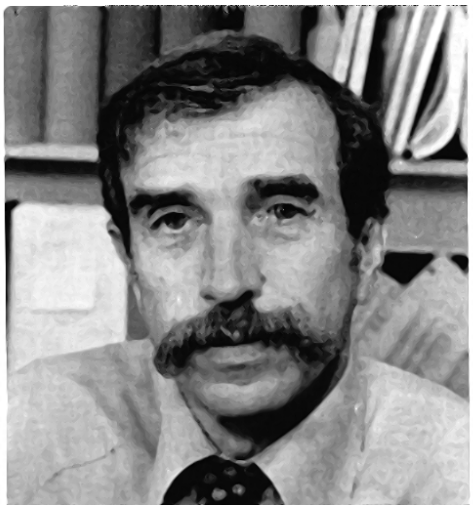
(b) 7×8 Manhattan sampling



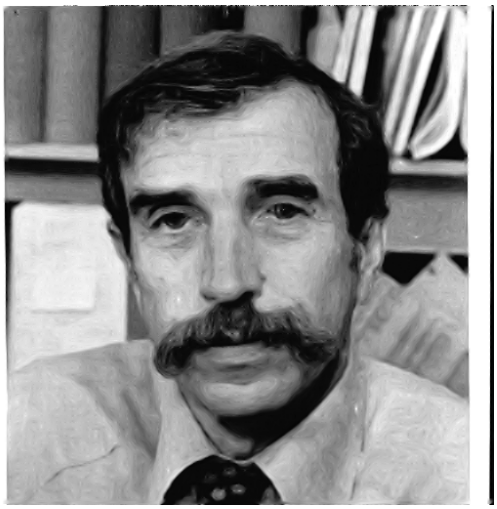
(c) SALSA Const. TV [9]



(d) Kernel Regression [10]

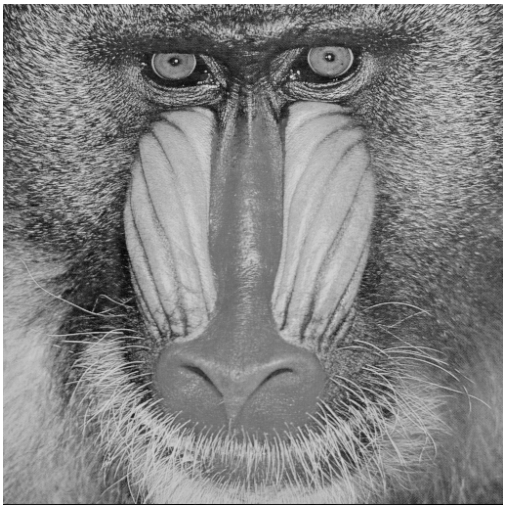


(e) Orthogonal Gradient Algorithm

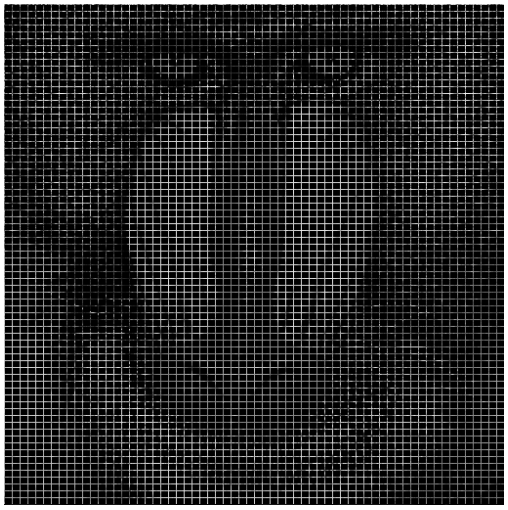


(f) LOOP Algorithm

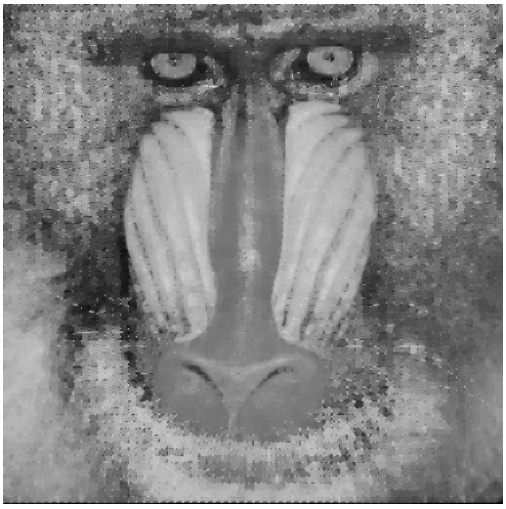
Figure 3.11: Method comparison for reconstructing "Al" from 7×8 Manh. sampling.



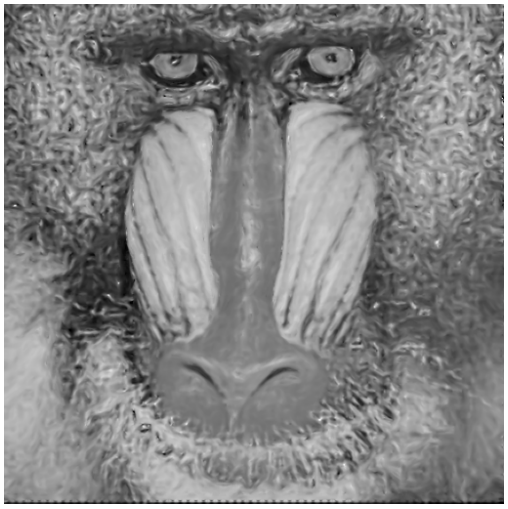
(a) "baboon"



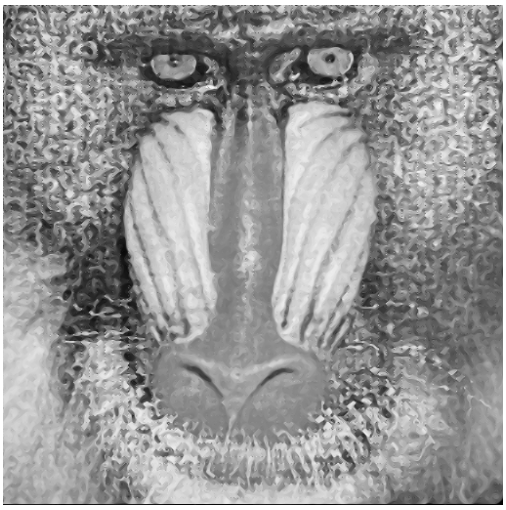
(b) 7×8 Manhattan sampling



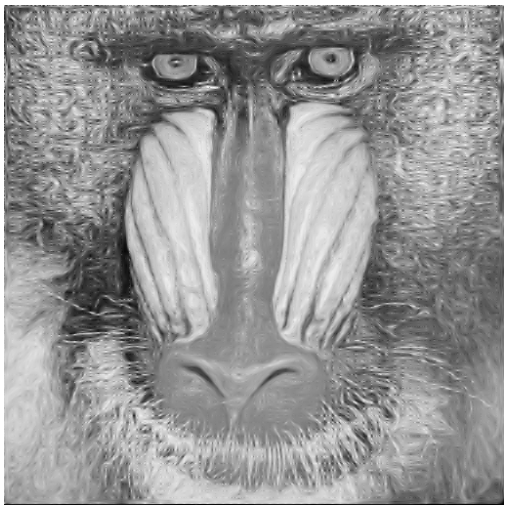
(c) SALSA Const. TV [9]



(d) Kernel Regression [10]



(e) Orthogonal Gradient Algorithm



(f) LOOP Algorithm

Figure 3.12: Method comparison for reconstructing "baboon" from 7×8 Manh. sampling.

(a) "Barbara"

(b) 7×8 Manhattan sampling

(c) SALSA Const. TV [9]

(d) Kernel Regression [10]

(e) Orthogonal Gradient Algorithm

(f) LOOP Algorithm

Figure 3.13: Method comparison for reconstructing "Barbara" from 7×8 Manh. sampling.

(a) "boat"

(b) 7×8 Manhattan sampling

(c) SALSA Const. TV [9]

(d) Kernel Regression [10]

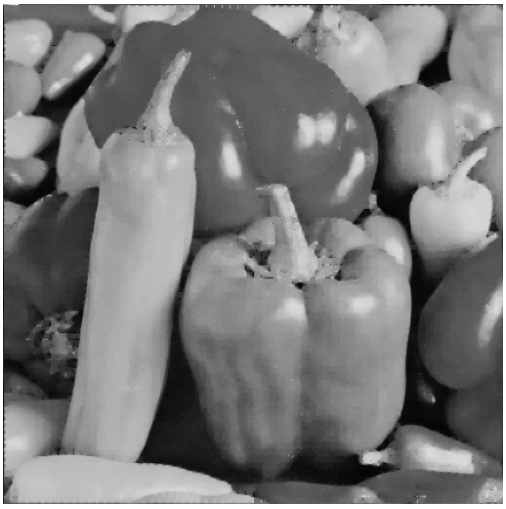
(e) Orthogonal Gradient Algorithm

(f) LOOP Algorithm

Figure 3.14: Method comparison for reconstructing "boat" from 7×8 Manh. sampling.

(a) “peppers”

(b) 7×8 Manhattan sampling



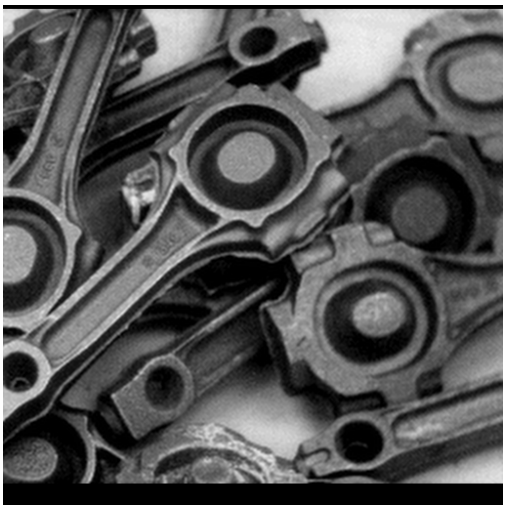
(c) SALSA Const. TV [9]

(d) Kernel Regression [10]

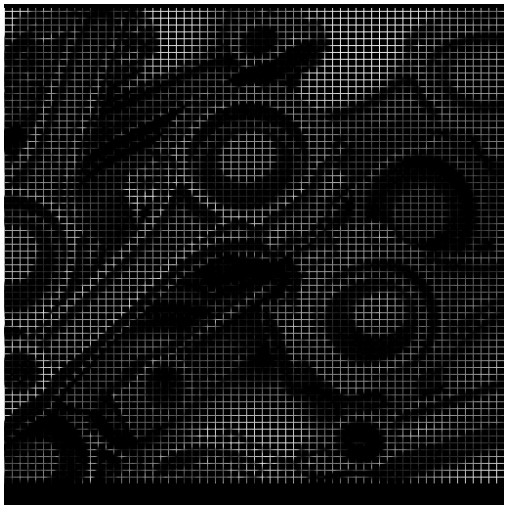
(e) Orthogonal Gradient Algorithm

(f) LOOP Algorithm

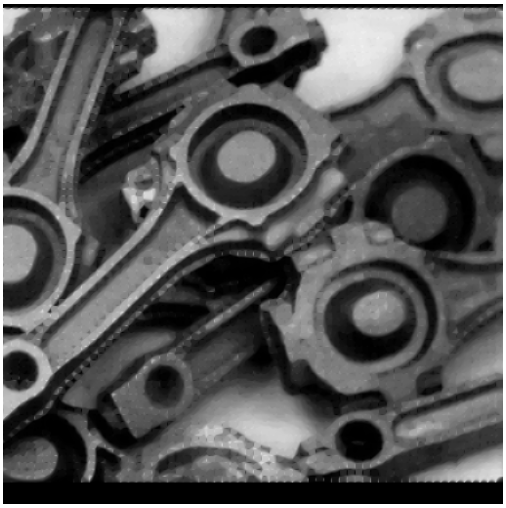
Figure 3.15: Method comparison for reconstructing “peppers” from 7×8 Manh. sampling.



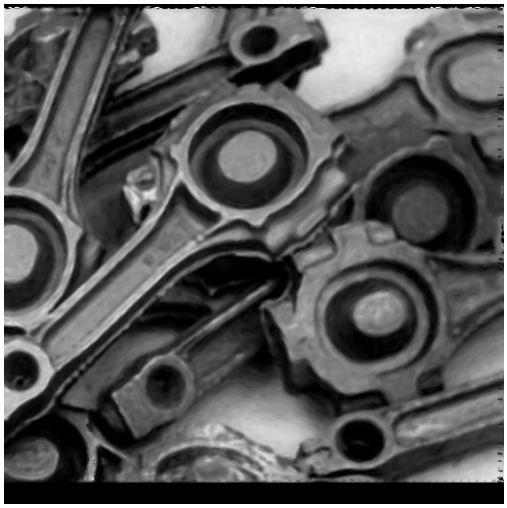
(a) “tools”



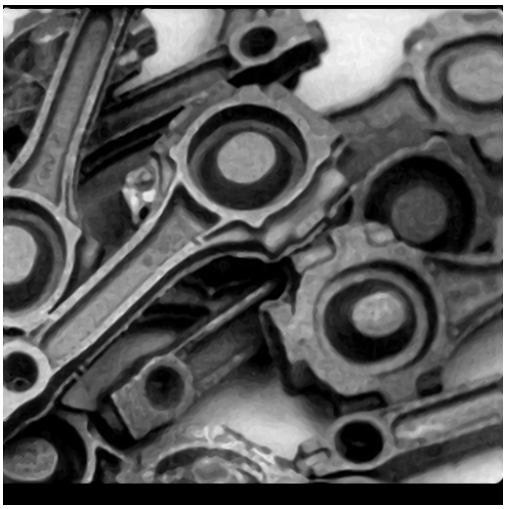
(b) 7×8 Manhattan sampling



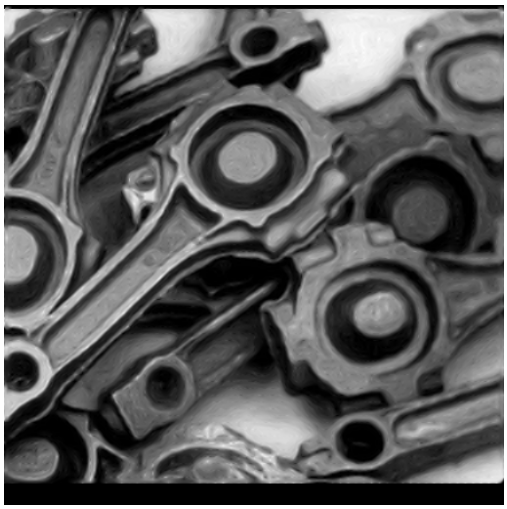
(c) SALSA Const. TV [9]



(d) Kernel Regression [10]

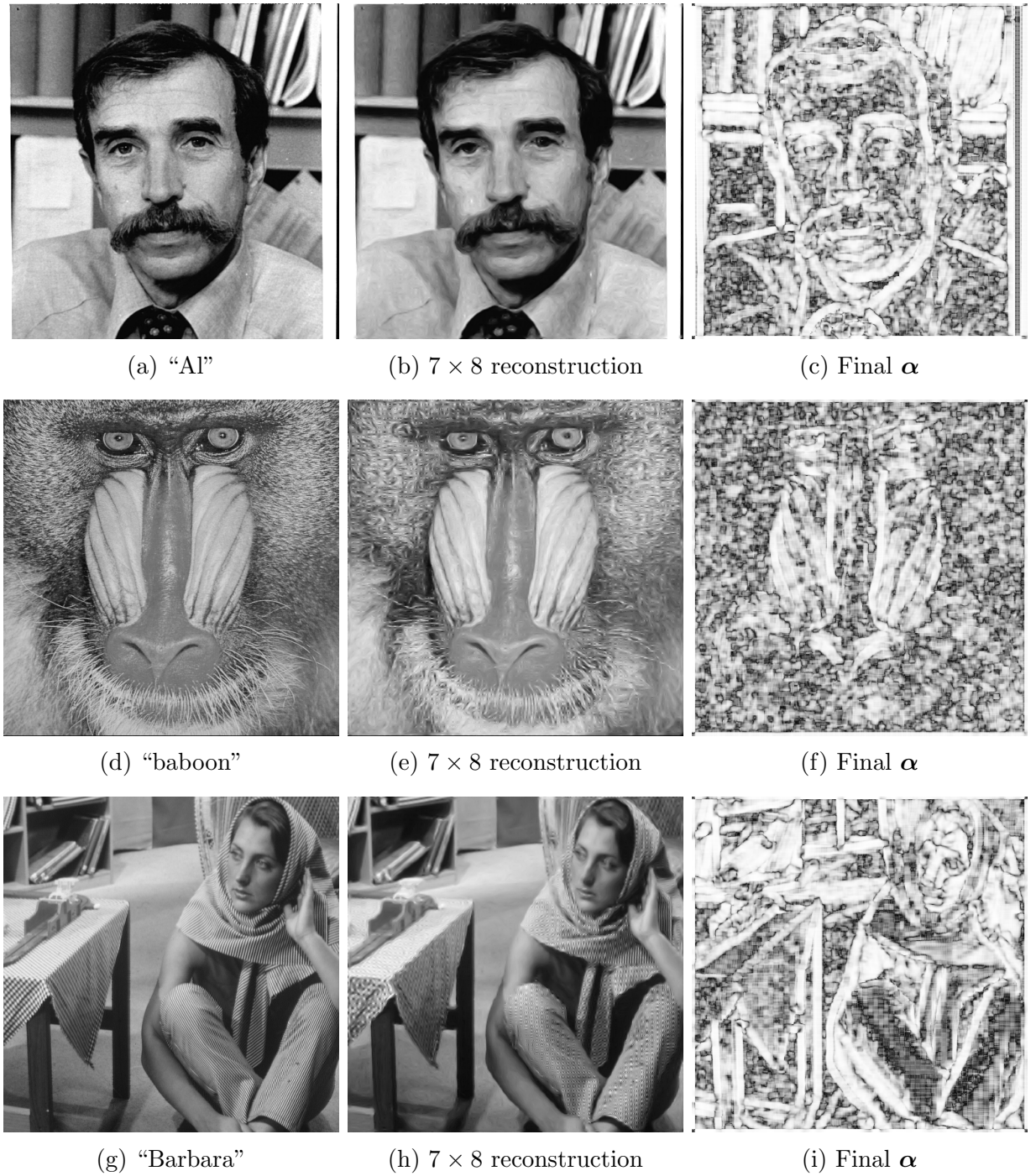


(e) Orthogonal Gradient Algorithm



(f) LOOP Algorithm

Figure 3.16: Method comparison for reconstructing “tools” from 7×8 Manh. sampling.



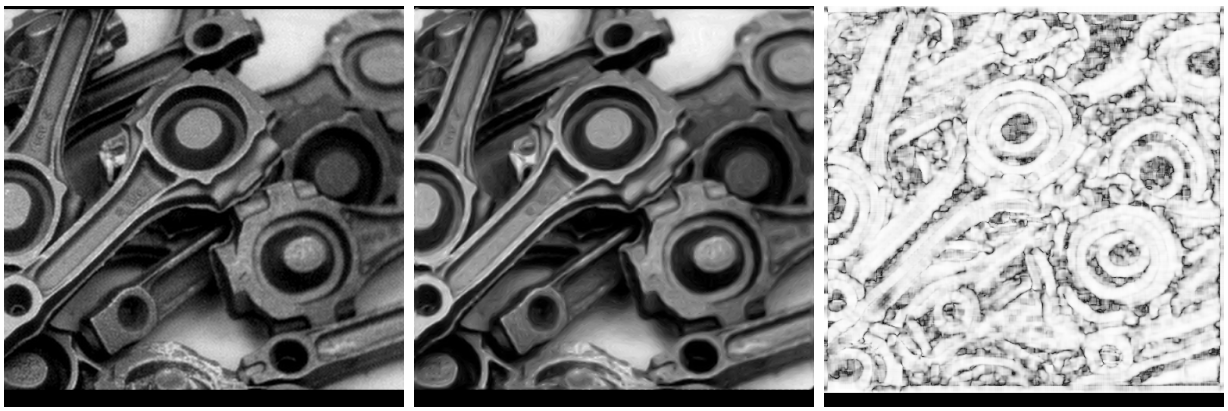
(a) "Al" (b) 7×8 reconstruction (c) Final $\boldsymbol{\alpha}$

(d) "baboon" (e) 7×8 reconstruction (f) Final $\boldsymbol{\alpha}$

(g) "Barbara" (h) 7×8 reconstruction (i) Final $\boldsymbol{\alpha}$

Figure 3.17: Part 1: Comparison of final $\boldsymbol{\alpha}$ parameter to reconstructions. $\alpha_i \approx 0$ in black regions and $\alpha_i \approx 1$ in white regions.

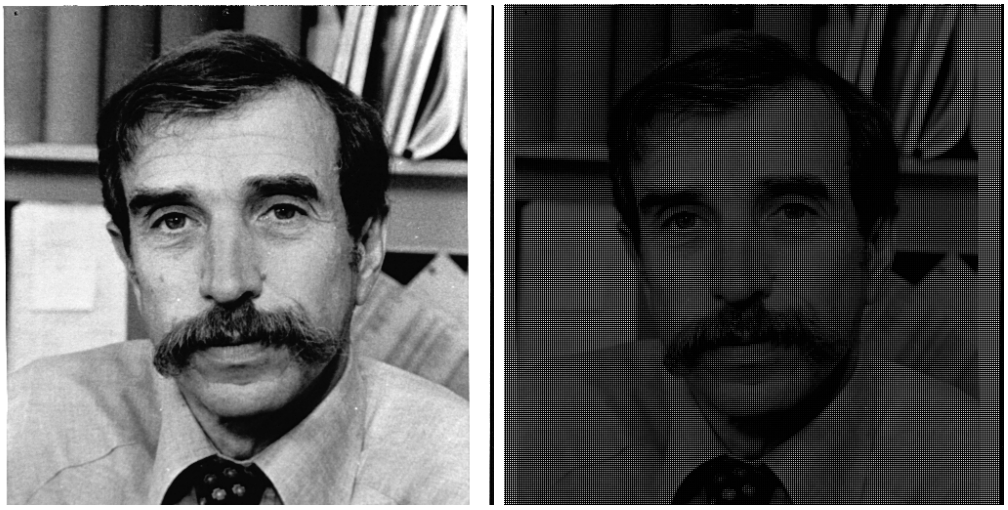
(a) "boat" (b) 7×8 reconstruction (c) Final $\boldsymbol{\alpha}$



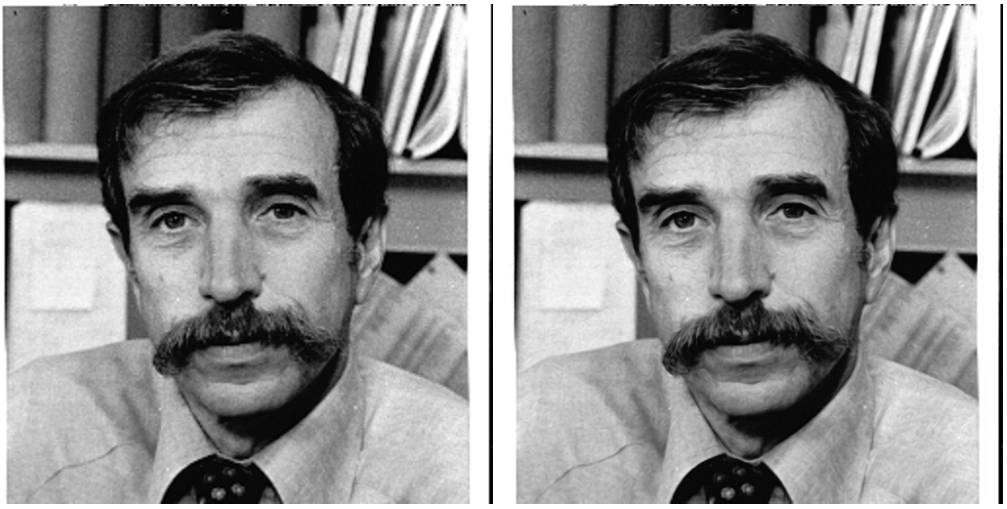
(d) "tools" (e) 7×8 reconstruction (f) Final $\boldsymbol{\alpha}$

(g) "peppers" (h) 7×8 reconstruction (i) Final $\boldsymbol{\alpha}$

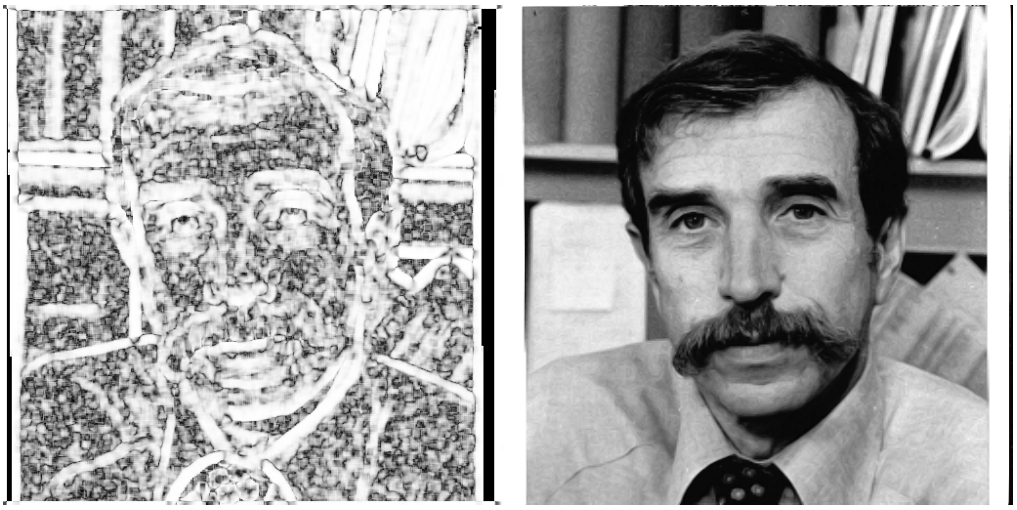
Figure 3.18: Part 2: Comparison of final $\boldsymbol{\alpha}$ parameter to reconstructions. $\alpha_i \approx 0$ in black regions and $\alpha_i \approx 1$ in white regions.



(a) "Al" (b) 2×2 lattice downsampling

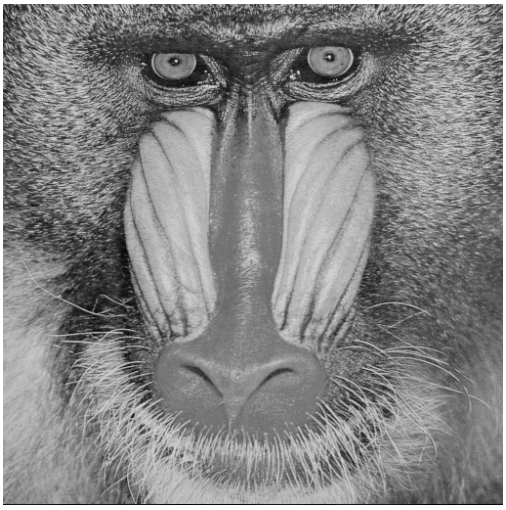


(c) Bicubic interpolation (d) SAI [11]

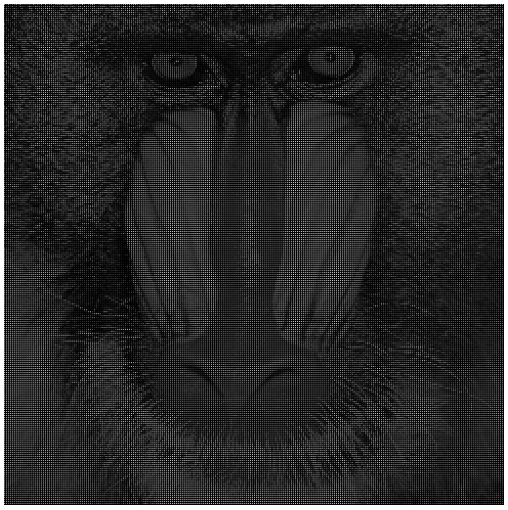


(e) Final $\boldsymbol{\alpha}$ for LOOP Algorithm (f) LOOP Algorithm

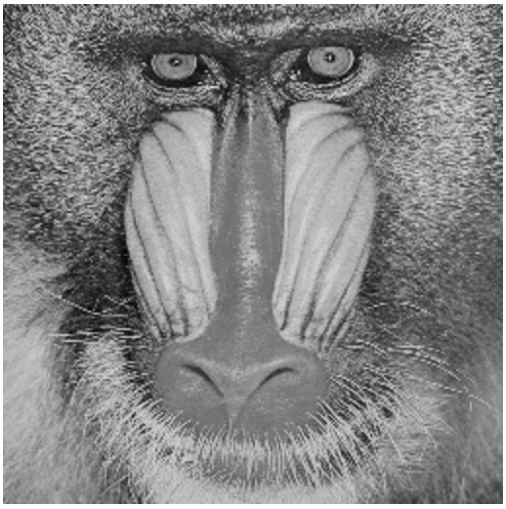
Figure 3.19: Method comparison for reconstructing "Al" from 2×2 lattice samples.



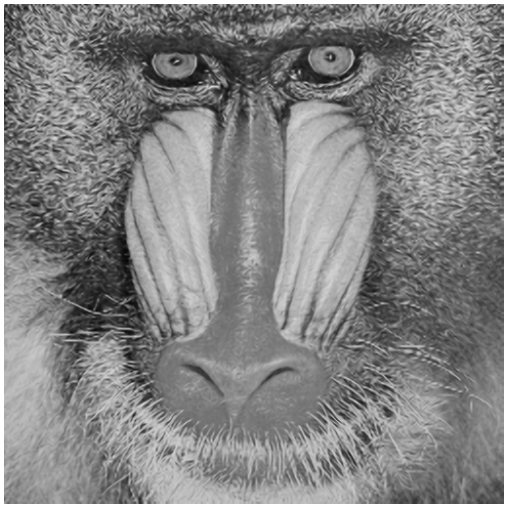
(a) "baboon"



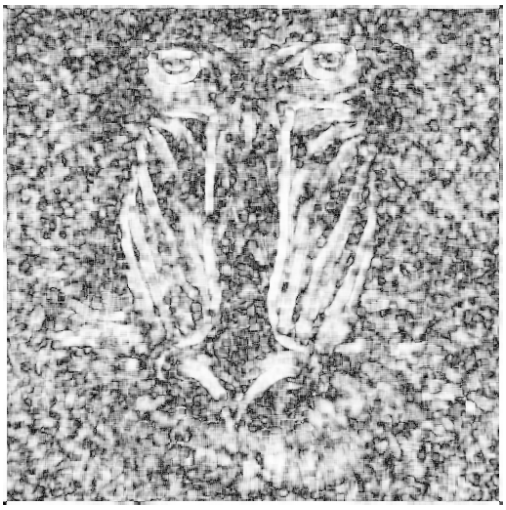
(b) 2×2 lattice downsampling



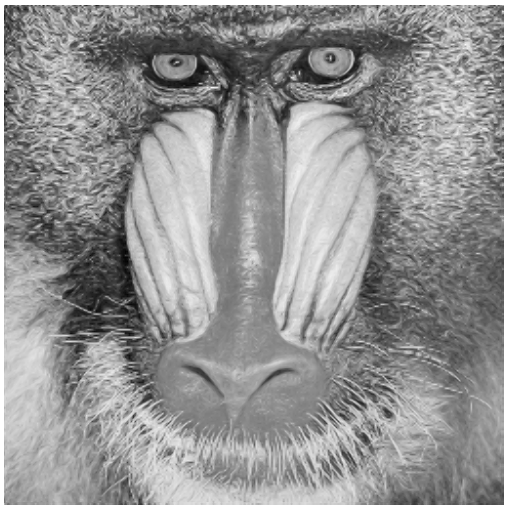
(c) Bicubic interpolation



(d) SAI [11]



(e) Final $\boldsymbol{\alpha}$ for LOOP Algorithm



(f) LOOP Algorithm

Figure 3.20: Method comparison for reconstructing "baboon" from 2×2 lattice samples.

(a) "Barbara"

(b) 2×2 lattice downsampling

(c) Bicubic interpolation

(d) SAI [11]

(e) Final $\boldsymbol{\alpha}$ for LOOP Algorithm

(f) LOOP Algorithm

Figure 3.21: Method comparison for reconstructing "Barbara" from 2×2 lattice samples.

(a) "boat"

(b) 2×2 lattice downsampling

(c) Bicubic interpolation

(d) SAI [11]

(e) Final $\boldsymbol{\alpha}$ for LOOP Algorithm

(f) LOOP Algorithm

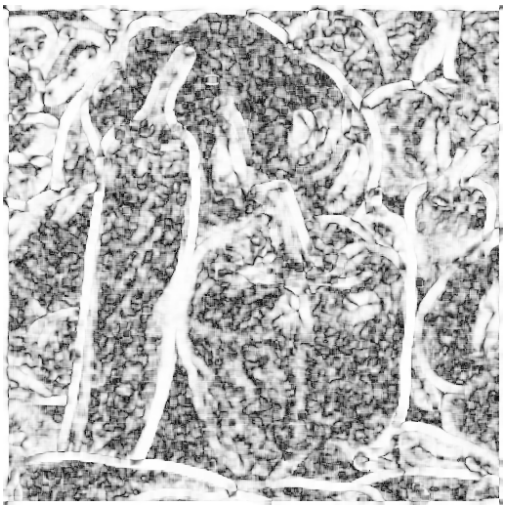
Figure 3.22: Method comparison for reconstructing "boat" from 2×2 lattice samples.

(a) “tools”

(b) 2×2 lattice downsampling

(c) Bicubic interpolation

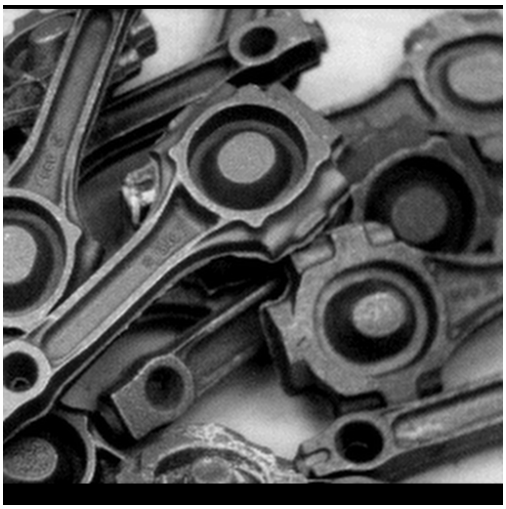
(d) SAI [11]



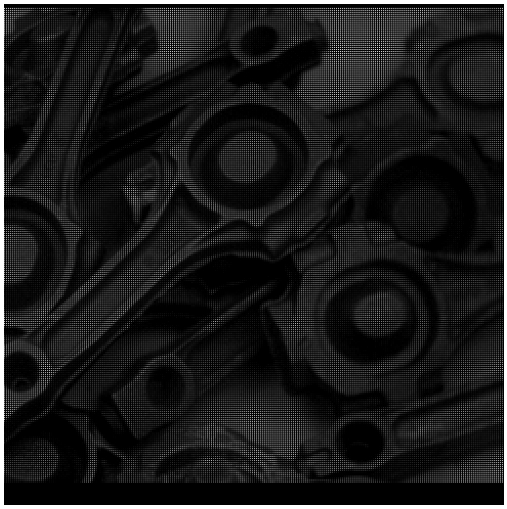
(e) Final $\boldsymbol{\alpha}$ for LOOP Algorithm

(f) LOOP Algorithm

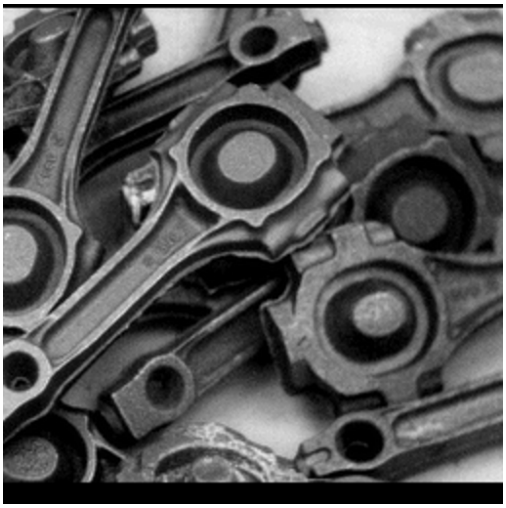
Figure 3.23: Method comparison for reconstructing “peppers” from 2×2 lattice samples.



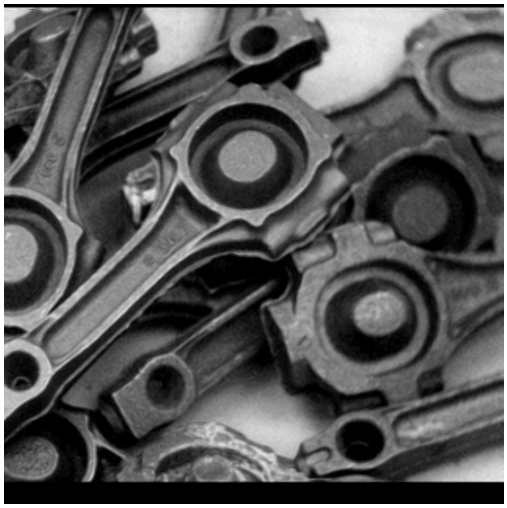
(a) "tools"



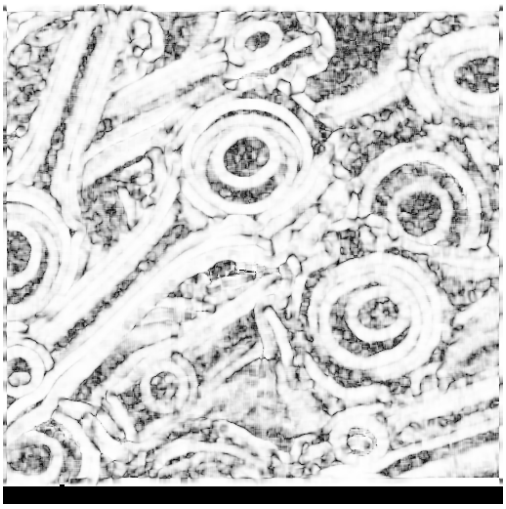
(b) 2×2 lattice downsampling



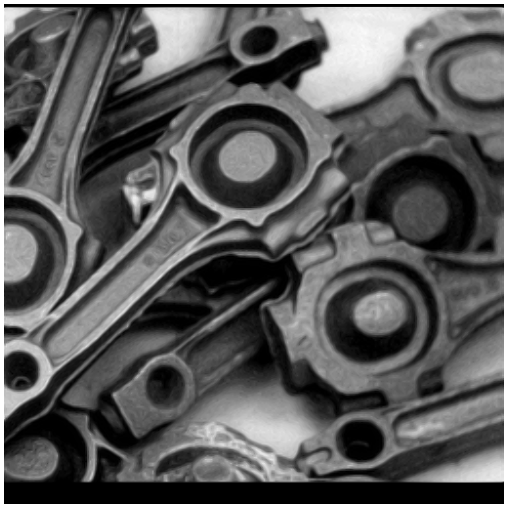
(c) Bicubic interpolation



(d) SAI [11]

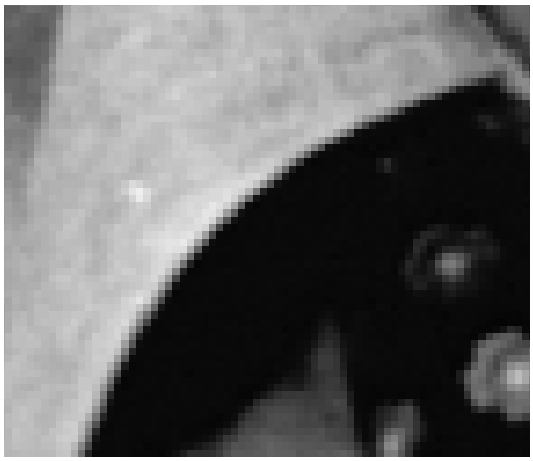


(e) Final $\boldsymbol{\alpha}$ for LOOP Algorithm



(f) LOOP Algorithm

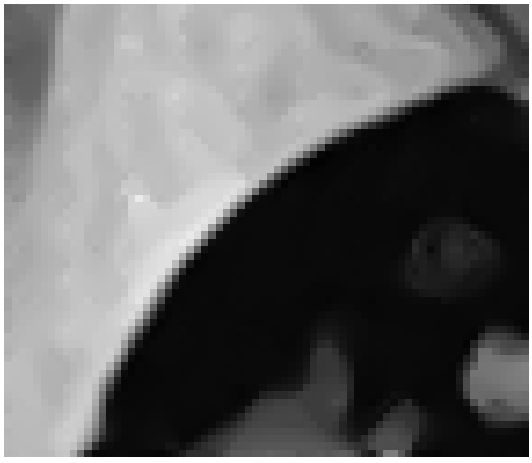
Figure 3.24: Method comparison for reconstructing "tools" from 2×2 lattice samples.



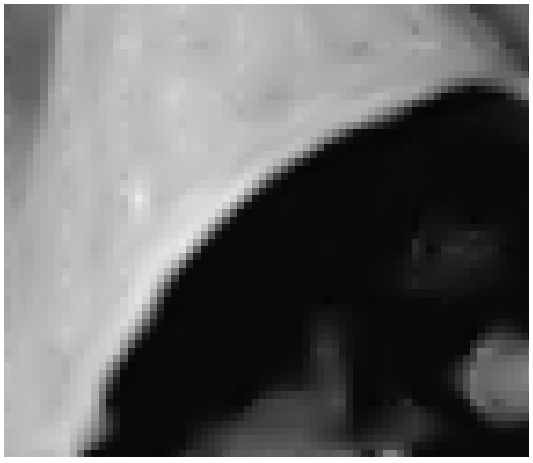
(a) Collar in "Al"

(b) Bicubic recon. from 2×2 lattice

(c) SAI [11] recon. from 2×2 lattice



(d) OG recon. from 7×8 M-Grid.



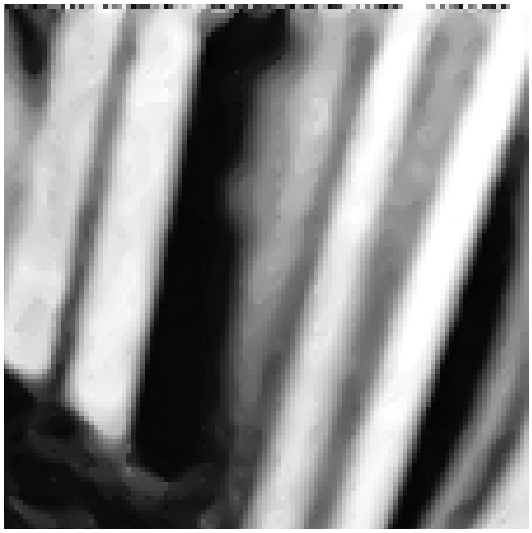
(e) LOOP recon. from 7×8 M-Grid.

Figure 3.25: Edge comparison of collar in "Al"

(a) Books in "Al"

(b) Bicubic recon. from 2×2 lattice

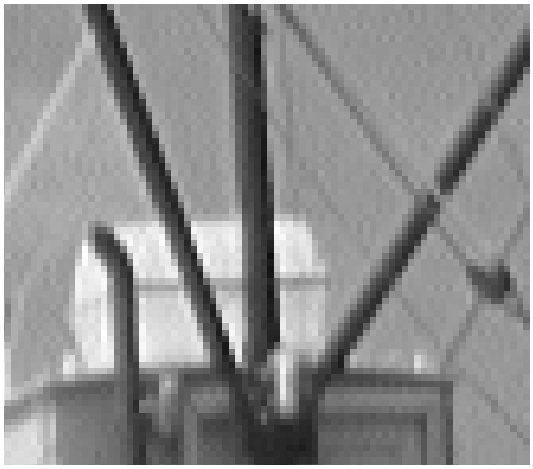
(c) SAI [11] recon. from 2×2 lattice



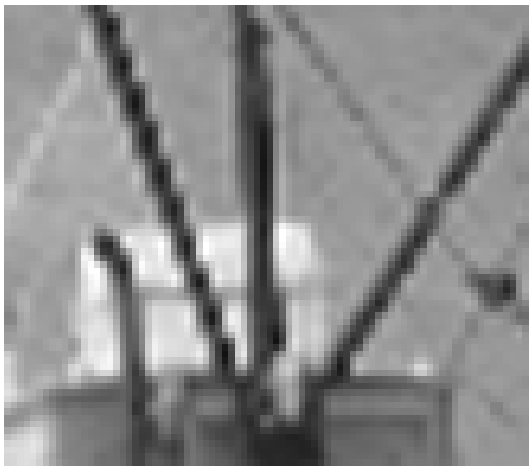
(d) OG recon. from 7×8 M-Grid.

(e) LOOP recon. from 7×8 M-Grid.

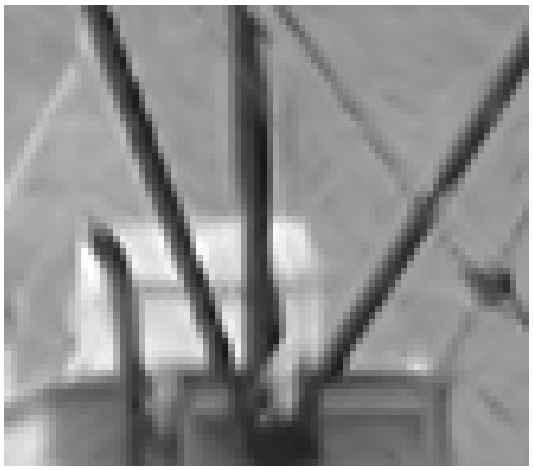
Figure 3.26: Edge comparison of books in "Al"



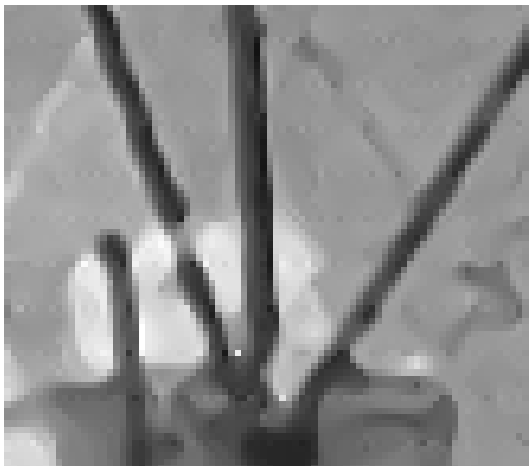
(a) Poles in "Boat"



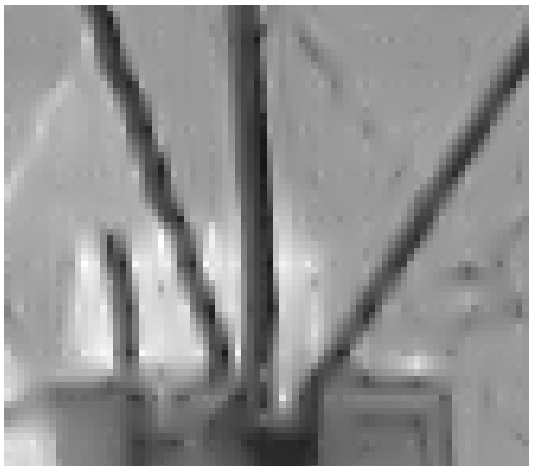
(b) Bicubic recon. from 2×2 lattice



(c) SAI [11] recon. from 2×2 lattice



(d) OG recon. from 7×8 M-Grid.



(e) LOOP recon. from 7×8 M-Grid.

Figure 3.27: Edge comparison for poles in "Boat"

CHAPTER 4

Cutset Sensor Networks, Relay-efficient Functions, and Efficient Communication

A common topic of research in the area of wireless sensor networks is that of finding ways to reduce energy consumption of battery-powered sensors nodes, thereby increasing the overall lifetime of the network. In these wireless networks, there is often a need to transmit sensing data over long distances, for example, to a common sink. Furthermore, these transmissions typically dominate the energy usage of the wireless sensors, since the actual data sensing is usually passive and requires little to no energy. Thus, it is of utmost importance that data can be efficiently relayed through the network, typically by using short "hops" of communication between neighboring sensors. Using many short hops is often desirable because the energy required for communication scales super-linearly with distance. For fixed hardware constraints, the network energy consumption can be reduced by using efficient communication protocols [50–52], distributed algorithms [2,53], and sensor-placement strategies [54,55]. This chapter considers elements of all three of the above strategies, but attention is primarily given to finding deployment strategies that reduce the energy costs required for communication. In particular, these so-called *cutset networks* have sensors placed along the boundaries of square, triangular and hexagonal tessellations of the plane, as shown in Figures 4.1(e,f,g).

For a fixed sensor density, as the number of sensors per side k of the tessellating polygon increases in a cutset network, the spacing between neighboring sensors decreases. Suppose that the energy required for data transmission between two sensors follows a simple power law with exponent greater than one, such as (4.1). Under this simple power-law assumption, as the sensors-per-side k increases for the cutset network, the energy cost for neighboring sensors to communicate decreases very quickly. Further suppose that communication in a sensor network always follows a multi-hop path consisting of neighboring sensors, i.e., two sensors separated at a long distance do not communicate directly, but instead use a sequence of short hops, each from a sensor to a nearby or nearest neighbor. Under this multi-hop assumption, a cutset network will use more hops than a lattice network, but since

each communication hop will be between neighboring sensors, the length of each hop will be smaller for the cutset network. The energy savings obtained using smaller hops will typically outweigh the extra energy required to use more hops in the cutset network; in other words, for multi-hop communication under a simple power law model, "many short hops are more efficient than a few long hops."

Initial discussions and simulations demonstrating the energy-saving possibilities of cutset networks are presented first in Section 4.1. Specifically, under a simple power-law for communication model with exponent greater than one (equation (4.1)), we demonstrate that cutset networks offer reduced communication costs over traditional network geometries such as random deployment (Figure 4.1(a)) or lattices (Figures 4.1(b,c,d)). We predict that efficient communication paths in cutset networks follow multi-hop paths between neighboring sensors, and then show that these predictions agree with cheapest paths obtained using a shortest-path algorithm, such as Dijkstra's Algorithm.

This chapter also explores the energy-accuracy tradeoffs of cutset wireless sensor networks in the context of solving a source localization problem, where the goal is to estimate the location of a source that is emitting isotropic acoustic or electromagnetic waves using received power readings at each sensor. Section 4.2 introduces the received-signal-strength (RSS)-based source localization problem and provides two common noise models for RSS measurements. For the centralized source localization problem where all sensor readings are made available to a common sink, Section 4.2 also provides the Cramèr–Rao bounds for any estimator of a source under these models, as well as a "brute force" method for calculating the Maximum Likelihood Estimator of the source location from noisy samples. Combining the analysis of Sections 4.1 and 4.2, we compare the accuracy-energy tradeoffs provided by cutset networks in Section 4.3 for centralized source localization applications, and conclude that cutset networks offer significant energy savings at the price of reduced accuracy. Later, in Section 4.4.1, we will then focus specifically on Manhattan wireless sensor networks in a decentralized application setting, where nodes must locally determine (a) whether a source is present nearby, and if so, (b) an estimate of the source's location. We design both a decentralized localization algorithm, called the *Midpoint Algorithm*, and a communication protocol, and compare their performance to the decentralized POCS algorithm [2].

Finally, in Sections 4.5 and 4.6, with the goal of understanding the ability of a periodic network topology to provide a communication infrastructure (agnostic to the sensor network task), we present a general method for predicting the required energy-per-distance cost of long distance communication in a lattice sensor network when the model for transmission model is *relay-efficient*. Specifically, given a particular lattice and a particular model for how transmission energy changes with distance (that is what we will define as relay-efficient), we

wish to predict the minimal energy-per-distance cost of communicating as a function of the direction from source to destination. With such, for any given energy model, we can compare and contrast different deployment lattices on the basis of their ability to provide an energy-efficient communication infrastructure. An example of a relay-efficient model is the aforementioned power-law model with exponent greater than one.

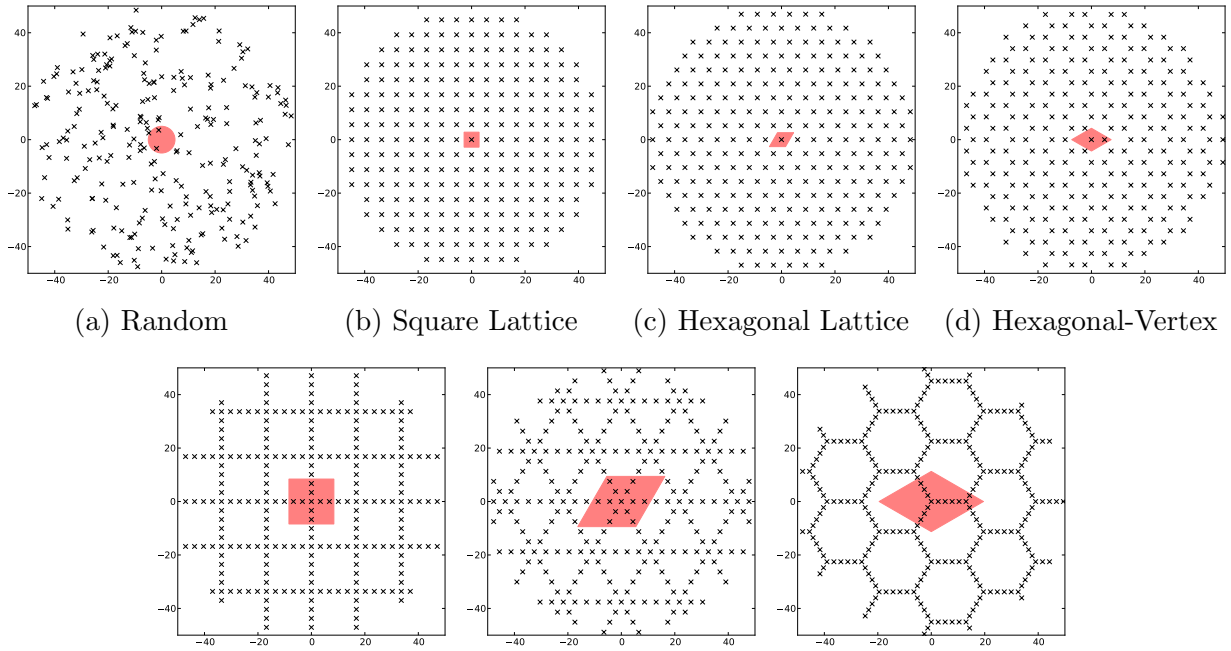
We begin in Section 4.5, by focusing on the energy transmission model and only one pair of sensors. Specifically, we define the *relay region* between two sensors to be the set of locations where a third *relay sensor* can be placed in order to reduce the total communication communication energy exerted by the pair of sensors. In particular, a function $f(x)$ models the energy required for a pair of sensors to communicate at distance x, and we find conditions under which $f(x)$ generates a nonempty relay region for sufficiently large distances x; such functions are said to be *relay-efficient*, meaning that for such energy models, efficient communication use multihop relaying. A main focus of Section 4.5 is thus to find inner and outer bounds to the relay region, and study how these bounds grow as the distance between sensors increases. This key result is given by Fact 52. Finally, we summarize these results with some examples, including a power-law model.

In Section 4.6, under the assumption that our model is relay-efficient, we characterize the cost of paths of minimal energy through a lattice sensor network as a function of the energy model, the lattice, and the direction and length of the communication path. A lattice sensor network is of interest since any periodic sensor network (such as a cutset network) is defined using an underlying lattice, and therefore is a natural place to begin an investigation. We anticipate that in the future, the methods presented in this section can be extended to include general periodic deployments of sensors, including cutset topologies. We begin the section by defining relevant terms. In Section 4.6.1, it is shown that for asymptotically large long distances at some angle, the minimum energy required for communication is equal to the total distance between sensors times the output of some linear program. An avenue for solving this linear program is explored in Section 4.6.2, where we determine that energy-efficient hops in a lattice network must have a length that is upper bounded by a quantity d^*. This result is also used in Section 4.6.3 to find a finite set of hops V^* that are used to form efficient long-distance paths through the lattice network. We conjecture that the solution to the linear program of Section 4.6.1 can be solved in closed form in Section 4.6.4. Finally, we test our results in Section 4.6.5, where for a square lattice, several choices of energy model and a fairly uniform sampling of possible communication directions, we compute the energy of shortest paths through the lattice network using Dijkstra's algorithm, and then compare the agnostic output of Dijkstra's algorithm to the solution to the linear program of Section 4.6.1 and the conjectured closed-form solution in Section 4.6.4. We find that all three

quantities are either the same, or very close. Thus, we conclude that efficient long-distance communication in a lattice network can be approximated with little to no error by solving the linear program of Section 4.6.1, or the closed-form expressions of Section 4.6.4.

The centralized work was presented at ICASSP 2014 [15], and the decentralized work, including the Midpoint algorithm, was presented at ICASSP 2013 [14].

4.1 Cutset Networks



(a) Random (b) Square Lattice (c) Hexagonal Lattice (d) Hexagonal-Vertex

(e) Manhattan, $k = 5$ (f) Triangular, $k = 5$ (g) Honeycomb, $k = 5$

Figure 4.1: Wireless networks with $n \approx 250$ sensors placed in a circular region of radius $R = 50$m. The shaded region depicts possible locations of a randomly placed source in localization experiments. (a,b,c,d) show traditional network layouts; (e,f,g) show proposed *cutset networks*.

In this section, we consider wireless sensor networks consisting of n wireless sensors placed at locations $x_1, \cdots, x_n$ in a circular region B centered at the origin. Each sensor has a wireless transceiver, and we will model the energy required for two sensors to communicate with one another as a power law. We begin by considering a centralized scenario where each sensor must transmit one packet of data from each sensor to a data sink located at the origin. The goal of this section is to compare the minimum energy cost required to complete this task for various sensor deployment topologies at the same density.

It is common to assume that sensors are randomly distributed, as in Fig. 4.1(a). However, in some applications, the network designers are free to choose where sensors are placed, perhaps using one of the lattice layouts in Fig. 4.1(b,c,d). We now show that these are not as efficient at transmitting data as the cutset networks shown in Figure 4.1(e,f,g), which are formed by first placing sensors at the vertices of square, triangular, and hexagonal tessellations, respectively, and then evenly placing $k-1$ sensors between each vertex. Figures 4.1(e,f,g) show cutset networks for $k = 5$.

We are interested in comparing networks with the same sensor density ρ. For fixed ρ, a cutset network with parameter k will have an intersensor spacing λ given by Table 4.1, and the tessellating polygon will have side length $k\lambda$. Thus, as k increases for fixed ρ, the intersensor spacing λ decreases as $\mathcal{O}(k^{-1/2})$, but the tessellating cell area increases as $\mathcal{O}(k)$.

For experimental tractability, we restrict our networks to a circular region $B_R = \{t : \|t\| \leq R\}$ of radius R centered at 0. To generate a network with approximately 250 sensors, we first choose our density to be $\rho = 250/\pi R^2$. The first sensor is placed at the origin, and then an infinite network is generated using the intersensor spacing λ found in Table 4.1. Finally, we truncate our network to B_R. As a result of this process, the final number of sensors n may differ slightly from 250 for each network, since λ is chosen to match a fixed density. Let $\mathcal{X} = \{x_i \in B_R,\, i = 0, \cdots, n-1\}$ denote the set of sensor locations, where our first sensor $x_0 = 0$ is placed at the origin. For random networks, instead of following the above procedure, we placed 250 sensors randomly within B_R.

Our analysis assumes the following far-field energy model: Let α be some communication path-loss exponent between 2 and 4. If two sensors separated by distance d communicate at received power P_0, we model the required transmission energy per packet as

$$w(d) = P_0 d^\alpha.$$

Now suppose all sensors wish to transmit one packet of data to the sensor at x_0 in order to run a centralized algorithm or make a centralized decision; we assume there is a time-division schedule so there is no interference between sensors. Each sensor $x \in \mathcal{X}$ chooses a path, i.e., a sequence of sensor locations $x_{i_0} = x, x_{i_1}, x_{i_2}, \ldots, x_{i_m}$ in $\mathcal{X}$, to relay its data to x_0, engendering a path cost equal to $P_0 \sum_{j=1}^{m} \|x_{i_j} - x_{i_{j-1}}\|^\alpha$, and a total energy that is the sum of such over all $x \in \mathcal{X}$. It is possible to compute the minimum possible total communication energy consumed by the network. First, form the complete weighted graph $G = (\mathcal{X}, \mathcal{X} \times \mathcal{X}, W)$, where W is the weighting matrix containing costs of direct communication between sensor nodes i and j, i.e. $[W]_{ij} = w(\|x_i - x_j\|)$. Dijkstra's algorithm [56] is then used to compute a set of minimum cost paths from the central node to all other nodes requiring $\mathcal{O}(n^2)$

operations; the total minimum cost $\mathcal{E}_{true}$ is then the total sum of weights along each of these paths. Note that because our assumed α is greater than one, all hops in any optimal path in a cutset network will connect nearest neighbors.

This cost can be estimated using fewer computations. Specifically, we seek to derive a function $c(r, \phi)$ that estimates the cost of transmitting a packet from a sensor at radius r and angle ϕ to the central node $x_0 = 0$.[1] The total minimum energy is approximately

$$\mathcal{E}_{true} \approx \mathcal{E}_{sum} \triangleq \sum_{i=1}^{n} c(\|x_i\|, \angle x_i)\,. \tag{4.1}$$

This sum requires $\mathcal{O}(n)$ operations. Expressions for $c(r, \phi)$ for three different cutset networks are given in Table 4.1. These expressions are derived under the assertion that all minimum cost paths follow the cell boundaries of the tessellation that generated the cutset network, and hops are always between neighboring sensors (see Appendix C). If n is large (specifically, the density $n/\pi R^2$ is large), then we may be able to model its local sensor density with some function $\rho(r, \phi)$. The total minimum energy is approximately

$$\mathcal{E}_{true} \approx \mathcal{E}_{int} \triangleq \int_0^R \int_0^{2\pi} \rho(r, \phi)\, c(r, \phi)\, r\, dr d\phi\,. \tag{4.2}$$

For large networks based on tessellations, we approximate $\rho(r, \phi)$ as a constant ρ. For ρ constant, it is straightforward to calculate closed-form expressions for $\mathcal{E}_{int}$ using integration; the resulting quantities are given in Table 4.1. When the number of sensors per tessellation boundary k is large, and k, R, ρ and α are fixed, honeycomb networks require the least energy. Specifically, Manhattan networks and triangular networks require $3^{(\alpha-1)/4}$ and $\frac{1}{2}3^{\alpha/2}$ times more energy than honeycomb networks, respectively.

Fig. 4.2 compares the output of Dijkstra's algorithm to the sum (4.1) and integral (4.2) approximations for $\rho = 250/\pi R^2$ and $R = 50$m. These approximations performed reasonably well. As expected, the honeycomb networks outperformed the other networks for fixed k. A misleading part of Fig. 4.2 is that the honeycomb network consumed less energy than predicted and required more energy for $k = 5$ than 4. This occurred because the $k = 4$ network contained only 235 sensors, whereas the $k = 5$ network contained 259. Actually, because λ decreases, the energy-per-sensor decreased from the $k = 4$ to 5. Fig. 4.3 shows the results for the same experiment except on a dense network with $n \approx 1000$ sensors and $k = 1, \cdots, 20$. Note that the downward energy trends continue with increased k.

Referring to Table 4.1, for fixed k and ρ, the sensor spacing λ will be smallest for honey-

[1] It's worth noting that the $c(r, \phi)$ functions mentioned here for cutset networks are separable in r and ϕ, i.e. $c(r, \phi) = r\widetilde{c}(\phi)$ for some $\widetilde{c}(\phi)$.

Network	ρ: Density	λ : Sensor Spacing	$c(r,\phi)$: Min path cost	$\mathcal{E}_{int}$: Energy estimate
Manhattan	$\frac{2k-1}{k^2\lambda^2}$	$\frac{1}{\sqrt{\rho}}\frac{\sqrt{2k-1}}{k}$	$\lambda^{\alpha-1}\, r\, (\lvert\cos\phi\rvert + \lvert\sin\phi\rvert)$	$\frac{8}{3}\left(\frac{\sqrt{2k-1}}{k}\right)^{\alpha-1} R^3\rho^{\frac{3-\alpha}{2}}$
Triangular	$\frac{2}{\sqrt{3}}\frac{3k-2}{k^2\lambda^2}$	$\frac{1}{\sqrt{\rho}}\sqrt{\frac{2}{\sqrt{3}}}\frac{\sqrt{3k-2}}{k}$	$\lambda^{\alpha-1}\, r\, \left(\cos\phi + \frac{\lvert\sin\phi\rvert}{\sqrt{3}}\right),\ \lvert\phi\rvert \le \frac{\pi}{6}$	$\frac{4\sqrt{3}}{3}\left(\sqrt{\frac{2}{\sqrt{3}}}\frac{\sqrt{3k-2}}{k}\right)^{\alpha-1} R^3\rho^{\frac{3-\alpha}{2}}$
Honeycomb	$\frac{2}{3\sqrt{3}}\frac{3k-1}{k^2\lambda^2}$	$\frac{1}{\sqrt{\rho}}\sqrt{\frac{2}{3\sqrt{3}}}\frac{\sqrt{3k-1}}{k}$	$\frac{4}{3}\lambda^{\alpha-1}\, r\, \cos\phi,\ \lvert\phi\rvert \le \frac{\pi}{6}$	$\frac{8}{3}\left(\sqrt{\frac{2}{3\sqrt{3}}}\frac{\sqrt{3k-1}}{k}\right)^{\alpha-1} R^3\rho^{\frac{3-\alpha}{2}}$

Table 4.1: Network quantities. Note that $c(r,\phi)$ is $\frac{\pi}{3}$ periodic for Triangle and Honeycomb networks.

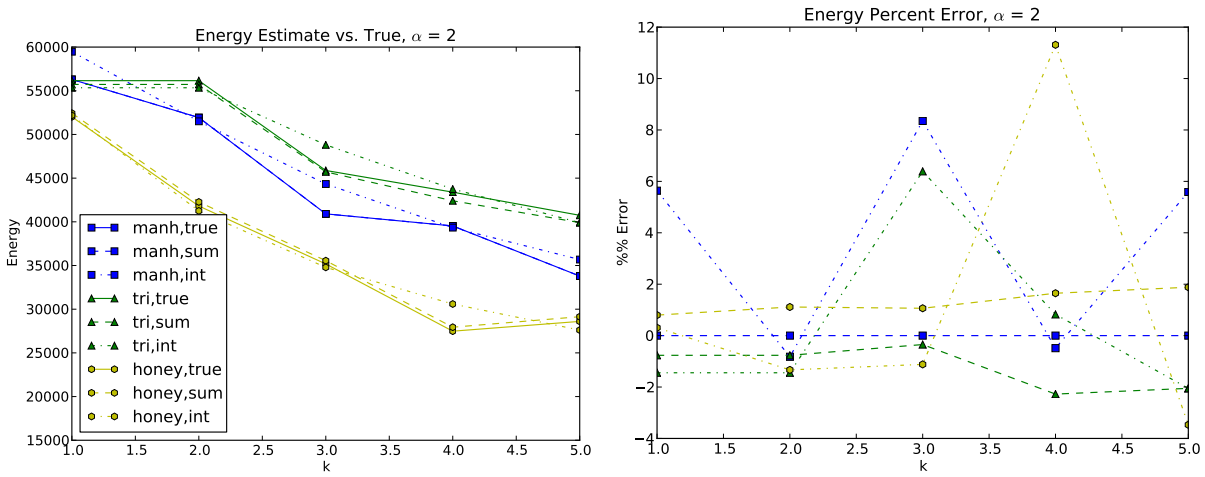


Figure 4.2: Energy estimates for $n \approx 250$ and $k = \{1, 2, 3, 4, 5\}$. All sum approximation estimate (4.1) had no more than 3% error, and all integral approximation estimates (4.2) had no more than 12% error.

comb networks, which is consistent with the honeycomb networks outperforming the other networks in our simulations. We suspect that this is related to the classic (and recently proven [57]) "honeycomb conjecture," which states that "any partition of the plane into regions of equal area has perimeter at least that of the regular hexagonal honeycomb tiling" (quote from [57]). Under the constraint that the "covering radius" of an infinite sensor network is bounded (i.e. every point in the plane is within some distance r_{max} of a sensor), for the power law model, we hypothesize that there may be a way to connect the smallest perimeter problem in [57] with the problem of finding cheapest paths in a wireless sensor network. We also hypothesize that the good performance of the Honeycomb networks is derived from the fact that when transmitting a message between two sensors in a Honeycomb network, the shortest path typically deviates less from an ideal straight-line path than for the other networks considered here. These hypotheses warrant further investigation in future work.

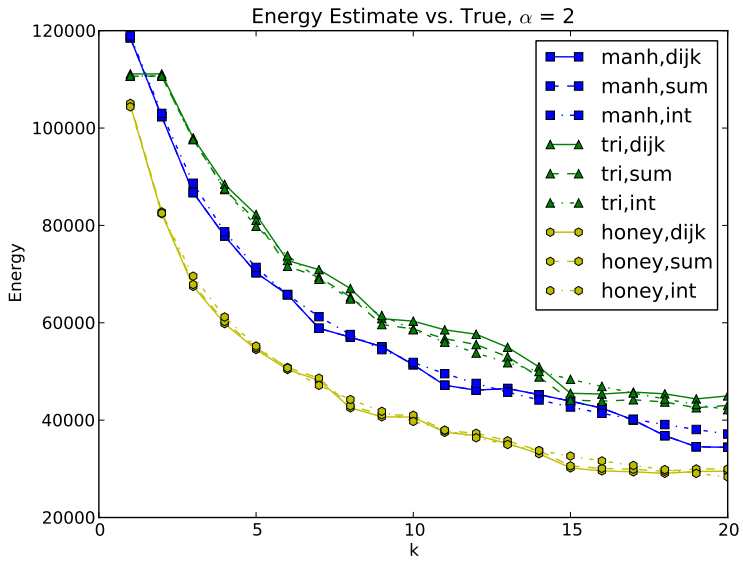


Figure 4.3: Energy estimates for $n \approx 1000$ and $k = \{1, \cdots, 20\}$.

4.2 The Source Localization Problem

In this section, we review the problem of source localization for sensor networks. Suppose a source at location $\theta \in B_R$ is emitting electromagnetic or acoustic waves. Each sensor in our sensor network makes a noisy measurement of the received signal strength of the source. The goal of a source localization problem is to estimate the source location θ using the noisy sensor measurements. Specifically, in this section we will consider the centralized source localization problem, where all sensor measurements are made available at a central data sink, so that the estimate $\widehat{\theta}$ is based on all of the data, and not a subset. This is in contrast to a decentralized source localization problem, where individual sensors must (a) decide whether a source is present, and (b) if a source is present, they must collaborate with neighboring sensors to make an estimate. Our performance metric is the root-mean squared error between our estimate $\widehat{\theta}$ and the true location θ. We present two noise models for signal strength measurements, derive Cramér-Rao lower bounds to the MSE of the best unbiased estimation rules for each of these models, and outline a "brute force" Maximum Likelihood Estimator (MLE) for each model, which estimates the source location by plugging in candidate values of θ on a search grid into the likelihood function; θ is then estimated to be the grid value that produces the largest likelihood.

After the problem is introduced in this section, in Section 4.3, we will compare both the energy performance and accuracy performance of various sensor deployments, and see if there is a tradeoff between the energy required to make a source location estimate $\widehat{\theta}$, and the quality of the estimate in terms of root mean-squared error.

We begin by letting $\overline{\mathbf{y}}(\theta) \in \mathbb{R}_+^n$ denote a vector of received signal strength (RSS) measurements at the sensor nodes under no noise (one component for each sensor). If the sensors are sufficiently far from the source (no closer than some $\epsilon > 0$), we can assume the following far-field sensing model, where the ith element of $\overline{\mathbf{y}}(\theta)$ is modeled according to

$$\overline{y_i}(\theta) = \frac{A}{\|x_i - \theta\|^\beta},$$

where A is the known[2] reference power of the signal at one meter and β is some sensing path-loss exponent, typically between 2 and 4 [58].

We consider two measurement (or signal strength) noise models: the Additive White Gaussian Noise (AWGN) model and the log-normal (LN) model. Excellent descriptions of both models are given in [58, 59]. Under the AWGN model, the observed RSS at node i is

$$y_i = \overline{y_i}(\theta) + u_i, \tag{4.3}$$

where each u_i is zero-mean i.i.d. Gaussian noise with variance σ^2. Thus, our observations $\mathbf{y} = [y_i]_{i=1}^n$ are distributed as $\mathcal{N}(\overline{\mathbf{y}}(\theta), \sigma^2 I)$. In our experiments, we thresholded any negative y_i value to zero in order to avoid negative RSS readings.

The log-normal model is slightly more realistic than the AWGN model, as it as been observed in practice [60–63] and derived analytically [64]. Under the LN model our observations are Gaussian in the log domain, i.e. the RSS in dB at the ith sensor node is

$$y_{i,db} = 10 \log_{10} \overline{y}_i(\theta) + v_i,$$

where each v_i is zero-mean i.i.d. Gaussian noise with known standard deviation σ_{db}. Our vector of observations $\mathbf{y}_{db} = [y_{i,db}]_{i=1}^n$ is normally distributed as $\mathcal{N}(10 \log_{10} \overline{\mathbf{y}}(\theta), \sigma_{db}^2)$, where the $\log_{10}$ is an abuse of notation denoting an element-wise logarithm. The standard deviation σ_{db} is typically observed to be between 4 and 12 [61].

4.2.1 Cramér–Rao Bounds

We briefly derive the Cramér–Rao bounds (CRB's) for these models, which are lower bounds on the variance of any unbiased estimator of θ based on a set of observations. The reader may refer to [58, 59] for more thorough derivations.

[2]Knowing A is a reasonable assumption in applications where we have access to additional information about the source that we are sensing. For example, in some applications we will have access to a vehicle/cellphone/animal that emits EM/acoustic waves, thereby allowing us to measure the signal power at a distance of 1 meter.

Recall that our goal is to estimate $\theta = [\theta_1, \theta_2]^T$ from RSS observations. If $\mathbf{z}$ is a random vector distributed as $\mathcal{N}(\boldsymbol{\mu}(\theta), \sigma^2 I)$, then the jk-th element of the Fisher information matrix F is

$$[F]_{jk} = \frac{1}{\sigma^2} \frac{\partial \boldsymbol{\mu}^T}{\partial \theta_j} \frac{\partial \boldsymbol{\mu}}{\partial \theta_k}.$$

The variance of any unbiased estimator of the jth unknown coordinate $\widehat{\theta}_j$ given some observations $\mathbf{z}$ is lower bounded according to $\mathrm{Var}(\theta_j) \geq [F^{-1}]_{jj}$; this is known as the *Cramèr–Rao Bound*. Under the AWGN model, $\mathbb{E}[\mathbf{z}] = \overline{\mathbf{y}}$ and

$$\frac{\partial \overline{y}_i}{\partial \theta_j} = \beta A \, \frac{(x_{ij} - \theta_j)}{\|x_i - \theta\|^{\beta+2}}, \; j = 1, 2.$$

Similarly, under the LN model, $\mathbb{E}[\mathbf{z}] = 10 \log_{10} \overline{\mathbf{y}}(\theta)$, and

$$\frac{\partial (10 \log_{10} \overline{y}_i)}{\partial \theta_j} = \frac{10\beta}{\ln 10} \, \frac{(x_{ij} - \theta_j)}{\|x_i - \theta\|^2}, \; j = 1, 2.$$

These derivatives are easily computable, and the CRB can be calculated via an inversion of the 2×2 Fisher information matrix F.

4.2.2 Maximum Likelihood Estimation

Given some or all sensor observations y_i, under both noise models, the maximum likelihood (ML) solution can be found by minimizing the negative-log likelihood function. Under the AWGN model, the ML solution for estimating θ from $\mathbf{y}$ is given by solving the nonlinear least-squares problem

$$\widehat{\theta}_{ML} = \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{n} \left[y_i - \overline{y}_i(\theta) \right]^2.$$

Similarly, under the LN model, the ML solution of estimating θ from $\mathbf{y}_{db}$ is given by

$$\widehat{\theta}_{ML} = \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{n} \left[y_{i,db} - 10 \log_{10} \overline{y}_i(\theta) \right]^2.$$

In centralized applications, which we consider in Section 4.3, all sensor observations are transmitted to a central sensor node or data fusion center. In this case, any methods provided in [59] can be used to solve the ML problem. We implemented their multiresolution search method, where $\widehat{\theta}_{ML}$ is calculated by substituting a large number of candidate values of θ on a grid, first at a coarse search resolution, and then at a fine resolution centered at the coarse estimate.

4.3 Centralized Source Localization on Cutset Networks

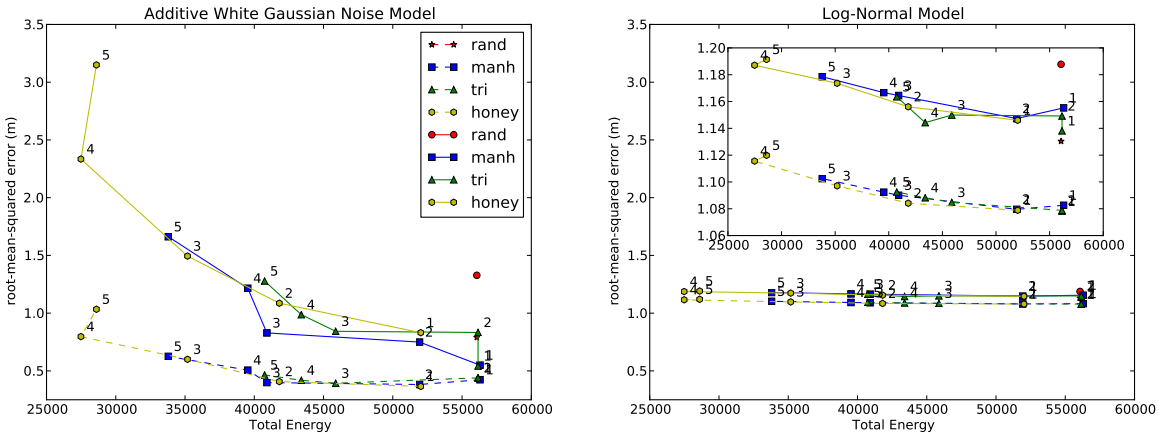


Figure 4.4: Experimental results for MLE and CRB experiments for both noise models. Solid lines indicate MLE, and dashed lines indicate CRB. The log-normal plot includes a zoomed-in plot for closer comparison. The numbers along each data point indicate the k value of the network.

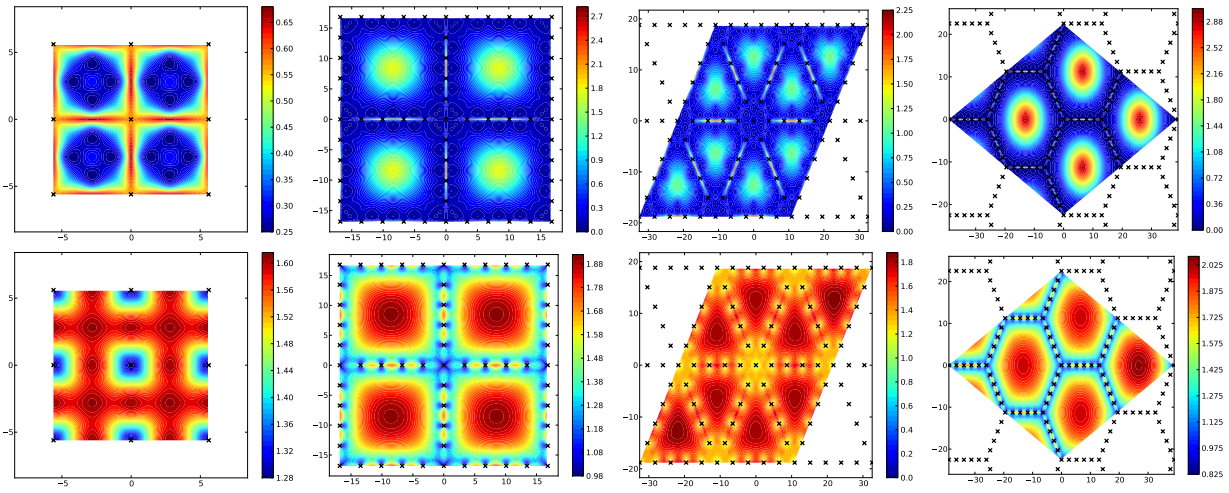
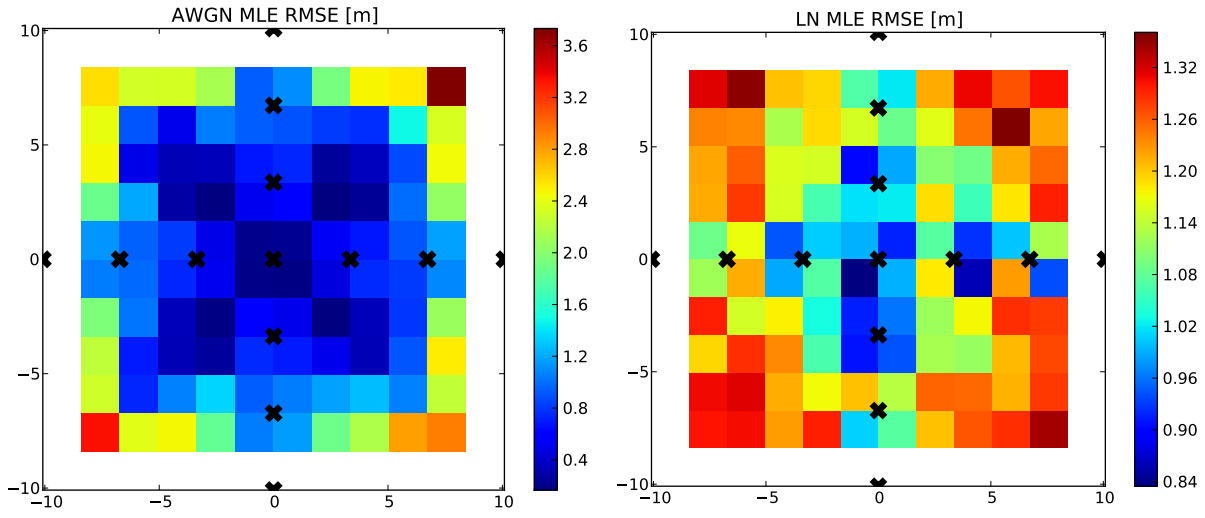


Figure 4.5: Cramèr-Rao bound as a function of source position θ; each $\times$ marks a sensor position. Top row: AWGN model. Bottom row: LN model. Left-to-right: Square lattice, $k = 1$, Manhattan, $k = 5$, Triangular, $k = 5$, Honeycomb, $k = 5$.

In this section, we merge the results of Sections 4.1 with the review of the source localization problem in Section 4.2. Specifically, we run a series of simulations of a centralized source localization problem, where we generate noisy RSS measurements at each sensor, and then compute (a) the energy required to transmit all RSS measurements to a central hub, (b) the Maximum Likelihood Estimate of θ based on all sensor readings, and (c) the Cramér-Rao



(a) AWGN Model (b) LN Model

Figure 4.6: MLE error distribution for Manhattan grid with $n \approx 250$ sensors and $k = 5$, search window of circle of radius 30, coarse search resolution 0.15m, fine search resolution 0.01m, 20000 trials.

bound on the variance of any unbiased estimator of θ. Our goal is to compare the energy required to make an estimate of θ with the quality of the estimate (in MSE) for various sensor network topologies. In particular, we would like to demonstrate that cutset networks offer a tradeoff between energy and accuracy when solving a centralized source localization problem.

4.3.1 Procedure

To test the performance of cutset networks, we followed the following procedure. For a circular region B_R with radius $R = 50$, we generated various networks of network density $\rho = 250/\pi R^2$ according to the procedure in Section 4.1. To test the performance of a random network (Fig. 4.1(a)), the sensor positions were randomized for each trial. To test the performance of deterministic networks, we generated Manhattan, triangular and honeycomb networks with $k = \{1, 2, 3, 4, 5\}$. Note that $k = 1$ corresponds to the lattices shown in Fig. 4.1(b,c,d), and a triangular network with $k = 1$ is exactly equivalent to a triangular network with $k = 2$ at the same density. Cutset networks with $k = 5$ are shown in Fig. 4.1(e,f,g). Energy use was measured by calculating $\mathcal{E}_{true}$ using Dijkstra's Algorithm, as described in Section 4.1. For each network type, $10,000$ trials were performed for both the AWGN and LN noise models using reference powers $A = 100$, sensing path-loss $\alpha = 2$, communication path-loss $\beta = 2$, communication received-power $P_0 = 1$, AWGN noise variance $\sigma^2 = 1$, and

log-normal noise standard deviation $\sigma_{db} = 4$. In each trial, the source location θ was chosen randomly within the red shaded regions in Fig. 4.1. The CRB was calculated for the current θ value, a new realization of noisy data was generated, and the multiresolution "brute force" MLE algorithm (Section 4.2.2 was performed to obtain an estimate $\widehat{\theta}_{ML}$ using a coarse grid search of 1m over the entire network, followed by a fine grid search of 0.01m centered at the coarse estimate. Upon completion of all trials, the average CRB along each coordinate θ_1 and θ_2 was calculated, and then the root-sum of these two average CRB's was computed, obtaining a lower bound on Root Mean Squared Error (RMSE) under a uniform prior for θ. Additionally, the RMSE of the maximum likelihood estimates were calculated. The results for both noise models are plotted in Figure 4.4. Figure 4.6 shows the results of an additional experiment designed to show the distribution of MLE errors for a $k = 5$ Manhattan grid.

4.3.2 Discussion

Figure 4.4 shows how cutset networks offer significant increases in energy efficiency over random networks and lattice networks without surrendering much accuracy. This is shown in both the results of the MLE experiments and the average CRB calculations. The honeycomb networks with $k = 4$ had the greatest gains in energy efficiency, offering a factor of 2 improvement over random networks and lattices. These energy gains are even more significant for larger values of the communication path-loss exponent α. We note that an explanation was given in Section 4.1 for why energy increased from $k = 4$ to 5 for honeycomb networks. Finally, as expected, Figure 4.6 show how MLE performed much better near the intersection of Manhattan grid lines, and much worse near the center of squares where the distance to the nearest sensor was maximized.

Additionally, Figure 4.5 shows how the CRB varies over position in various cutset networks; it was generated by calculating the CRB over a deterministic meshgrid. One expected result is that the CRB has a local maximum towards the center of the cutset tessellation shapes, where the smallest distance to any network sensor is maximized. One surprising result is that the CRB tends to also increase drastically along the cutset lines. We did not observe a significant increase in errors along these lines while running our MLE experiments; thus, we would like to determine if these increases lie along a set of zero measure (i.e. along the cutset lines) or if they increase sharply but smoothly along the direction orthogonal to the cutset lines. Additionally, we believe that this behavior may be attributed to the fact that it is difficult to detect the "sign" of the source location when it is close to a cutset line, i.e. a source on either side of the cutset line at the same distance will produce the same RSS under no noise.

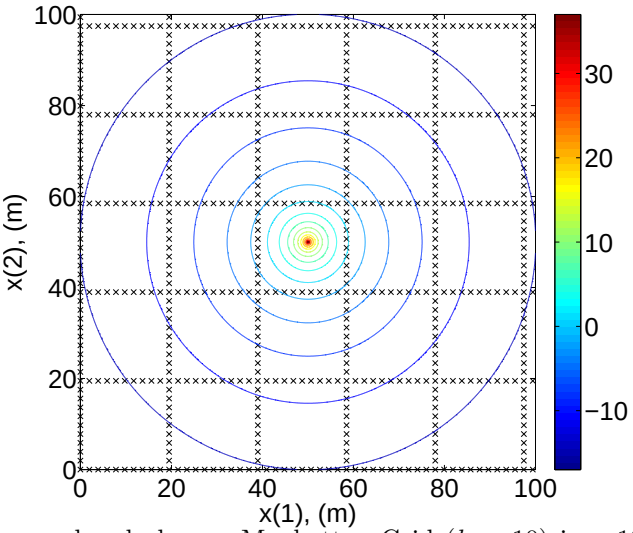


Figure 4.7: $n = 500$ sensors placed along a Manhattan Grid ($k = 10$) in a 100m $\times$ 100m square; each $\times$ denotes one sensor. A source is located at $\theta = [50, 50]$. Contours of constant power under no noise are shown in dB.

4.4 Decentralized Source Localization on a Manhattan Network

Thus far, we have focused our attention on using cutset to solve the problem of RSS-based source localization. In this section, we consider the task of estimating the position of a source emitting electromagnetic or acoustic waves in a decentralized manner; specifically, this means that sensors in a sensor network must locally determine (a) if a sensor is present, and if so, then (b) they must communicate locally with one-another to make an estimate of the source position θ. We are also interested in solving this decentralized task in an energy-efficient manner. As a first investigation, we focus only on Manhattan wireless sensor networks.

We begin by reiterating geometric properties of a Manhattan grid. An infinite Manhattan grid is described by two parameters: its intersensor spacing λ and the Manhattan grid parameter k. This grid partitions space into $k\lambda \times k\lambda$ blocks. Associating $2k-1$ sensors with each block, we see that the sensor density is $\frac{2k-1}{(k\lambda)^2}$. Now suppose we would like to construct a finite Manhattan grid with n sensors and parameter k over a $w \times w$ square region. Since for some values of k it is impossible to choose λ so that a Manhattan grid with n sensors and parameters (k, λ) exactly covers the $w \times w$ square, we choose sensor spacing $\lambda = w\sqrt{(2k-1)/(nk^2)}$, and generate a finite Manhattan grid with $B = \left(\lfloor \frac{w}{k\lambda} \rfloor\right)^2$ grid blocks and $(2k-1)B + 2k\sqrt{B} - 1 \approx n$ sensors.

We now describe the energy cost advantages of communicating data within a Manhattan grid sensor network, and later we propose the *Midpoint Algorithm* for further reductions in energy. Mimicking the analysis in [53,65], the total energy needed for any source localization

algorithm is

$$\mathcal{E} = b \times h \times e, \tag{4.4}$$

where b is the number of total sensor transmissions made by all sensors in the network, h is the number of hops through the network per transmission, and e is the energy required to transmit a single hop. As we will now discuss, e is greatly affected by the choice of sensor placement, in particular the intersensor spacings, and since for typical algorithms h and b are not nearly as affected, a first-order approach for comparing the energy performance of various sensor layouts is to estimate e. To do so, note that the average power emitted by a sensor at distance d will be modeled as decaying as $\mathcal{O}(d^{-\alpha})$, where α is between 2 and 4. Thus, the energy required for one communication hop between neighboring sensors is $e = \mathcal{O}(d^{\alpha})$. If we place n sensors randomly over a $w \times w$ square, the average distance between neighboring sensors is $w/\sqrt{n}$ and $e = \mathcal{O}(n^{-\alpha/2})$. However, when the sensors are placed with spacing λ along a Manhattan grid of parameter k and density n/w^2, the distance between neighboring sensors is $\lambda = w\sqrt{(2k-1)/nk^2}$, and $e = \mathcal{O}(k^{-\alpha/2}n^{-\alpha/2})$, thus reducing energy by a factor $\mathcal{O}(k^{\alpha/2})$ over a random placement.

We formulate problem of estimating a source localization θ from noisy observations in a decentralized manner in Section 4.4.1. In Section 4.4.2, the decentralized *Midpoint Algorithm* is proposed to solve the source localization problem on a Manhattan grid. The accuracy vs. energy performance of the Midpoint Algorithm is discussed in Section 4.4.5 and compared to the recent decentralized POCS algorithm [2].

4.4.1 Problem Statement

Suppose n sensors are distributed over a $w \times w$ square along a Manhattan grid. Along each row/column of the Manhattan grid, $m = w/\lambda$ sensors are spaced $\lambda = w\sqrt{(2k-1)/nk^2}$ apart. For some positive integer k, each row/column of the grid is spaced $k\lambda$ apart. These four quantities n, m, k, λ are dependent, so we fix the number of sensors n and vary the Manhattan grid parameter k.

Let $x_i = [x_i(1),\ x_i(2)]$ denote the location of the ith sensor. A source with known intensity A is positioned at unknown location $\theta = [\theta(1),\ \theta(2)]$ within the $w \times w$ square. The source emits a signal whose strength decays with distance to the power β, where β is typically in the range of 2 to 4. For this application, we consider only the AWGN noise model (4.3). For convenience, we reprint it here; recall that for each i, the ith sensor makes a noisy measurement y_i of the received signal strength (RSS), modeled as

$$y_i = \frac{A}{\|x_i - \theta\|^{\beta}} + u_i. \tag{4.5}$$

where $\|\cdot\|$ is the Euclidean norm, A and β are fixed and known, and the u_i's are i.i.d. zero-mean Gaussian noise with known variance σ^2. [2] contains more information about the theory behind this model, and an application using real-world data can be found in [66]. Extensions to noise models involving fading [58] are possible.

Depending upon the sensor deployment geometry and localization algorithm, groups of neighboring sensors must communicate with each other and decide if the source is within their vicinity. If so, they must also estimate its location. Hence, our proposed algorithm must operate in a decentralized manner. A decentralized algorithm's performance is determined by (a) the probability that sensors locally close to the source will actually detect the source, (b) the false alarm probability, i.e., the probability that a sensor located far from the source will mistakenly declare a detection, and (c) the average squared error between the true location θ and its estimate $\widehat{\theta}$ in cases of a correct detection. We also consider the communication cost (4.4).

In [53], Rabbat and Nowak proposed a decentralized source localization algorithm based on incremental subgradient optimization, but they did not specify how to detect the presence of a source before making an estimate. In [2], Blatt and Hero proposed a decentralized source localization method based on projections onto convex sets (POCS). This algorithm required choosing a threshold γ such that all sensors with received RSS greater than γ were considered *active*; thus, the detection probability was determined by this threshold. The active sensors collaborated to produce an estimate of θ based on their RSS measurements. One improvement of POCS over Rabbat and Nowak's method was a smaller energy cost (4.4); in particular, POCS reduced the number of sensor transmissions b required for their algorithm to converge. Both iterative algorithms can be applied to a Manhattan grid sensor network to solve the problem of source localization. However, we will propose a *non-iterative* algorithm, called the *Midpoint Algorithm*, that exploits the Manhattan grid geometry to reduce communication costs at the price of higher estimation error.

Finally, in [67], Rabbat and Nowak consider various estimators to solve the problem of source localization, including

$$\widehat{\theta} = \frac{\sum_{i=1}^{n} x_i 1_{\{y_i > \gamma\}}}{\sum_{i=1}^{n} 1_{\{y_i > \gamma\}}}, \tag{4.6}$$

where γ is a threshold and $1_{\{y_i>\gamma\}}$ is the indicator function. This estimator is simply an average of active sensor locations, and we modify it for a Manhattan grid in the next section.

4.4.2 Midpoint Algorithm

We can take advantage of the Manhattan grid structure to reduce the communication cost of forming an estimate of θ. The straight rows of sensors in a Manhattan grid suggest that

when a source is close to a row, e.g. a horizontal row, one can estimate the horizontal position of the source along the row by exploiting the fact that the intensity distribution is expected to be symmetric. Thus, we can expect the horizontal position to lie at the average position of the endpoints of the symmetric distribution, i.e., the *midpoint* of the endpoints of the distribution. Calculating this midpoint will require very little communication. Likewise, one can repeat in the vertical direction using Mahattan columns to estimate the vertical position of the source. One hopes that a very simple, very low energy localization algorithm will result.

Thus, we propose the *Midpoint Algorithm*, which differs from the (4.6) in two principal ways. First, instead of jointly estimating $\theta(1)$ and $\theta(2)$ from all active sensors, the Midpoint Algorithm estimates $\theta(1)$ from active sensors in Manhattan grid rows, and $\theta(2)$ from active sensors in grid columns. Second, it replaces the average location in (4.6) with the midpoint between the active sensors in each grid row (grid column) that are farthest apart.

For concreteness, denote the set of sensors in the jth row by

$$\mathcal{H}_j = \{i : x_i(1) = \ell\lambda,\ x_i(2) = jk\lambda,\ \ell = 0, \cdots, m-1\}.$$

We say that sensor i in $\mathcal{H}_j$ is *active* if $y_i > \gamma$ for some predetermined threshold γ, and we say that the jth row $\mathcal{H}_j$ is *active* if it contains an active sensor. Whereas one could estimate $\theta(1)$ from an active row as in (4.6):

$$\widehat{\theta}_j(1) = \frac{\sum_{i\in\mathcal{H}_j} x_i(1) \cdot 1_{\{y_i>\gamma\}}}{\sum_{i\in\mathcal{H}_j} 1_{\{y_i>\gamma\}}},$$

the Midpoint Algorithm estimates $\theta(1)$ as

$$\widehat{\theta}_j(1) = \frac{x_a(1) + x_b(1)}{2}, \tag{4.7}$$

which is simply the midpoint of the first coordinates of the left- and rightmost active sensors x_a and x_b in grid row j. These will be called *endpoints* of the active row. Similarly, from sensors in grid columns, an estimate $\widehat{\theta}(2)$ of $\theta(2)$ is made for every active column. If there is at least one active row and one active column, then an estimate $\widehat{\theta} = (\widehat{\theta}(1), \widehat{\theta}(2))$ can be made for each pairing of an active row and active column. For each such pair, the corner point shared by the row and column is called a *decision corner*.

In Section 4.4.3 it will be shown how to choose γ to ensure that with high probability at least one of the four corners of the $k\lambda \times k\lambda$ block containing the source is a decision corner and there are no decision corners outside the block. Section 3.2 describes a distributed

communication protocol that distributes endpoint locations so as to (1) enable those corners lying on the aforementioned block to determine whether or not they are decision corners, and (2) to enable such decision corners to make their estimates of θ.

Note that in the absence of noise, sensors in a grid row (column) will lie in a single consecutive interval. Although this does not necessarily happen in the presence of noise, results in Section 3.2.6 show that the Midpoint Algorithm works well nevertheless.

4.4.3 Choosing the threshold

Our choice of γ heavily impacts the performance of the Midpoint Algorithm. If γ is too large, then the probability of having at least one decision corner will not be large, i.e., the probability of missed detection will be too large. If γ is too small, there will be spurious decision corners, leading to high probability of false alarms and poor estimates of θ. Thus the goal of this section is to find an upper bound γ_1 and lower bound γ_2 such that the these undesirable events occur with low probability for any threshold satisfying $\gamma_2 \leq \gamma \leq \gamma_1$.

We begin with the upper bound γ_1. Let E_1 be the event that at least one of the corners of the $k\lambda \times k\lambda$ block containing the source is a decision corner, thereby indicating a successful detection. We want to choose γ_1 small enough that the $\Pr(E_1) \geq 1 - \varepsilon_1$, where ε_1 is some small tolerance. A useful fact is that the closest row sensor and the closest column sensor to the source are within distance $\frac{\lambda}{2}\sqrt{k^2+1}$. If these sensors are active, then E_1 occurs. Therefore, using this fact and the union bound, for any γ,

$$\begin{aligned}
\Pr(E_1) &\geq \Pr\left(\text{closest row sensor and closest column sensor are both active}\right) \\
&= 1 - \Pr\left(\text{closest row sensor inactive or closest column sensor inactive}\right) \\
&\geq 1 - \Pr\left(\text{closest row sensor inactive}\right) - \Pr\left(\text{closest column sensor inactive}\right) \\
&= 1 - 2\Pr\left(\text{closest row sensor inactive}\right) \\
&= 1 - 2\Pr\left(\frac{A}{(\frac{\lambda}{2}\sqrt{k^2+1})^\beta} + u \leq \gamma\right).
\end{aligned}$$

Since u is Gaussian with known variance σ, equating the RHS of the above to $1 - \varepsilon_1$ yields the fact that if

$$\gamma \leq \gamma_1 \triangleq \frac{2^\beta A}{(\lambda\sqrt{k^2+1})^\beta} - \sigma\mathcal{Q}^{-1}\left(\frac{\varepsilon_1}{2}\right), \tag{4.8}$$

then $\Pr(E_1) \geq 1 - \varepsilon_1$. In the above, $\mathcal{Q}(x) = \Pr(X > x)$ for a zero mean, unit variance

Gaussian random variable X, and $\mathcal{Q}^{-1}$ is its inverse function.

In a similar manner, we calculate a threshold lower bound γ_2. Consider the event E_2 that there are no decision corners outside the $k\lambda \times k\lambda$ block containing the source, so there are at most four decisions corners. We want to choose γ so that $\Pr(E_2) \geq 1 - \varepsilon_2$. This also ensures that the false alarm rate will be at most ε_2. Using the fact that E_2 occurs when all sensors farther than $k\lambda$ from the source are inactive, similar to our derivation for E_1, it can be shown using the union bound that

$$\Pr(E_2) \geq 1 - n \Pr\left(\frac{A}{(k\lambda)^\beta} + u > \gamma\right).$$

Again, equating the RHS of the above event to $1 - \varepsilon_2$ yields the fact that if

$$\gamma \geq \gamma_2 \triangleq \frac{A}{(\lambda k)^\beta} + \sigma \mathcal{Q}^{-1}\left(\frac{\varepsilon_2}{n}\right), \tag{4.9}$$

then $\Pr(E_2) \geq 1 - \varepsilon_2$.

For large k values, the Manhattan grid "block size" $k\lambda$ becomes very large, and it becomes physically impossible to detect certain source locations without incurring a large false alarm rate. Thus, in our experiments, we only choose values of k small enough that $\gamma_1 > \gamma_2$. We found that the Midpoint Algorithm performs better for large γ, so we set our threshold to be the upper bound $\gamma = \gamma_1$.

4.4.4 Communication protocol and costs

By our choice of γ in the previous section, with high probability there will be at least one decision corner on the $k\lambda \times k\lambda$ block containing the source, and there will be no decision corners outside this block. We now describe a distributed communication protocol by which sensors efficiently report endpoint data to corner points, enabling those that are decision corners to recognize that they are such and to make their estimates.

Assume sensor clocks are synchronized. Time is slotted and the system operates with cycles of $8m$ slots, where m is the number of sensors in a row or column. The following protocol operates during the first $4m$ slots along rows, and repeats during the next $4m$ slots along columns.

During the first m time slots, messages are sent left-to-right across each row of the grid. Specifically, during slot t, only sensor t of each row may transmit, and neighboring sensor $t+1$ listens[3]. If sensor t of row j did not hear a message (from $t-1$) during the previous time

[3]Sensors in adjacent Manhattan grid rows are presumed to be far enough away (at least $k\lambda$) that transmissions from adjacent grid rows do not interfere.

slot, it knows the first active sensor in its grid row has not been found (the first endpoint). It then compares its measured RSS to the threshold. If $y_t < \gamma$, sensor t is not active and does not transmit. However, if $y_t > \gamma$, sensor t is active and transmits 0 to its neighbor $t+1$, thereby marking t as the first endpoint.

If, on the other hand, sensor t did hear a message from $t-1$, it increments the message by 1 and transmits the new message to sensor $t+1$. Thus, each message is an integer representing the distance to the first active sensor in the row. Message-passing ends after the message is received by two corner sensors (we assume sensors know whether they are placed on a corner *a priori*). This requires an extra "corner counting" bit to be sent along with each transmission.

During the next m time slots, the sensor order is reversed and messages are passed right-to-left in a similar manner in order to determine the second endpoint in the row. In some cases, after these $2m$ time slots, at least one corner knows the locations of both endpoints and can estimate $\theta(1)$. However, if both endpoints are less than $k-2$ sensors apart, it is possible that the endpoints lie entirely between two adjacent corners, and these corners will only know one endpoint apiece. In this case, the two endpoints (and the sensors inbetween) will know both endpoint locations. Thus, the next $2m$ time slots are reserved for the endpoints to transmit the missing endpoint locations to their closest corner sensor.

As mentioned earlier, this protocol is repeated for columns in the next $4m$ time slots. After all $8m$ time slots, any corner that has received both horizontal and vertical pairs of endpoints, recognizes itself as a decision corner and makes an estimate $\widehat{\theta}$. It can be seen that this protocol finds at least one decision corner on the block containing the source, if there is one, which happens with high probability.

We now find an upper bound to the communication costs of the protocol. Due to our choice of threshold, it is easy to see that each decision corner will be within $2k$ sensors of an endpoint with probability $1-\varepsilon_2$. It can be shown that this protocol requires at most $4k$ transmissions per active row. Each distance transmission requires $\lceil \log_2(2k) \rceil$ bits to transmit endpoint data with an overhead of 1 bit for corner counting, totaling $2 + \lceil \log_2 k \rceil$ bits per transmission.

4.4.5 Experiments and Results

For our experiments, we chose $w = 1000$, $n = 10,000$, and for various values of k we designed finite Manhattan grids as described in the introduction. We set $A = 10,000$, $\sigma = 1$, $\beta = 2$, and the threshold γ was set to the upper bound $\gamma = \gamma_1$ with $\varepsilon_1 = \varepsilon_2 = 10^{-5}$. To avoid edge effects, θ was distributed randomly in a $k\lambda \times k\lambda$ block near the center of the 1000 m $\times$1000

m square. We tested $k = 2, \ldots, 14$, all of which satisfied the condition $\gamma_1 > \gamma_2$. For each value of k, the squared error was calculated for estimates produced by both the Midpoint Algorithm and the POCS algorithm [2]. 20,000 trials of this experiment were performed, during which we did not observe any missed detections or false alarms for either algorithm. In some trials, multiple estimates of θ were generated by the Midpoint Algorithm using different decision corners. The choice of these multiple estimates did not impact the overall performance of our algorithm, so we chose an estimate randomly.

The POCS algorithm was chosen for comparison because of its high accuracy and low communication costs, needing many fewer cycles to converge than other algorithms such as [53]. POCS also required a choice of threshold γ_{POCS}; we observed that POCS performed better when a slightly smaller threshold was used than the Midpoint Algorithm. To still ensure a detection probability of at least $1 - \varepsilon_1$ and a false alarm rate less than ε_2, we chose $\gamma_{POCS} = \gamma_2$. The POCS algorithm also required a convergence threshold; we used the value of 10^{-3} as used in [2]. In addition to running POCS on a Manhattan grid, we ran POCS for sensors placed on a uniform lattice (labeled $k = 1$) as well as for randomly placed sensors. For these experiments, we found that thresholds of $\gamma_{lattice} = 360$ and $\gamma_{random} = 15$ worked well (for comparison, $\gamma_2 = 106$ with $k = 1, \varepsilon_1 = \varepsilon_2 = 10^{-5}$). Note that we need a much smaller threshold for the random network because, unlike the uniform lattice, we are not guaranteed to have a sensor close to the source.

In addition, bounds on the energy cost of each algorithm were calculated. Suppose there are r active sensors above threshold. The POCS algorithm needs some number of cycles c to converge and one extra cycle to calculate an average estimate of θ. Note that c typically depends on some convergence criteria; we used the default criteria suggested in [2], which was that the previous estimate of θ during the last cycle is within 10^{-3} of the new estimate for the first sensor in the cycle. For our experiments, c ranged between 4 and 7. Although we used double precision for $\widehat{\theta}$ in our simulation, we assumed that each coordinate of θ was quantized to 3 significant decimal places when being transmitted, which corresponds to $\lceil \log_2(10^3) \rceil = 10$ bits per coordinate. Thus, $b_{POCS} = 20 \cdot (c + 1) \cdot r$. Finally, we conservatively assumed that $h = 1$ for POCS since *most* transmissions are between neighbors. This is conservative because some transmissions require inactive sensors to relay data between active sensors, in which case $h > 1$. Thus, for each trial we calculated

$$\mathcal{E}_{POCS} = 20r(c + 1)\lambda^{\alpha}.$$

Note that this is a conservative lower bound on the true energy.

Now we consider the energy cost of the Midpoint Algorithm. Define N_{row} and N_{col} to

be the number of active rows and columns, respectively. Following the discussion in Section 4.4.4, at most $4k$ transmission are needed per active row/column, and we transmit $2+\lceil \log_2 k \rceil$ bits per transmission. Each transmission is always between neighboring sensors, so unlike the POCS algorithm, we always have that $h = 1$. Thus, the total energy required is at most

$$\mathcal{E}_{midpoint} = 4k(2 + \lceil \log_2 k \rceil)(N_{row} + N_{col})\lambda^\alpha.$$

Observe that the $4k$ transmissions per active row/column is an upper bound on the number of transmissions. We emphasize that $\mathcal{E}_{midpoint}$ is a conservative upper bound for the Midpoint Algorithm, whereas $\mathcal{E}_{POCS}$ is a conservative lower bound for POCS. These energy cost bounds were calculated for both $\alpha = 2$ and $\alpha = 4$. We plotted both the root mean squared error (RMSE) and root median squared error (RMedSE) vs. energy cost in Figure 4.8. Plotting the root median squared error is useful because of its insensitivity to outliers.

First, we consider the performance of POCS for either value of α. When POCS was run on a uniform lattice and Manhattan grid, less energy was used than on a randomly distributed lattice. However, error also gradually increased as k increased. This shows the fundamental tradeoff between a random network, a uniform lattice network, and a Manhattan grid. That is, if we are willing to tolerate an increase in error, the Manhattan grid requires much less energy.

Now let us compare the performance of the Midpoint Algorithm to POCS. If we are willing to sacrifice more accuracy, the Midpoint Algorithm uses even less energy than POCS for all values of k and fixed α. For a fixed accuracy level, it is possible to make POCS more competitive by choosing a convergence threshold larger than 10^{-3}, thereby reducing the required number of transmissions while decreasing the accuracy. However, even when we increased this threshold, we found that the Midpoint Algorithm outperformed POCS for a fixed achievable accuracy. It is interesting to point out that for k increasing and n fixed, the energy cost of the Midpoint Algorithm increases for $\alpha = 2$ and decreases for $\alpha = 4$. Note that $\mathcal{E}_{midpoint} = \mathcal{O}(k^{1-\alpha/2} \log k)$. Thus, the energy cost increases as $\mathcal{O}(\log k)$ for $\alpha = 2$, but decreases as $\mathcal{O}(\log(k)/k)$ when $\alpha = 4$.

4.5 Relay Regions and Relay-efficient Functions

The end-goal of this section and Section 4.6 is to describe energy efficient communication paths through lattice networks, and determine their energy-per-distance cost. However, before we investigate communication in a network of *many* sensors, it is important to consider the task of communication between a *pair* of sensors. For some energy models, the best long

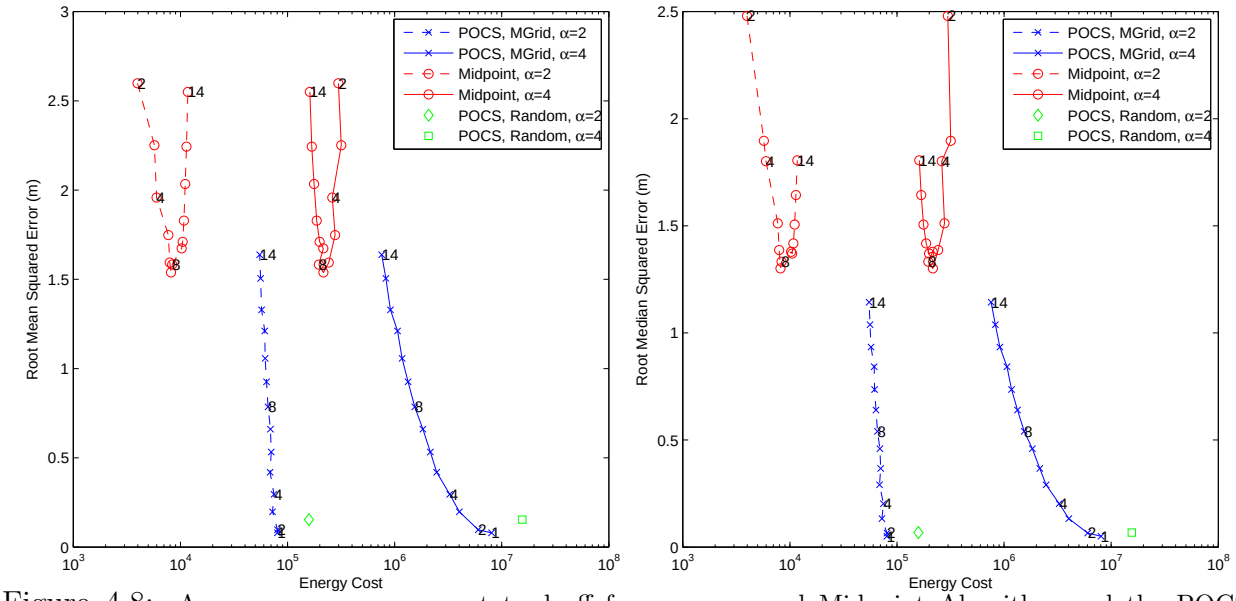


Figure 4.8: Accuracy vs. energy cost tradeoff for our proposed Midpoint Algorithm and the POCS algorithm [2] for $\alpha = 2$ and $\alpha = 4$. Values of k are labeled for some points. POCS was also run on a uniform lattice (labeled $k = 1$) and a randomly distributed network. RMSE vs. energy is shown in (a) and RMedSE vs. energy cost is shown in (b).

distance communication may simply be to use direct transmission and avoid any relaying whatsoever, in which case the most efficient path is simply a single, direct hop. A rather simple example of this is when the cost of communication between two sensors at distance x is simply a constant, i.e., $f(x) = C$, for $C > 0$. However, in some cases, like a power law $f(x) = x^\beta$ for $\beta \geq 2$, or a power law plus a constant $f(x) = x^\beta + C$ with $\beta \geq 2$, it may be cheaper to avoid direct communication, and instead relay messages through the network.

Therefore, in this section, we take a mathematical look at functions that are used to model the energy costs of wireless transmission. We are particularly interested in knowing when it is *strictly better* to relay through an intermediate sensor when transmitting a message. Specifically, we are concerned with functions $f : \mathbb{R}^+ \to \mathbb{R}^+$ that model the energy required to transmit a packet of data between two sensor at distance x. We restrict our attention to f that are continuous, convex, and nondecreasing. These functions may be zero-valued at the origin ($f(0) = 0$) or they may have a positive-valued "overhead cost" $f(0) > 0$. The set of all points where a relay sensor can be placed in order to reduce the total communication energy between two sensors property will be called a *relay region*, and functions f that generate a nonempty relay region for sufficiently large x will be called *relay-efficient functions.* An example of a relay-efficient function is a power-law with exponent greater than or equal to 2; the anxious reader is encouraged to look ahead to Section 4.5.6 for more examples of relay-efficient and non-relay-efficient functions.

Later, when we consider the problem of finding efficient paths in lattice networks in

Section 4.6, it will be important to assume that $f(x)$ is relay efficient. Otherwise, the best (trivial) path will always be direct transmission.

4.5.1 Relay regions and their properties

Definition 13 (Relay region)**.** *Suppose $f : \mathbb{R}^+ \to \mathbb{R}^+$ is a continuous, convex, nondecreasing function. For any two points $\mathbf{x}, \mathbf{y} \in \mathbb{R}^2$, define the* relay region $\mathcal{R}(\mathbf{x}, \mathbf{y})$ *to be*

$$\mathcal{R}(\mathbf{x}, \mathbf{y}) = \left\{ \mathbf{z} \in \mathbb{R}^2 : f(\|\mathbf{x} - \mathbf{z}\|) + f(\|\mathbf{z} - \mathbf{y}\|) < f(\|\mathbf{x} - \mathbf{y}\|) \right\}, \tag{4.10}$$

where $\|\cdot\|$ denotes the usual Euclidean 2-norm.

Note that when transmitting a message from a sensor at location $\mathbf{x}$ to a sensor at location $\mathbf{y}$, if $R(\mathbf{x}, \mathbf{y})$ is not empty, the total energy cost will be strictly cheaper to relay the message through a sensor located at any point in $\mathcal{R}(\mathbf{x}, \mathbf{y})$.

Definition 14 (Standard relay region)**.** *Suppose $f : \mathbb{R}^+ \to \mathbb{R}^+$ is a continuous, convex, nondecreasing function. Define the* standard relay region *to be*

$$\widetilde{\mathcal{R}}(x) = \left\{ \mathbf{z} \in \mathbb{R}^2 : f\left(\left\| \left(-\frac{x}{2}, 0 \right) - \mathbf{z} \right\| \right) + f\left(\left\| \mathbf{z} - \left(\frac{x}{2}, 0 \right) \right\| \right) < f(x) \right\}. \tag{4.11}$$

An example of $\widetilde{\mathcal{R}}(x)$ for $f(x) = Px^2$ for some $P > 0$ is shown in Figure 4.9(a).

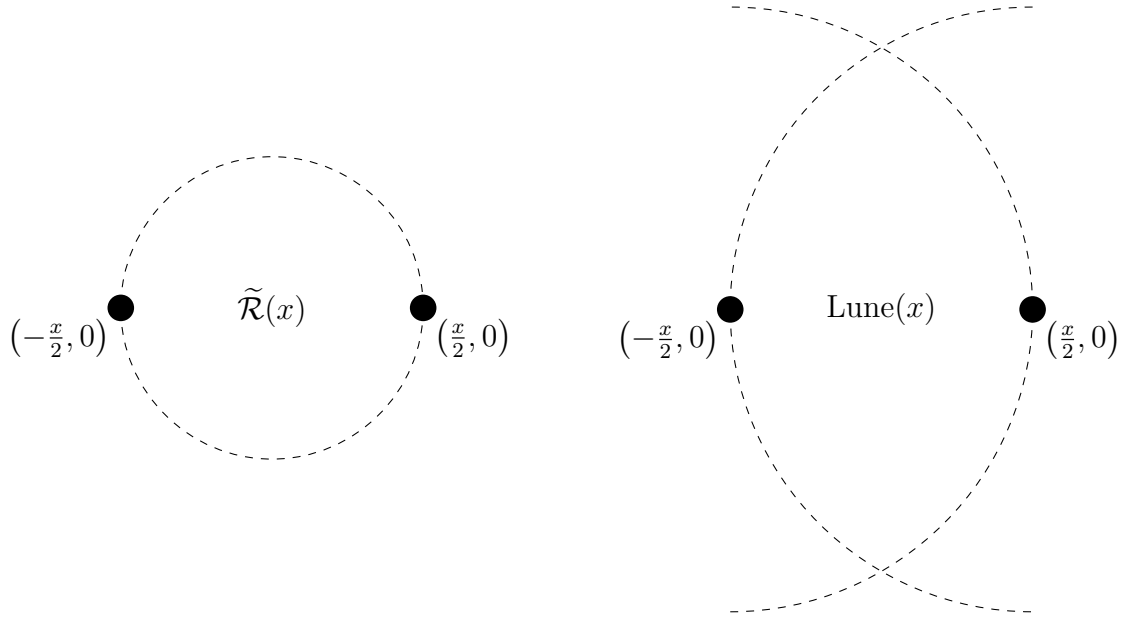
Fact 15. *There is a one-to-one correspondence between $\mathcal{R}(\mathbf{x}, \mathbf{y})$ and $\widetilde{\mathcal{R}}(x)$ defined according to the rigid motion*

$$\mathcal{R}(\mathbf{x}, \mathbf{y}) = T_{\angle(\mathbf{y}-\mathbf{x})} \widetilde{\mathcal{R}}(\|\mathbf{x} - \mathbf{y}\|) + \frac{\mathbf{x} + \mathbf{y}}{2},$$

where T_θ denotes a counter-clockwise rotation by θ about the origin.

Proof. Let $\widetilde{\mathbf{z}} \in \widetilde{\mathcal{R}}(\|\mathbf{x} - \mathbf{y})\|)$ and define $\theta = \angle(\mathbf{y} - \mathbf{x})$. Define $\mathbf{e}_1 = (1, 0)$ and let

$$\mathbf{z} = T_\theta \widetilde{\mathbf{z}} + \frac{\mathbf{x} + \mathbf{y}}{2}.$$



(a) $\widetilde{\mathcal{R}}(x)$ for$f(x) = Px^2$ (b) Lune(x)

Figure 4.9: Example of a relay region, and the boundary Lune(x).

Then it is clear that $\mathbf{z} \in \mathcal{R}(\mathbf{x}, \mathbf{y})$ since it satisfies

$$\begin{aligned}
f(\|\mathbf{x}-\mathbf{z}\|) + f(\|\mathbf{z}-\mathbf{y}\|) &= f\left(\left\|\mathbf{x} - T_\theta\widetilde{\mathbf{z}} - \frac{\mathbf{x}+\mathbf{y}}{2}\right\|\right) + f\left(\left\|T_\theta\widetilde{\mathbf{z}} + \frac{\mathbf{x}+\mathbf{y}}{2} - \mathbf{y}\right\|\right) \\
&= f\left(\left\|\frac{\mathbf{x}-\mathbf{y}}{2} - T_\theta\widetilde{\mathbf{z}}\right\|\right) + f\left(\left\|T_\theta\widetilde{\mathbf{z}} + \frac{\mathbf{x}-\mathbf{y}}{2}\right\|\right) \\
&= f\left(\left\|-\frac{\mathbf{y}-\mathbf{x}}{2} - T_\theta\widetilde{\mathbf{z}}\right\|\right) + f\left(\left\|T_\theta\widetilde{\mathbf{z}} - \frac{\mathbf{y}-\mathbf{x}}{2}\right\|\right) \\
&= f\left(\left\|-\frac{\|\mathbf{y}-\mathbf{x}\|T_\theta\mathbf{e}_1}{2} - T_\theta\widetilde{\mathbf{z}}\right\|\right) + f\left(\left\|T_\theta\widetilde{\mathbf{z}} - \frac{\|\mathbf{y}-\mathbf{x}\|T_\theta\mathbf{e}_1}{2}\right\|\right) \\
&= f\left(\left\|-\frac{\|\mathbf{y}-\mathbf{x}\|\mathbf{e}_1}{2} - \widetilde{\mathbf{z}}\right\|\right) + f\left(\left\|\widetilde{\mathbf{z}} - \frac{\|\mathbf{y}-\mathbf{x}\|\mathbf{e}_1}{2}\right\|\right) \\
&= f\left(\left\|-\left(\frac{\|\mathbf{y}-\mathbf{x}\|}{2},0\right) - \widetilde{\mathbf{z}}\right\|\right) + f\left(\left\|\widetilde{\mathbf{z}} - \left(\frac{\|\mathbf{y}-\mathbf{x}\|}{2},0\right)\right\|\right) \\
&< f(\|\mathbf{y}-\mathbf{x}\|).
\end{aligned}$$

A similar argument holds for the converse. □

Fact 16 (Convexity of $\mathcal{R}(\mathbf{x}, \mathbf{y})$ and $\widetilde{\mathcal{R}}(x)$)**.** *If $f : \mathbb{R}^+ \to \mathbb{R}^+$ is a continuous, convex, nondecreasing function, then $\mathcal{R}(\mathbf{x}, \mathbf{y})$ is convex for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^2$. Furthermore, $\widetilde{\mathcal{R}}(x)$ must also be convex.*

Proof. Suppose $\mathbf{r}, \mathbf{s} \in \mathcal{R}(\mathbf{x}, \mathbf{y})$, $0 < \theta < 1$, and $\mathbf{t} = \theta\mathbf{r} + (1-\theta)\mathbf{s}$. It suffices to show $\mathbf{t} \in \mathcal{R}(\mathbf{x}, \mathbf{y})$, i.e., $f(\|\mathbf{x}-\mathbf{t}\|) + f(\|\mathbf{t}-\mathbf{y}\|) < f(\|\mathbf{x}-\mathbf{y}\|)$. We have

$$\begin{aligned}
f(\|\mathbf{x}-\mathbf{t}\|) + f(\|\mathbf{t}-\mathbf{y}\|) &= f(\|\mathbf{x}-\theta\mathbf{r}-(1-\theta)\mathbf{s}\|) + f(\|\theta\mathbf{r}+(1-\theta)\mathbf{s}-\mathbf{y}\|) \\
&= f(\|\theta(\mathbf{x}-\mathbf{r}) + (1-\theta)(\mathbf{x}-\mathbf{s})\|) + f(\|\theta(\mathbf{r}-\mathbf{y}) + (1-\theta)(\mathbf{s}-\mathbf{y})\|) \\
&\leq f(\|\theta(\mathbf{x}-\mathbf{r})\| + \|(1-\theta)(\mathbf{x}-\mathbf{s})\|) + f(\|\theta(\mathbf{r}-\mathbf{y})\| + \|(1-\theta)(\mathbf{s}-\mathbf{y})\|) \\
&\quad \ldots \text{by triangle inequality with monotonic nondecreasing } f(x) \\
&= f(\theta\|\mathbf{x}-\mathbf{r}\| + (1-\theta)\|\mathbf{x}-\mathbf{s}\|) + f(\theta\|\mathbf{r}-\mathbf{y}\| + (1-\theta)\|\mathbf{s}-\mathbf{y}\|) \\
&\leq \theta f(\|\mathbf{x}-\mathbf{r}\|) + (1-\theta) f(\|\mathbf{x}-\mathbf{s}\|) + \theta f(\|\mathbf{r}-\mathbf{y}\|) + (1-\theta) f(\|\mathbf{s}-\mathbf{y}\|) \\
&\quad \ldots \text{using convexity of } f(x) \\
&= \theta\left[f(\|\mathbf{x}-\mathbf{r}\|) + f(\|\mathbf{r}-\mathbf{y}\|)\right] + (1-\theta)\left[f(\|\mathbf{x}-\mathbf{s}\|) + f(\|\mathbf{s}-\mathbf{y}\|)\right] \\
&< \theta f(\|\mathbf{x}-\mathbf{y}\|) + (1-\theta) f(\|\mathbf{x}-\mathbf{y}\|) \\
&\quad \ldots \text{by definition of } \mathcal{R}(\mathbf{x}, \mathbf{y}) \\
&= f(\|\mathbf{x}-\mathbf{y}\|).
\end{aligned}$$

Finally, the convexity of $\widetilde{\mathcal{R}}(x)$ follows the fact that relay regions and standard relay regions are related via a rigid motion, as noted in Fact 15. □

Fact 17 (Rectangular bound on standard relay region)**.** *Suppose $f : \mathbb{R}^+ \to \mathbb{R}^+$ is a continuous, convex, nondecreasing function. Then the standard relay region $\widetilde{\mathcal{R}}(x)$ is contained in the rectangular region*

$$\widetilde{\mathcal{R}}(x) \subset \left[-\frac{1}{2}x, \frac{1}{2}x\right] \times \left[-\frac{\sqrt{3}}{2}x, \frac{\sqrt{3}}{2}x\right].$$

Proof. If $\widetilde{\mathcal{R}}(x) = \emptyset$, then the result holds trivially. Thus, suppose $\widetilde{\mathcal{R}}(x)$ is not empty. We first demonstrate that if a point lies outside the horizontal interval $[-\frac{x}{2}, \frac{x}{2}]$, then the point cannot lie in the relay region. Without loss of generality, take $\mathbf{z} = (z_1, z_2)$ where $z_1 > \frac{x}{2}$, i.e., $z_1 \notin [-\frac{x}{2}, \frac{x}{2}]$. Then we have

$$\begin{aligned}
f\left(\left\|\left(-\frac{x}{2}, 0\right) - \mathbf{z}\right\|\right) + f\left(\left\|\mathbf{z} - \left(\frac{x}{2}, 0\right)\right\|\right) & \\
&\geq f\left(\left\|\left(-\frac{x}{2}, 0\right) - (z_1, 0)\right\|\right) + f\left(\left\|(z_1, 0) - \left(\frac{x}{2}, 0\right)\right\|\right) \\
&> f\left(\left\|\left(-\frac{x}{2}, 0\right) - \left(\frac{x}{2}, 0\right)\right\|\right) + f\left(\left\|\left(\frac{x}{2}, 0\right) - \left(\frac{x}{2}, 0\right)\right\|\right) \\
&= f(x) + f(0) \\
&\geq f(x),
\end{aligned}$$

and thus $\mathbf{z}$ cannot be contained in $\widetilde{\mathcal{R}}(x)$ by definition. A similar argument holds for $z_1 < -\frac{x}{2}$.

We now demonstrate that if a point lies outside the vertical interval $[-\frac{\sqrt{3}}{2}, \frac{\sqrt{3}}{2}]$, then the point cannot lie in the relay region. Without loss of generality, assume that $\mathbf{z} = (z_1, z_2)$, where $z_1 \geq 0$ and $z_2 > \frac{\sqrt{3}}{2}$. Then we have

$$\begin{aligned} f\left(\left\|\left(-\frac{x}{2}, 0\right) - \mathbf{z}\right\|\right) + f\left(\left\|\mathbf{z} - \left(\frac{x}{2}, 0\right)\right\|\right) & \\ > f\left(\left\|\left(-\frac{1}{2}x, 0\right) - \left(0, \frac{\sqrt{3}}{2}x\right)\right\|\right) + f\left(\left\|\mathbf{z} - \left(\frac{x}{2}, 0\right)\right\|\right) & \\ = f(x) + f\left(\left\|\mathbf{z} - \left(\frac{x}{2}, 0\right)\right\|\right) & \\ \geq f(x), & \end{aligned}$$

and thus $\mathbf{z}$ cannot be contained in $\widetilde{\mathcal{R}}(x)$ by definition. Similar arguments hold for the symmetric cases where $z_1 \leq 0$ and/or $z_2 < -\frac{\sqrt{3}}{2}$. □

Fact 18 (Lune-shaped bound on standard relay region)**.** *Define the lens-shaped region*

$$\mathit{Lune}(x) = \left\{\mathbf{z} \in \mathbb{R}^2 : \max\left\{\left\|\left(-\frac{x}{2}, 0\right) - \mathbf{z}\right\|, \left\|\mathbf{z} - \left(\frac{x}{2}, 0\right)\right\|\right\} \leq x\right\}$$

to be the set of all points that are as least as close to both $(-\frac{x}{2}, 0)$ *and* $(\frac{x}{2}, 0)$ *as* $(-\frac{x}{2}, 0)$ *is to* $(\frac{x}{2}, 0)$*, as illustrated in Figure 4.9(b). Then*

$$\widetilde{\mathcal{R}}(x) \subset \mathit{Lune}(x).$$

Proof. If $\widetilde{\mathcal{R}}(x) = \emptyset$, then the result holds trivially. Thus, suppose $\widetilde{\mathcal{R}}(x)$ is not empty. Let $\mathbf{z} \in \widetilde{\mathcal{R}}(x)$ and define $d_{max} = \max\left\{\left\|\left(-\frac{x}{2}, 0\right) - \mathbf{z}\right\|, \left\|\mathbf{z} - \left(\frac{x}{2}, 0\right)\right\|\right\}$. We must have $d_{max} \leq x$ since otherwise $d_{max} > x$ would imply

$$\begin{aligned} f\left(\left\|\left(-\frac{x}{2}, 0\right) - \mathbf{z}\right\|\right) + f\left(\left\|\mathbf{z} - \left(\frac{x}{2}, 0\right)\right\|\right) & \geq f(d_{max}) \\ & \geq f(x), \quad \text{since } f \text{ nondecreasing,} \end{aligned}$$

which contradicts the fact that $\mathbf{z} \in \widetilde{\mathcal{R}}(x)$ by the definition of $\widetilde{\mathcal{R}}(x)$. Thus, we conclude that $d_{max} \leq x$ and $\mathbf{z}$ satisfies the definition of Lune(x). □

Fact 19. $\widetilde{\mathcal{R}}(x)$ *is an open set.*

Proof. If $\widetilde{\mathcal{R}}(x) = \emptyset$, then $\widetilde{\mathcal{R}}(x)$ is open (and closed). Suppose then that $\widetilde{\mathcal{R}}(x) \neq \emptyset$. Since $\widetilde{\mathcal{R}}(x)$ is bounded by Fact 17, there exists elements of $\mathbb{R}^2$ that are not contained in $\widetilde{\mathcal{R}}(x)$. In particular, we can choose any sequence of points $\mathbf{z}_i$, satisfying $\mathbf{z}_i \notin \widetilde{\mathcal{R}}(x)$ and whose limit

is $\lim_{i\to\infty} \mathbf{z}_i = \mathbf{z}$. Let $r_x(\mathbf{z}_i)$ denote the cost of relaying from $\left(-\frac{x}{2}, 0\right)$ to $\left(\frac{x}{2}, 0\right)$ through $\mathbf{z}_i$, i.e., $r_x(\mathbf{z})$ is the left-hand side of the inequality defined in (4.11). Thus, since each $\mathbf{z}_i$ is not in the standard relay region, we must have that $r_x(\mathbf{z}_i) \geq f(x)$ for all i. It is easy to check that $r_x(\mathbf{z}_i)$ is a continuous function of $\mathbf{z}_i$. By continuity, we must have that $\lim_{i\to\infty} r_x(\mathbf{z}_i) = r_x(\lim_{i\to\infty} \mathbf{z}_i) = r_x(\mathbf{z}) \geq f(x)$, and thus $\mathbf{z}$ is also not contained in $\widetilde{\mathcal{R}}(x)$. This demonstrates that the complement of $\widetilde{\mathcal{R}}(x)$ is closed, so we must have that $\widetilde{\mathcal{R}}(x)$ is open. □

4.5.2 Relay-efficient functions and their properties

We would like to investigate conditions on x and f such that $\widetilde{\mathcal{R}}(x)$ is nonempty. Such functions that generate nonempty relay regions for certain values of x will be called *relay-efficient*.

Definition 20 (Relay-efficient function)**.** *Let $f : \mathbb{R}^+ \to \mathbb{R}^+$ be a continuous, convex, non-decreasing function. We say that $f(x)$ is a* relay-efficient function *if there exists some finite* efficiency threshold $x^* \geq 0$ *generating a nonempty relay region $\widetilde{\mathcal{R}}(x) \neq \emptyset$ for all larger x.*

We would like to determine sufficient conditions for a a function to be relay-efficient, and also determine some conditions for when a function is not relay efficient. We begin by noting a specific family of functions that are *not* relay-efficient.

Fact 21 (Affine functions are not relay-efficient)**.** *For any $m \geq 0$ and $b \geq 0$, the affine function $f(x) = mx + b$ is not relay efficient.*

Proof. For all $\mathbf{z} \in \mathbb{R}^2$, we have that

$$\begin{aligned} f(\|\mathbf{x}-\mathbf{z}\|) + f(\|\mathbf{z}-\mathbf{y}\|) &= \|\mathbf{x}-\mathbf{z}\| + b + \|\mathbf{z}-\mathbf{y}\| + b \\ &\geq \|\mathbf{x}-\mathbf{y}\| + 2b, \quad \text{by tri. ineq.} \\ &\geq \|\mathbf{x}-\mathbf{y}\| + b \\ &= f(\|\mathbf{x}-\mathbf{y}\|). \end{aligned}$$

Therefore, the definition of a relay region is never satisfied, so $\widetilde{\mathcal{R}}(x) = \emptyset$, and $f(x)$ is not relay efficient. □

To see what other functions are or are not relay-efficient, we begin by noting that under reasonable assumptions on f, the cost of relaying is always lower bounded by the cost of relaying through the midpoint between two sensors. In the case of $\widetilde{\mathcal{R}}(x)$, this is the origin.

Fact 22. *Suppose $f : \mathbb{R}^+ \to \mathbb{R}^+$ is a continuous, convex, nondecreasing function. For any $\mathbf{z} \in \mathbb{R}^2$, the cost of relaying from $(-\frac{x}{2}, 0)$ to $(\frac{x}{2}, 0)$ through $\mathbf{z}$ is lower bounded by the cost of relaying through the origin. Mathematically, we have*

$$f\left(\left\|\left(-\frac{x}{2},0\right)-\mathbf{z}\right\|\right)+f\left(\left\|\mathbf{z}-\left(\frac{x}{2},0\right)\right\|\right)\geq 2f\left(\frac{x}{2}\right). \tag{4.12}$$

Proof. We consider two cases. In the first case, let $\mathbf{z} \in \{(z_1, z_2) \in \mathbb{R}^2 : |z_1| \leq \frac{x}{2}\}$. Then

$$\begin{aligned}
f\left(\left\|\left(-\frac{x}{2},0\right)-\mathbf{z}\right\|\right)+&f\left(\left\|\mathbf{z}-\left(\frac{x}{2},0\right)\right\|\right)\\
&\geq f\left(\left\|\left(-\frac{x}{2},0\right)-(z_1,0)\right\|\right)+f\left(\left\|(z_1,0)-\left(\frac{x}{2},0\right)\right\|\right)\\
&= f(|\tfrac{1}{2}x+z_1|)+f(|z_1-\tfrac{1}{2}x|)\\
&= f(|\tfrac{1}{2}x+z_1|)+f(|\tfrac{1}{2}x-z_1|)\\
&= f(\tfrac{1}{2}x+z_1)+f(\tfrac{1}{2}x-z_1) \quad \text{since } x/2 \geq z_1\\
&= 2\left[\tfrac{1}{2}f(\tfrac{1}{2}x+z_1)+\tfrac{1}{2}f(\tfrac{1}{2}x-z_1)\right]\\
&\geq 2f(\tfrac{1}{2}[(\tfrac{1}{2}x+z_1)+(\tfrac{1}{2}x-z_1)]), \quad \text{by convexity}\\
&= 2f(\tfrac{x}{2}).
\end{aligned}$$

In the second case $\mathbf{z} \in \{(z_1, z_2) \in \mathbb{R}^2 : |z_1| > \frac{x}{2}\}$. The result follows by noting it is simply cheaper to skip the first hop and only use the second hop. Mathematically, this is represented as

$$\begin{aligned}
f\left(\left\|\left(-\frac{x}{2},0\right)-\mathbf{z}\right\|\right)+&f\left(\left\|\mathbf{z}-\left(\frac{x}{2},0\right)\right\|\right)\\
&\geq f\left(\left\|\left(-\frac{x}{2},0\right)-\left(-\frac{x}{2},0\right)\right\|\right)+f\left(\left\|\left(-\frac{x}{2},0\right)-\left(\frac{x}{2},0\right)\right\|\right)\\
&= f(0)+f(x)\\
&\geq 2f(\tfrac{x}{2}), \quad \text{by above result with } \mathbf{z}=(\pm\tfrac{x}{2},0).
\end{aligned}$$

□

As a result of this lower bound, it is important to note that $(0, 0) \in \widetilde{\mathcal{R}}(x)$ is a necessary and sufficient condition for $\widetilde{\mathcal{R}}(x)$ being nonempty.

Corollary 23. *Suppose $f : \mathbb{R}^+ \to \mathbb{R}^+$ is a continuous, convex, nondecreasing function. $\widetilde{\mathcal{R}}(x) \neq \emptyset$ if and only if $(0, 0) \in \widetilde{\mathcal{R}}(x)$.*

Proof. If $(0,0) \in \widetilde{\mathcal{R}}(x)$, then we trivially have that $\widetilde{\mathcal{R}}(x) \neq \emptyset$. To show the converse,, assume that $\widetilde{\mathcal{R}}(x)$ is nonempty, and let $\mathbf{z} \in \widetilde{\mathcal{R}}(x)$. By Fact 22, the cost of relaying through $\mathbf{z}$ is lower bounded by the cost of relaying through the origin. Combining these facts, we get

$$f(x) > f\left(\left\|\left(-\frac{x}{2}, 0\right) - \mathbf{z}\right\|\right) + f\left(\left\|\mathbf{z} - \left(\frac{x}{2}, 0\right)\right\|\right) \geq 2f\left(\frac{x}{2}\right)$$

which demonstrates that the origin satisfies the definition of the standard relay region (4.11). □

Due to the result of Corollary 23, it will be useful to define a function that models the cost of relaying through the midpoint of two sensors, and then compare it to $f(x)$.

Definition 24 (Midpoint relay function)**.** *Suppose $f : \mathbb{R}^+ \to \mathbb{R}^+$ is a continuous, convex, nondecreasing function. Define the* midpoint relay function *r corresponding to f to be*

$$r(x) \triangleq 2f\left(\frac{x}{2}\right). \tag{4.13}$$

We note that $r(x)$ must also be continuous, convex, and nondecreasing. Later on in this section, we will also find it useful to model the cost of relaying through other points on the line segment connecting two sensors.

Definition 25 (ε-relay function)**.** *Suppose $f : \mathbb{R}^+ \to \mathbb{R}^+$ is a continuous, convex, nondecreasing function with corresponding midpoint relay function r. For any $\varepsilon \in [0, 1)$, the* ε-relay *function $r_\varepsilon(x)$ is defined as*

$$r_\varepsilon(x) \triangleq f\left(\frac{(1-\varepsilon)x}{2}\right) + f\left(\frac{(1+\varepsilon)x}{2}\right), \tag{4.14}$$

or equivalently,

$$r_\varepsilon(x) = \frac{1}{2}r\left((1-\varepsilon)x\right) + \frac{1}{2}r\left((1+\varepsilon)x\right). \tag{4.15}$$

We note that r_ε must also be continuous, convex, and nondecreasing. Also note that $r_0(x) = r(x)$ in the special case of $\varepsilon = 0$.

We would now like to compare how the cost of direct transmission $f(x)$ compares to the cost of relaying through the midpoint $r(x)$. Note that at zero, these functions satisfy $r(0) = 2f(0) \geq f(0)$. Since $f(x)$ is nonnegative, either these functions have the same cost of transmission at $x = 0$, or it is more expensive to relay. Thus, it will be useful to show that the difference between $f(x)$ and the midpoint relay function $r(x)$ is nondecreasing. This monotonicity will eventually help us determine if and when the functions are guaranteed to

intersect. However, before we do so, the following equivalent definition of a convex function will be useful for remaining analysis.

Fact 26 (Nondecreasing slope of secant lines for convex functions)**.** *Let f be defined on an interval containing x_1, x_2. f is convex if and only if the function*

$$M(x_1, x_2) = \frac{f(x_1) - f(x_2)}{x_1 - x_2} \tag{4.16}$$

is nondecreasing in x_1 for fixed x_2, and vice-versa. We note that $M(x_1, x_2)$ is the slope of the secant line connecting $(x_1, f(x_1))$ with $(x_2, f(x_2))$.

Proof. This is an equivalent definition for a convex function. □

Corollary 27. *Let $f : \mathbb{R}^+ \to \mathbb{R}^+$ be a convex function that is not identically zero for all x. Then $\lim_{x\to\infty} \frac{f(x)}{x}$ exists and satisfies $\lim_{x\to\infty} \frac{f(x)}{x} > 0$.*

Proof. By Fact 26, for $x > 0$, the function $M(x, 0) = \frac{f(x)-f(0)}{x-0}$ is nondecreasing in x. Since f is not identically zero, then $M(x, 0)$ must be positive for some $x' \geq 0$. Thus, $M(x, 0)$ has a limit, and it must be either some positive number C or ∞.

Now, since $\frac{f(x)}{x}$ can be rewritten as

$$\begin{aligned}\frac{f(x)}{x} &= \frac{f(x) - f(0)}{x - 0} + \frac{f(0)}{x} \\ &= M(x, 0) + \frac{f(0)}{x},\end{aligned}$$

and the limit of the RHS is either some positive C or ∞, we must have that the limit of $\frac{f(x)}{x}$ exists and is either some positive C or ∞. □

Fact 28. *Let $f : \mathbb{R}^+ \to \mathbb{R}^+$ be a continuous, convex, nondecreasing function with corresponding relay function r. For any $y > x \geq 0$, the difference function*

$$g(x) = f(x) - r(x).$$

satisfies

$$\frac{g(y) - g(x)}{y - x} \geq 0, \quad \forall y > x \geq 0. \tag{4.17}$$

Additionally, (4.17) *implies that g is nondecreasing since $y > x$ and* (4.17) *together imply*

$$g(y) - g(x) \geq 0, \quad \forall y > x \geq 0. \tag{4.18}$$

Proof. We have that

$$\begin{aligned}
\frac{g(y)-g(x)}{y-x} &= \frac{f(y)-2f(\frac{y}{2})-f(x)+2f(\frac{x}{2})}{y-x} \\
&= \frac{f(y)-f(x)}{y-x} - \frac{2f(\frac{y}{2})-2f(\frac{x}{2})}{y-x} \\
&= \frac{f(y)-f(x)}{y-x} - \frac{f(\frac{y}{2})-f(\frac{x}{2})}{\frac{y}{2}-\frac{x}{2}} \\
&\geq \frac{f(y)-f(x)}{y-x} - \frac{f(y)-f(\frac{x}{2})}{y-\frac{x}{2}} \quad \text{by Fact 26} \\
&\geq \frac{f(y)-f(x)}{y-x} - \frac{f(y)-f(x)}{y-x} \quad \text{by Fact 26} \\
&= 0.
\end{aligned}$$

The nondecreasing property 4.18 follows from multiplying both sides of the above by $(y-x) > 0$. □

From Fact 28, if $r(x) < f(x)$, then any larger $x' > x$ must also satisfy $r(x') < f(x')$.

Fact 29. *Let $f : \mathbb{R}^+ \to \mathbb{R}^+$ be a continuous, convex, nondecreasing function. If $\widetilde{\mathcal{R}}(x) \neq \emptyset$ for some x, then for any $x' > x$, we must also have that $\widetilde{\mathcal{R}}(x') \neq \emptyset$.*

Proof. Since $g(x) = f(x) - r(x)$ is nondecreasing by Fact 28, $g(x) = f(x) - r(x) > 0$, then we must also have that $g(x') = f(x') - r(x') > 0$ for all $x' > x$, which implies $f(x') > r(x')$, which implies that the origin must be contained in $\widetilde{\mathcal{R}}(x')$. □

From the previous fact, if $\widetilde{\mathcal{R}}(x)$ is nonempty for some x, then there must exist a unique smallest x^* such that it will be strictly better to relay if x is bigger than x^*, and it will be as good or worse to relay if $x \leq x^*$. If there exists no such x^*, then we will define $x^* = \infty$.

Definition 30 (Efficiency threshold for relay-efficient function)**.** *Let $f : \mathbb{R}^+ \to \mathbb{R}^+$ be a continuous, convex, nondecreasing function. Define the* efficiency threshold *x^* to be the largest x such that the relay region is empty:*

$$x^* = \sup\{x \geq 0 : r(x) \geq f(x)\}. \tag{4.19}$$

Note that x^* is well defined since $x = 0$ is always in the set. Note that x^* is finite if and only if f is relay-efficient.

Proof. By Fact 29. If x^*, is finite, then all larger values of x will generate a nonempty relay region, and thus f is relay-efficient by definition. If x^* is infinite, the relay-region will always be empty, and thus f is not relay-efficient. □

Fact 31 (Properties of the efficiency threshold)**.** *Let $f : \mathbb{R}^+ \to \mathbb{R}^+$ be a continuous, convex, nondecreasing function with efficiency threshold x^*. The efficiency threshold x^* satisfies the following properties:*

1. *If x^* is finite, then x^* satisfies the equality*

$$2f\left(\frac{x^*}{2}\right) = r(x^*) = f(x^*), \quad x^* \text{ finite.} \tag{4.20}$$

2. *All larger values of x satisfy*

$$2f\left(\frac{x}{2}\right) = r(x) < f(x), \quad x > x^*. \tag{4.21}$$

3. *All smaller values of x satisfy*

$$2f\left(\frac{x}{2}\right) = r(x) \geq f(x), \quad 0 \leq x < x^*. \tag{4.22}$$

4. *All smaller values of x satisfy*

$$\frac{f(\frac{x}{2}) - f(0)}{\frac{x}{2}} \leq \frac{f(x) - f(0)}{x} \leq \frac{f(\frac{x}{2}) - f(0)}{\frac{x}{2}} + \frac{f(0)}{x}, \quad 0 < x < x^*. \tag{4.23}$$

5. *$f(x)$ is increasing for $x > x^*$.*[4]

6. *If $f(0) = 0$ and $f(x)$ is not linear on $[0, x']$ for every $x' > 0$, then $x^* = 0$.*

7. *If $f(0) = 0$ and $f(x)$ is strictly convex on $x > 0$, then $x^* = 0$.*

Proof. **Item 1:** From Fact 28, we know that $g(x) = f(x) - r(x)$ is nondecreasing, and it must also be continuous since both f and r are continuous. For Item 1, assume that x^* is finite. By continuity, we must have that $g(x^*) = 0$, which implies $r(x^*) = f(x^*)$. To see why, note that $g(x^*) = c > 0$ will violate the continuity of $g(x)$. Specifically, by definition of supremum, for any $0 < \varepsilon \leq x^*$, we have that $x^* - \varepsilon$ must be in the set $\{x \geq 0 : r(x) \geq f(x)$, which is equivalent to $g(x^* - \varepsilon) \leq 0$. However, this implies that there is a discontinuity of at least c at x^*, which violates our continuity assumption.

Items 2 and 3: From Item 1, we have that x^* finite implies that $x^* \in \{x \geq 0 : r(x) \geq f(x)\}$, so this set is either a closed interval of the form $[0, x^*]$ if x^* is finite, or

[4]Although this may seem obvious, we have to account for the fact that $f(x)$ may be constant on the interval $[0, c]$ for some $c > 0$ while still satisfying convexity and nondecreasing properties.

the interval $[0, \infty)$ if x^* is infinite, which implies (4.22) in Item 3. Similarly, since the set $\{x \geq 0 : r(x) < f(x)\}$ is the complement of the above set, it must either be an interval of the form (x^*, ∞) if x^* is finite, or $\emptyset$ if x^* is infinite, which implies (4.21).

Item 4: The lower bound follows directly from convexity, as noted in Fact 26. For the upper bound, since $x < x^*$, we can rearrange (4.22) to obtain

$$\begin{aligned}
f(x) \leq r(x) = 2f\left(\frac{x}{2}\right) \\
\Rightarrow \frac{f(x)}{2} \leq f\left(\frac{x}{2}\right) \\
\Rightarrow \frac{f(x) - 2f(0)}{2} \leq f\left(\frac{x}{2}\right) - f(0) \\
\Rightarrow \frac{f(x) - 2f(0)}{2x} \leq \frac{f(\frac{x}{2}) - f(0)}{x} \\
\Rightarrow \frac{f(x) - 2f(0)}{x} \leq \frac{f(\frac{x}{2}) - f(0)}{\frac{x}{2}} \\
\Rightarrow \frac{f(x) - f(0)}{x} - \frac{f(0)}{x} \leq \frac{f(\frac{x}{2}) - f(0)}{\frac{x}{2}} \\
\Rightarrow \frac{f(x) - f(0)}{x} \leq \frac{f(\frac{x}{2}) - f(0)}{\frac{x}{2}} + \frac{f(0)}{x}.
\end{aligned}$$

Item 5: For contradiction, suppose f is not strictly increasing for $x > x^*$. Since f is convex and nondecreasing, we must have that f is constant for some interval after x^*, i.e., there exists some $\varepsilon > 0$ such that $f(x^*) = f(x^* + \varepsilon)$. By (4.20), we have that $f(x^* + \varepsilon) = r(x^*)$. Furthermore, $g(x^* + \varepsilon) = f(x^* + \varepsilon) - r(x^* + \varepsilon) > 0$ by (4.21). Substituting yields $r(x^*) - r(x^* + \varepsilon > 0)$, which implies that $r(x)$ is decreasing, which is a contradiction. Thus, we must have that f is monotonically increasing for $x > x^*$.

Item 6: Since $f(0) = 0 = r(0)$, then $x = 0$ satisfies $f(x) \leq r(x)$. For any larger $x > 0$, since $f(x)$ is not linear on $[0, x]$, it must be strictly convex on this interval. Strict convexity implies

$$\begin{aligned}
\frac{1}{2}f(x) + \frac{1}{2}f(0) > f\left(\frac{x}{2}\right) \\
\Rightarrow \frac{1}{2}f(x) > f\left(\frac{x}{2}\right), \quad \text{since } f(x) = 0 \\
\Rightarrow f(x) > 2f\left(\frac{x}{2}\right) \\
\Rightarrow f(x) > r(x),
\end{aligned}$$

and we must have that $x^* < x$. Thus, we must have that $x^* = 0$.

Item 7: Follows immediately from Item 6, since strictly convex functions are not linear.

□

Fact 32 (Distance-scaling property of relay-efficient functions)**.** *Let $f : \mathbb{R}^+ \to \mathbb{R}^+$ be a continuous, convex, nondecreasing, relay-efficient function with efficiency threshold x^*. For any $a > 0$, the* distance-scaled function $\widetilde{f} : \mathbb{R}^+ \to \mathbb{R}^+$ *defined by $\widetilde{f}(x) = f(ax)$ is also continuous, convex, nondecreasing, and relay-efficient with efficiency threshold $\widetilde{x}^* = \frac{x^*}{a}$.*

Proof. By properties of continuous, convex, nondecreasing functions, it is easy to see that $\widetilde{f}$ is also continuous, convex, and nondecreasing. Define $\widetilde{r}(x) = 2\widetilde{f}\left(\frac{x}{2}\right)$. It is easy to see that $\widetilde{x}^*$ satisfies

$$\begin{aligned}
\widetilde{r}(\widetilde{x}^*) &= 2\widetilde{f}\left(\frac{\widetilde{x}^*}{2}\right) \\
&= 2f\left(a\frac{x^*}{2a}\right) \\
&= 2f\left(\frac{x^*}{2}\right) \\
&= f\left(x^*\right) \\
&= f\left(a\frac{x^*}{a}\right) \\
&= f\left(a\widetilde{x}^*\right) \\
&= \widetilde{f}(\widetilde{x}^*),
\end{aligned}$$

and from a similar argument, it can be shown that $\widetilde{r}(x) < \widetilde{f}(x)$ for all larger $x > \widetilde{x}^*$ and $\widetilde{r}(x) \geq \widetilde{f}(x)$ for all smaller $0 \leq x \leq x^*$. Thus, $\widetilde{f}(x)$ is relay efficient with efficiency threshold $\widetilde{x}^*$. □

Fact 33 (Linear combinations of distance-scaled relay-efficient functions are relay-efficient)**.** *Let $f_i, i = 1, \cdots, n$ be finite set of continuous, convex, nondecreasing, relay-efficient functions with efficiency threshold x_i^*, and let $a_i, i = 1, \ldots, n$, and $b_i, i = 1, \ldots, n$ be sequences of positive numbers. The function*

$$f(x) = \sum_{i=1}^{n} b_i f_i(a_i x)$$

is relay efficient whose efficiency threshold satisfies

$$0 \leq x^* \leq \max_{i=1,\ldots,n} \left\{ \frac{x_i^*}{a_i} \right\}.$$

Proof. By properties of continuous, convex, and nondecreasing functions, f must also be continuous, convex and nondecreasing. By Fact 32, for each i we have that each $f_i(a_i x_i)$ must be relay efficient. To see that f is relay efficient, take any $x > \max_{i=1,\ldots,n} \left\{ \frac{x_i^*}{a_i} \right\}$, and check its midpoint relay function to obtain

$$\begin{aligned} r(x) &\triangleq 2f\left(\frac{x}{2}\right) \\ &= \sum_{i=1}^{n} 2b_i f_i\left(a_i \frac{x}{2}\right) \\ &= \sum_{i=1}^{n} r_i\left(a_i x\right) \\ &< \sum_{i=1}^{n} f_i\left(a_i x\right), \quad \text{since } x > \max_{i=1,\ldots,n} \left\{ \frac{x_i^*}{a_i} \right\} \\ &= f(x), \end{aligned}$$

which implies that $\widetilde{\mathcal{R}}(x)$ is not empty, and that there exists a finite efficiency threshold x^* that must be equal to or smaller than $\max_{i=1,\ldots,n} \left\{ \frac{x_i^*}{a_i} \right\}$. □

Corollary 34. *Let $f : \mathbb{R}^+ \to \mathbb{R}^+$ be a continuous, convex, nondecreasing function, and suppose that f is relay efficient with efficiency threshold x^*. Then for every $\varepsilon \in [0, 1)$, there exists a finite value x_ε^*, such that $x > x_\varepsilon^*$ implies that it is cheaper to relay through a sensor located at $\left(\pm\frac{\varepsilon}{2}x, 0\right)$, i.e., the ε-relay function r_ε satisfies*

$$f\left(\frac{(1-\varepsilon)x}{2}\right) + f\left(\frac{(1+\varepsilon)x}{2}\right) = r_\varepsilon(x) < f(x), \quad x > x_\varepsilon^*.$$

and thus the points $\left(\pm\frac{\varepsilon}{2}x, 0\right)$ must be contained in $\widetilde{\mathcal{R}}(x)$. Furthermore, the threshold x_ε^* is upper bounded by

$$x_\varepsilon^* \leq \frac{2x^*}{1-\varepsilon}. \tag{4.24}$$

Finally, for all $0 \leq \varepsilon' < \varepsilon$, if $x > x_\varepsilon^*$, then $\left(\pm\frac{\varepsilon'}{2}x, 0\right) \in \widetilde{\mathcal{R}}(x)$.

Proof. Since r_ε is a linear combination of scaled relay-efficient functions, the result follows directly from Fact 33, since the efficiency threshold must exist and satisfies

$$x_\varepsilon^* \leq \max\left\{\frac{2x^*}{1+\varepsilon}, \frac{2x^*}{1-\varepsilon}\right\} = \frac{2x^*}{1-\varepsilon}.$$

Furthermore, by the definition of the standard relay region, since $r_\varepsilon(x) < f(x)$, and $r_\varepsilon(x)$ is the cost of relaying through $\left(\pm\frac{\varepsilon}{2}x, 0\right)$, then these two points must be contained in $\widetilde{\mathcal{R}}(x)$.

Finally, by the convexity of $\widetilde{\mathcal{R}}(x)$, for all $0 \leq \varepsilon' < \varepsilon$, if $x > x^*_\varepsilon$, then $\left(\pm\frac{\varepsilon'}{2}x, 0\right) \in \widetilde{\mathcal{R}}(x)$. □

Fact 35 (Sufficient conditions for relay efficiency)**.** *Let $f : \mathbb{R}^+ \to \mathbb{R}^+$ be a continuous, convex, nondecreasing function. f is relay efficient if any of the following are true:*

(a) $\lim_{x\to\infty} \frac{f(x)}{x} = \infty$*, or*

(b) $f(0) = 0$ *and f is not a linear function.*[5]

Proof. For (a), if f satisfies $\lim_{x\to\infty} \frac{f(x)}{x} = \infty$, then we must have that x^* is finite. To see why, using a proof by contradiction, suppose x^* is not finite. Then all $x > 0$ must satisfy the inequality (4.23). However, rearranging inequality (4.23) yields

$$\frac{f(\frac{x}{2})}{\frac{x}{2}} - \frac{f(0)}{x} \leq \frac{f(x)}{x} \leq \frac{f(\frac{x}{2})}{\frac{x}{2}}.$$

If we recursively apply the upper bound of this inequality, we get

$$\frac{f(\frac{x}{2})}{\frac{x}{2}} \geq \frac{f(x)}{x} \geq \frac{f(2x)}{2x} \geq \frac{f(4x)}{4x} \geq \cdots$$

and we see that it is impossible to have $\lim_{x\to\infty} \frac{f(x)}{x} = \infty$. Thus, we must have that x^* is finite, and thereby f is relay efficient.

For (b) we have that $f(0) = 0$ and $f(x)$ is not a linear function Once again, for contradiction, suppose x^* is infinite so that the inequality (4.23) holds for all $x > 0$. If we apply $f(0) = 0$ to the inequality (4.23) we get

$$\frac{f(\frac{x}{2})}{\frac{x}{2}} \leq \frac{f(x)}{x} \leq \frac{f(\frac{x}{2})}{\frac{x}{2}}.$$

which implies the equality

$$\frac{f(\frac{x}{2})}{\frac{x}{2}} = \frac{f(x)}{x}$$

which reduces to $2f(x/2) = f(x)$ for all $x > 0$. By the definition of a convex function, since this holds for all $x > 0$, we must have that f is linear, which contradicts our original assumption that f is not linear. We thus conclude that x^* must be finite. □

[5]We tried to show that these were also necessary conditions, but there were some counterexamples. See the later section on examples/counterexamples.

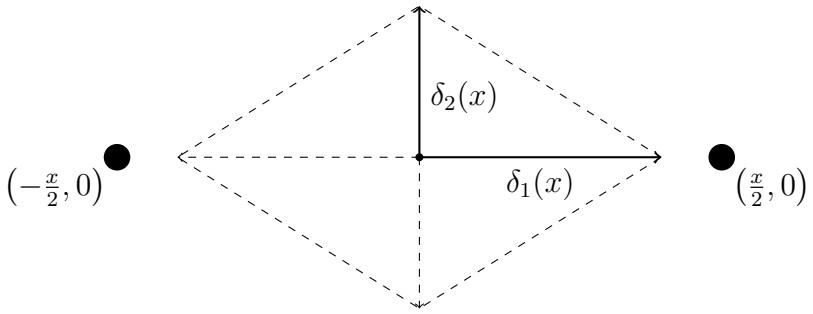


Figure 4.10: $\mathcal{D}(x;\delta_1(x),\delta_2(x))$

4.5.3 The size and shape of $\widetilde{\mathcal{R}}(x)$

In this section, for relay-efficient f, we want to see how the size and shape of $\widetilde{\mathcal{R}}(x)$ behaves as x grows. This will be useful in demonstrating that for large distances x, the relay region must contain a sensor, e.g., the relay region must contain a sensor in a lattice sensor network.

Definition 36 (Strongly relay-efficient function)**.** *Let $f:\mathbb{R}^+\to\mathbb{R}^+$ be a continuous, convex, nondecreasing function. f is* strongly relay-efficient *if it is relay efficient and its corresponding relay region satisfies*

$$\liminf_{x\to\infty}\frac{|\widetilde{\mathcal{R}}(x)|}{x^2}>0.$$

Alternatively, f is strongly relay efficient if the relay region $\widetilde{\mathcal{R}}(x)$ grows as $\Theta(x^2)$.

It will be easy to analyze the limiting behavior of $\widetilde{\mathcal{R}}(x)$ if we can identify a diamond-shaped subset of $\widetilde{\mathcal{R}}(x)$ of the form

$$\mathcal{D}(x;\delta_1(x),\delta_2(x))=\mathrm{Conv}\left\{(-\delta_1(x),0),(\delta_1(x),0),(0,-\delta_2(x)),(0,\delta_2(x))\right\}, \tag{4.25}$$

where $\delta_1(x)$ and $\delta_2(x)$ and positive-valued functions, and $\mathrm{Conv}(S)$ denotes the convex hull of S. By Fact 16, if the four points $\{(-\delta_1(x),0),(\delta_1(x),0),(0,-\delta_2(x)),(0,\delta_2(x))\}$ are contained in $\widetilde{\mathcal{R}}(x)$, then the convex hull is also contained in $\widetilde{\mathcal{R}}(x)$. Consequently, the area of $\widetilde{\mathcal{R}}(x)$ is lower bounded by the area of $\mathcal{D}(x;\delta_1(x),\delta_2(x)$, which is

$$|\mathcal{D}(x;\delta_1(x),\delta_2(x))|=2\delta_1(x)\delta_2(x)$$

Thus, to demonstrate strong relay-efficiency, it will be sufficient to show that both $\delta_1(x)$ and $\delta_2(x)$ are of order x. Note that the rectangular bounds of Fact 17 imply that δ_1 must satisfy $\delta_1(x)<\frac{x}{2}$, and $\delta_2(x)$ must satisfy $\delta_2(x)<\frac{\sqrt{3}}{2}$. An depiction of $\mathcal{D}(x;\delta_1(x),\delta_2(x))$ is shown in Figure 4.10.

4.5.3.1 Horizontal width $\delta_1(x)$

We begin by trying to find a horizontal width $\delta_1(x)$ such that the points $\pm(\delta_1(x), 0)$ are contained in the standard relay region.

Fact 37. *Let $f : \mathbb{R}^+ \to \mathbb{R}^+$ be a continuous, convex, nondecreasing function that is relay-efficient with efficiency threshold x^*. For $x \geq 4x^*$, define*

$$\delta_1(x) = \frac{x}{2} - 2x^*, \quad x \geq 4x^*. \tag{4.26}$$

Then the points $(\pm\delta_1(x), 0)$ are contained in $\widetilde{\mathcal{R}}(x)$.

Proof. First, the points $(\pm\delta_1(x), 0)$ can be rewritten as $\left(\pm\frac{\varepsilon(x)}{2}x, 0\right)$, where

$$\varepsilon(x) = \frac{2\delta_1(x)}{x}, \quad x \geq 4x^*.$$

It is easy to see that $\varepsilon(x) \in [0, 1)$ for all $x \geq 4x^*$, since it can be rewritten as

$$\varepsilon(x) = 1 - \frac{4x^*}{x}, \quad x \geq 4x^*.$$

Therefore, by Corollary 34, for every $\varepsilon(x) \in [0, 1)$, there must exist an $x^*_{\varepsilon(x)}$ such that $x' > x^*_{\varepsilon(x)}$ implies $\left(\pm\frac{\varepsilon(x)}{2}x', 0\right)$ are contained in $\widetilde{\mathcal{R}}(x')$. We now show that x always satisfies $x > x^*_{\varepsilon(x)}$, so that the points $(\pm\delta_1(x), 0)$ are always contained in $\widetilde{\mathcal{R}}(x)$. Beginning with the upper bound (4.24), we have

$$x^*_{\varepsilon(x)} \leq \frac{2x^*}{1 - \varepsilon(x)} = \frac{2x^*}{1 - 1 + 4\frac{x^*}{x}} = \frac{2x^*}{4x^*}x = \frac{x}{2} < x.$$

Since $x > x^*_{\varepsilon(x)}$ for all $x \geq x^*$, we must have that $(\pm\delta_1(x), 0)$ is contained in $\widetilde{\mathcal{R}}(x)$. □

It will also be interesting to see how $\delta_1(x)$ behaves in the limit.

Corollary 38. *Let $f : \mathbb{R}^+ \to \mathbb{R}^+$ be a continuous, convex, nondecreasing function that is relay-efficient with efficiency threshold x^*, and let $\delta_1(x)$ be defined as in (4.26). Then as x becomes large, we have that*

$$\lim_{x\to\infty} \frac{\delta_1(x)}{x} = \frac{1}{2}. \tag{4.27}$$

Thus, in the limit as x becomes large, the relay region $\widetilde{\mathcal{R}}(x)$ contains the interval $\left(-\frac{x}{2}, \frac{x}{2}\right)$ on the horizontal axis.

Proof. Follows directly from applying the limit to (4.26) and the fact that all points between $\delta_1(x)$ and the origin must be contained in $\widetilde{\mathcal{R}}(x)$ by the convexity of $\widetilde{\mathcal{R}}(x)$. □

4.5.3.2 Vertical height $\delta_2(x)$

We would now like to find a vertical height function $\delta_2(x)$. It will first be useful to state a few facts.

Fact 39. *Let $f : \mathbb{R}^+ \to \mathbb{R}^+$ be a continuous, convex, nondecreasing function, and suppose $f(x)$ is relay-efficient with efficiency threshold x^*. The following are true:*

(a) f is increasing for $x > x^*$ (same as Fact 31.5).

(b) For $y \geq f(x^*)$, the inverse function $f^{-1}(y)$ exists and is continuous, increasing, and concave.

(c) For $x \geq f^{-1}(2f(x^*))$, define the hypotenuse function[6]*.*

$$h(x) = f^{-1}\left(\frac{f(x)}{2}\right), \quad x \geq f^{-1}(2f(x^*)) \tag{4.28}$$

is well-defined, continuous, and increasing.

(d) For $x \geq f^{-1}(2f(x^*))$, the compound function $h(x)$ is upper-bounded by

$$h(x) < x, \quad x \geq f^{-1}(2f(x^*)). \tag{4.29}$$

(e) For $x > x^*$, the hypotenuse function $h(x)$ is lower-bounded by

$$h(x) > \frac{x}{2}, \quad x > x^*. \tag{4.30}$$

(f) The function $\frac{h(x)}{x}$ is bounded by

$$\frac{1}{2} < \frac{h(x)}{x} < 1. \tag{4.31}$$

(g) Let $f(x)$ be a relay-efficient function that is also twice-differentiable, $f''(x)$ is nondecreasing, and $f''(x) > 0$ for sufficiently large x. Additionally, suppose $f(x)$ and its derivatives satisfy $[f'(x)]^2 - f(x)f''(x) \geq 0$. Then

$$h(x) \geq x - \frac{f'(x)}{f''(x)} + \sqrt{\left(\frac{f'(x)}{f''(x)}\right)^2 - \frac{f(x)}{f''(x)}} \tag{4.32}$$

[6]The term "hypotenuse" refers to the fact that $h(x)$ is exactly the length of the hypotenuse of the right triangle formed by $(0, \delta_2(x))$ and $(\frac{x}{2}, 0)$

We note that $f^{-1}(y)$ may exist for smaller values of y than those noted here. However, since we will be looking at large values of y, and so course lower bounds on the existence of $f^{-1}(y)$ will suffice.

Proof. (a) This property is repeated here for convenience. See Fact 31.5 for a proof.

(b) Follows from the fact that f is increasing, continuous and convex for $x > x^*$.

(c) Follows from combining the fact that $\frac{f(x)}{2}$ is continuous and increasing with the fact that f^{-1} is continuous and increasing. We can check that h is increasing since for any $y > x > f^{-1}(2f(x^*))$,

$$\begin{aligned} h(y) - h(x) &= f^{-1}\left(\frac{f(y)}{2}\right) - f^{-1}\left(\frac{f(x)}{2}\right) \\ &> f^{-1}\left(\frac{f(y)}{2}\right) - f^{-1}\left(\frac{f(y)}{2}\right), \quad \text{both } f \text{ and } f^{-1} \text{ increasing} \\ &= 0. \end{aligned}$$

(d) Upper bound follows from fact that $f^{-1}(y)$ increasing implies $f^{-1}(f(x)/2) < f^{-1}(f(x)) = x$.

(e) Lower bound follows from fact that $x > x^*$ implies $r(x) < f(x)$ which implies $f(x/2) < f(x)/2$. Since f^{-1} is monotonic, this further implies $x/2 < f^{-1}(f(x)/2)$.

(f) Follows from (d) and (e).

(g) First, assume x is large so that h is defined. Assume that f is twice-differentiable and f'' is nondecreasing, and that $f''(x) > 0$ for large x. We begin by taking a 2nd order Taylor expansion of f at x

$$\widetilde{f}(z) = f(x) + f'(x)(z - x) + \frac{f''(x)}{2}(z - x)^2$$

and note that for $z < x$, by convexity and the fact that f'' is nondecreasing, this must be an upper bound $f(x) \leq \widetilde{f}(x)$. Since $\widetilde{f}(z)$ this is an upper bound for $f(x)$ when $z < x$, then $\widetilde{f}^{-1}(y')$ must be a lower bound for $f^{-1}(y)$. Thus, we would like to solve $\widetilde{f}(x) = y$ for x and take the larger root to get a tighter bound. If we expand $\widetilde{f}(x) - y = 0$ we get

$$\left(\frac{f''(x)}{2}\right) z^2 + \left(f'(x) - xf''(x)\right) z + \left(f(x) - xf'(x) + \frac{f''(x)}{2}x^2 - y\right) = 0.$$

Before applying the quadratic formula, let us calculate the discriminant D:

$$\begin{aligned}
D &= (f'(x) - xf''(x))^2 - 4\frac{f''(x)}{2}\left(f(x) - xf'(x) + \frac{f''(x)}{2}x^2 - y\right) \\
&= [f'(x)]^2 - 2xf'(x)f''(x) + x^2[f''(x)]^2 - 2f''(x)\left(f(x) - xf'(x) + \frac{f''(x)}{2}x^2 - y\right) \\
&= [f'(x)]^2 - 2xf'(x)f''(x) + x^2[f''(x)]^2 \\
&\quad - 2f(x)f''(x) + 2xf'(x)f''(x) - x^2[f''(x)]^2 + 2yf''(x) \\
&= [f'(x)]^2 - 2f(x)f''(x) + 2yf''(x).
\end{aligned}$$

Now we apply the quadratic formula to obtain

$$\begin{aligned}
\widetilde{f}^{-1}(y) &= \frac{-(f'(x) - xf''(x)) + \sqrt{[f'(x)]^2 - 2f(x)f''(x) + 2yf''(x)}}{2\frac{f''(x)}{2}} \\
&= x - \frac{f'(x)}{f''(x)} + \sqrt{\left(\frac{f'(x)}{f''(x)}\right)^2 - 2\frac{(f(x) - y)}{f''(x)}}.
\end{aligned}$$

Since $h(x) = f^{-1}\left(\frac{f(x)}{2}\right)$ and $f^{-1}(y) \geq \widetilde{f}^{-1}(y)$, we can lower bound $h(x)$ using our Taylor expansion

$$\begin{aligned}
h(x) &= f^{-1}\left(\frac{f(x)}{2}\right) \\
&\geq \widetilde{f}^{-1}\left(\frac{f(x)}{2}\right) \\
&= x - \frac{f'(x)}{f''(x)} + \sqrt{\left(\frac{f'(x)}{f''(x)}\right)^2 - 2\frac{(f(x) - \frac{f(x)}{2})}{f''(x)}} \\
&= x - \frac{f'(x)}{f''(x)} + \sqrt{\left(\frac{f'(x)}{f''(x)}\right)^2 - \frac{f(x)}{f''(x)}}.
\end{aligned}$$

If $f(x)$ and its derivatives satisfy $[f'(x)]^2 - f(x)f''(x) \geq 0$, then this bound will always be well-defined.

□

We now would like to apply these results and determine how $\widetilde{\mathcal{R}}(x)$ behaves along the vertical axis.

Fact 40. *Let $f : \mathbb{R}^+ \to \mathbb{R}^+$ be a continuous, convex, nondecreasing function that is relay-*

efficient with efficiency threshold x^*. For $x \geq f^{-1}(2f(x^*)) > x^*$, define

$$\delta_2(x) = \sqrt{h^2(x) - \frac{x^2}{4}}, \quad x \geq f^{-1}(2f(x^*)). \tag{4.33}$$

Then all points on the vertical open interval $(0,0) \times (-\delta_2(x), \delta_2(x))$ are contained in $\widetilde{\mathcal{R}}(x)$. Note that since the points $(0, \pm\delta_2(x)$ are not contained in $\widetilde{\mathcal{R}}(x)$, we must have that $\delta_2(x)$ is the tightest upper bound to our relay region along the vertical axis in $\mathbb{R}^2$.

Proof. For all $\varepsilon \in [0,1)$, define

$$\widetilde{\delta}_2(x,\varepsilon) = \varepsilon\delta_2(x), \quad \varepsilon \in [0,1), x \geq f^{-1}(2f(x^*)). \tag{4.34}$$

We demonstrate that the cost of relaying through any of the points $(0, \pm\widetilde{\delta}_2(x,\varepsilon)) \in \widetilde{\mathcal{R}}(x)$ is cheaper than direct transmission. Take any $x > f^{-1}(2f(x^*))$ (which is also greater than x^*). Since these points, the origin, and the transmission sensor forms a right triangle, we have the cost of transmission is

$$\begin{aligned}
2f\left(\sqrt{\widetilde{\delta}_2^2(x,\varepsilon) + \frac{x^2}{4}}\right) &= 2f\left(\sqrt{\varepsilon^2\delta_2^2(x) + \frac{x^2}{4}}\right) \\
&< 2f\left(\sqrt{\delta_2^2(x) + \frac{x^2}{4}}\right), \quad f \text{ strictly increasing by Fact 31.5} \\
&= 2f\left(\sqrt{h^2(x) - \frac{x^2}{4} + \frac{x^2}{4}}\right) \\
&= 2f(h(x)) \\
&= 2f\left(f^{-1}\left(\frac{f(x)}{2}\right)\right) \\
&= f(x).
\end{aligned}$$

Thus, these points are contained in $\widetilde{\mathcal{R}}(x)$ by definition. □

Once again, it will be useful to see how $\delta_2(x)$ behaves in the limit. Dividing by x and taking a limit yields

$$\lim_{x\to\infty} \frac{\delta_2(x)}{x} = \sqrt{\left(\lim_{x\to\infty} \frac{h(x)}{x}\right)^2 - \frac{1}{4}}, \tag{4.35}$$

Note that this limit may not always exist since it requires that $\lim_{x\to\infty} \frac{h(x)}{x}$ exist. However we can consider a few cases where the limit of $\frac{\delta_2(x)}{x}$ does exist.

Fact 41. *Let $f : \mathbb{R}^+ \to \mathbb{R}^+$ be a continuous, convex, nondecreasing function, and suppose f is relay-efficient with efficiency threshold x^*. If $\lim_{x\to\infty} \frac{f(x)}{x} = C$ for some $C > 0$, then*

$$\lim_{x\to\infty} \frac{h(x)}{x} = \frac{1}{2},$$

which in turn implies that

$$\lim_{x\to\infty} \frac{\delta_2(x)}{x} = 0.$$

and thus $\frac{f(x)}{x}$ cannot be strongly relay-efficient.

Proof. If $\lim_{x\to\infty} \frac{f(x)}{x} = C$ for some $C > 0$, we must have that $\lim_{x\to\infty} \frac{x}{f(x)} = \frac{1}{C}$. Using several variable transforms and the fact that both $f(x)$ and $f^{-1}(y)$ diverge to infinity for large values, we have that

$$\begin{aligned}
\lim_{x\to\infty} \frac{h(x)}{x} &= \lim_{x\to\infty} \frac{f^{-1}\left(\frac{f(x)}{2}\right)}{x} \\
&= \lim_{y\to\infty} \frac{f^{-1}\left(\frac{y}{2}\right)}{f^{-1}(y)}, \quad \text{let } x = f^{-1}(y) \\
&= \frac{1}{2} \lim_{y\to\infty} \frac{f^{-1}\left(\frac{y}{2}\right)}{\frac{y}{2}} \cdot \frac{y}{f^{-1}(y)} \\
&= \frac{1}{2} \lim_{y\to\infty} \frac{f^{-1}\left(\frac{y}{2}\right)}{\frac{y}{2}} \cdot \lim_{y'\to\infty} \frac{y'}{f^{-1}(y')} \\
&= \frac{1}{2} \lim_{x\to\infty} \frac{x}{f(x)} \cdot \lim_{x'\to\infty} \frac{f(x')}{x'}, \quad \text{le } y = 2f(x) \text{ and } y' = f(x') \\
&= \frac{1}{2} \cdot \frac{1}{C} \cdot C \\
&= \frac{1}{2},
\end{aligned}$$

and thus we must have that $H = \frac{1}{2}$. The result for $\frac{\delta_2(x)}{x}$ then follows from (4.35). Since this bounds $\widetilde{\mathcal{R}}(x)$ along the vertical axis by Fact 40, we must have that $\lim_{x\to\infty} \frac{|\widetilde{\mathcal{R}}(x)|}{x^2} = 0$, and $f(x)$ is not strongly relay efficient. □

4.5.4 Energy-efficient hop lengths

So far, we have characterized several thresholds (x^* and d^*) that describe when it is possible to relay efficiently, and when a relay sensor must exist in a lattice sensor network. In this section we ask a different question: At what distance $x = \lambda^*$ is the energy-per-distance quantity $\frac{f(x)}{x}$ minimized? Ideally, if we know that sensors are going to relay through a

sequence of sensors, we would like each pair of sensors to be separated by this distance. This is of particular interest for cutset networks, where sensors are deployed along straight lines.

First, let us define a set of such "efficient" hop distances.

Definition 42 (Efficient hop lengths)**.** *Let $f : \mathbb{R}^+ \to \mathbb{R}^+$ be a continuous, convex, nondecreasing, and differentiable function. Define the set of* efficient hop lengths Λ^* *to be*

$$\Lambda^* = \left\{ x > 0 : \frac{f(x)}{x} \leq \frac{f(y)}{y} \quad \forall y > 0 \right\}. \tag{4.36}$$

A geometric interpretation of Λ^* is that any $\lambda \in \Lambda^*$ minimizes the slope of the line connecting the origin with $(x, f(x))$. As the next fact demonstrates, all efficient hops must be smaller than x^*, since otherwise we can increase our efficiency by relaying.

Fact 43 (Efficient hop lengths are smaller than efficiency threshold)**.** *Let $f : \mathbb{R}^+ \to \mathbb{R}^+$ be a continuous, convex, and nondecreasing function. Let x^* be the (possibly infinite) efficiency threshold for f. If $\lambda^* \in \Lambda^*$, then $\lambda^* \leq x^*$.*

Proof. By definition of Λ^*, we must have that

$$\frac{f(\lambda^*)}{\lambda^*} \leq \frac{f\left(\frac{\lambda^*}{2}\right)}{\frac{\lambda^*}{2}},$$

which reduces to

$$f(\lambda^*) \leq 2f\left(\frac{\lambda^*}{2}\right) = r(\lambda^*).$$

Therefore, by Fact 31, equation (4.22), we must have that $\lambda^* \leq x^*$. □

If our energy function f starts at the origin (there is no "energy overhead" to transmission), then there is no "best" distance, since smaller distances are always more efficient. This is reflected in the following fact:

Fact 44. *Let $f : \mathbb{R}^+ \to \mathbb{R}^+$ be a continuous, strictly convex, and nondecreasing function. If $f(0) = 0$, then $\Lambda^* = \emptyset$, and for any $0 < x < y$,*

$$\frac{f(x)}{x} < \frac{f(y)}{y}, \tag{4.37}$$

i.e., smaller distances are always more energy efficient.

Proof. By Fact 31, $f(0) = 0$ and $f(x)$ strictly convex implies $x^* = 0$. By Fact 43, any $\lambda^* \in \Lambda^*$ must satisfy $\lambda^* \leq x^*$, so $\lambda^* \leq 0$. However, Λ^* is only defined for positive values of x, i.e., any $\lambda^* \in \Lambda^*$ satisfies $\lambda^* > 0$, which contradicts $\lambda^* \leq x^*$. Thus, we must have

that $\Lambda^* = \emptyset$. Finally, if we apply a version of Fact 26 for strictly convex functions, we have that the slope of the secant line $M(x_1, x_2)$ is increasing in x_1 for fixed x_2. Thus, for any $0 < x < y$, we must have that $M(x, 0) < M(y, 0)$, which implies (4.37). □

We now consider the case where $f(0) > 0$. In the next fact, we give an equation that can be solved to find a single threshold λ^*. If this equation has a solution, it will be unique for strictly convex functions.

Fact 45. *Let $f : \mathbb{R}^+ \to \mathbb{R}^+$ be a continuous nondecreasing, and twice-differentiable function that is strictly convex for $x > 0$ and satisfies $f(0) > 0$. If the equation*

$$f'(\lambda^*) = \frac{f(\lambda^*)}{\lambda^*} \tag{4.38}$$

holds for some $\lambda^* > 0$, then then Λ^* is equal to a single point $\{\lambda^*\}$.

Proof. To find the minimum value(s) of $\frac{f(x)}{x}$, we take the derivative of $\frac{f(x)}{x}$ to obtain

$$\frac{d}{dx}\left\{\frac{f(x)}{x}\right\} = \frac{xf'(x) - f(x)}{x^2}.$$

Setting this equal to zero and rearranging yields

$$f'(x) = \frac{f(x)}{x}.$$

We will now demonstrate that this equation has a unique solution. If λ^* satisfies this equation, then the line tangent to f at $x = \lambda^*$ must intersect the origin. One can see this by first defining the tangent line of f at x to be

$$\ell(y, x) = f(x) + f'(x)(y - x).$$

and then noting that for any solution λ^* to (4.38) satisfies $\ell(0, \lambda^*) = 0$. Thus, we would like to see how the intercept of the tangent line behaves as x increases. Let us define $a(x)$ to be the x-intercept of the tangent line $\ell(y, x)$. We can find $a(x)$ by setting $\ell(y, x) = 0$ and solving for y as a function of x:

$$\begin{aligned} f(x) + f'(x)(y - x) &= 0 \\ \Rightarrow f'(x)y &= xf'(x) - f(x) \\ \Rightarrow y &= x - \frac{f(x)}{f'(x)}, \end{aligned}$$

yielding

$$a(x) = x - \frac{f(x)}{f'(x)}.$$

We now demonstrate that $a(x)$ is a strictly increasing function of x. Doing so implies that $a(\lambda^*) = 0$ for a unique value of λ^*. Taking the derivative of $a(x)$ yields

$$\begin{aligned}
\frac{da(x)}{dx} &= 1 - \frac{[f'(x)]^2 - f(x)f''(x)}{[f'(x)]^2} \\
&= 1 - 1 + \frac{f(x)f''(x)}{[f'(x)]^2} \\
&= \frac{f(x)f''(x)}{[f'(x)]^2} \\
&> 0, \quad \text{since } f(x) > 0,\ f'(x) > 0, \text{ and } f''(x) > 0 \text{ for all } x > 0,
\end{aligned}$$

and we see that $\frac{da(x)}{dx} > 0$ for all $x > 0$, which implies that $a(x)$ is strictly increasing. Thus, if equation (4.38) has a solution, it must be unique. □

We will solve equation (4.38) for several example functions in Section 4.5.6.

4.5.5 Truncated transmission energy functions

Sensors in a real sensor network often specify a maximum transmission distance. In this section, we analyze how having a maximum transmission distance x_{max} affects our analysis.

Suppose we let $f : \mathbb{R}^+ \to \mathbb{R}^+$ be a continuous, convex, and nondecreasing, and nondecreasing function with (possibly infinite) efficiency threshold x^*. For some $x_{max} > 0$, define the *truncated function* to be

$$\widetilde{f}(x) = \begin{cases} f(x), & x \le x_{max} \\ \infty, & x > x_{max}. \end{cases} \tag{4.39}$$

We define the set of relay efficient hops to be

$$\widetilde{\Lambda}^* = \left\{ x \in (0, x_{max}] : \frac{f(x)}{x} \le \frac{f(y)}{y}, \quad \forall y \in [0, x_{max}] \right\}. \tag{4.40}$$

Note if f satisfies the conditions of Fact 45 and λ^* exists, if $x_{max} \le \lambda^*$, then $\widetilde{\Lambda}^* = \{x_{max}\}$ since $\frac{f(x)}{x}$ is strictly decreasing by the proof of Fact 45. In summary, if a sensor is modeled by $f(x)$ over an interval and specifies a maximum transmission distance, one can compare x^* and λ^* to x_{max} to make decisions about how the sensor network should be arranged. For

example:

- $x_{max} \leq x^*$ implies that relaying does not improve energy efficiency, so least-energy paths through the network will likely try to approximate the shortest linear path with large hops.
- $x_{max} \geq x^*$ implies that relaying does improve energy efficiency, so least-energy paths through the network will not use hops that contain a potential relay sensor in the transmission pairs' relay region.
- $x_{max} \leq \lambda^*$ implies that larger hops are always more efficient than shorter hops, and transmitting at distance x_{max} is most efficient.
- $x_{max} \geq \lambda^*$ implies that neighboring sensors spaced λ^* apart are most efficient, which can be exploited when planning sensor deployment.

4.5.6 Energy Function Examples

In this section, we consider several possible energy functions and calculate relevant quantities.

4.5.6.1 Power-law (plus a constant) functions are strongly relay-efficient

Here, we demonstrate an example where our transmission function follows a power law plus a constant, which is a common assumption for sensor networks (see [62, Model I] for a brief discussion of the (inverse) power law and additional citations). For any $c \geq 0$ and $\beta \geq 2$, the power law function

$$f(x) = Px^\beta + C \tag{4.41}$$

satisfies Fact 35 since it is continuous, convex, nondecreasing and $\lim_{x\to\infty} \frac{f(x)}{x} = \infty$. Thus, any power-law function is relay-efficient by Fact 35.

The midpoint relay function is

$$r(x) = \frac{P}{2^{\beta-1}}x^\beta + 2C.$$

To obtain the efficiency threshold x^*, we set $f(x) = r(x)$ and solve for x to obtain

$$x^* = \left(\frac{C}{P} \cdot \frac{2^{\beta-1}}{2^{\beta-1}-1}\right)^{\frac{1}{\beta}}.$$

To obtain the energy-efficient efficient hop length λ^*, we first calculate the derivative

$$f'(x) = \beta P x^{\beta-1}.$$

We then set $f'(x) = \frac{f(x)}{x}$ and solve for x to obtain

$$\lambda^* = \left(\frac{C}{P} \cdot \frac{1}{\beta - 1}\right)^{\frac{1}{\beta}}.$$

Note that $\lambda^* < x^*$ for any $\beta > 1$ since $\frac{2^{\beta-1}}{2^{\beta-1}-1} = \frac{1}{1-2^{1-\beta}} > 1 > \frac{1}{\beta-1}$.

We also have the inverse function

$$f^{-1}(y) = \left(\frac{y - C}{P}\right)^{\frac{1}{\beta}}$$

and hypotenuse function

$$h(x) = \frac{1}{2^{\frac{1}{\beta}}} \cdot \left(x^\beta - \frac{C}{P}\right)^{\frac{1}{\beta}},$$

which implies $\delta_2(x)$ is

$$\delta_2(x) = \sqrt{\frac{1}{2^{\frac{2}{\beta}}} \cdot \left(x^\beta - \frac{C}{P}\right)^{\frac{2}{\beta}} - \frac{x^2}{4}},$$

and

$$\lim_{x\to\infty} \frac{\delta_2(x)}{x} = \frac{1}{2}\sqrt{4^{\frac{\beta-1}{\beta}} - 1}$$

so that

$$\lim_{x\to\infty} \frac{\mathcal{D}(x; \delta_1(x), \delta_2(x))}{x^2} = \frac{1}{2}\sqrt{4^{\frac{\beta-1}{\beta}} - 1},$$

and thus $f(x)$ is strongly relay-efficient.

"Pure" Power-law function ($C = 0$): We briefly note that in the case of a pure power-law function with no constant overhead, $x^* = \lambda^* = 0$, and therefore the relay region always exists. Furthermore $\lambda^* = 0$ implies that smaller hops are always more energy-efficient.

4.5.6.2 Exponential function

Here, as a purely mathematical exercise to demonstrate an extreme example of a strongly relay-efficient f, we briefly go over the case where f is exponential. This is not necessarily a realistic model.

Suppose for $a > 0$ and $C > 0$,

$$f(x) = Ce^{ax}.$$

The midpoint relay function is

$$r(x) = 2Ce^{\frac{a}{2}x}.$$

The efficiency threshold is

$$x^* = \frac{2\ln 2}{a}.$$

The derivative is

$$f'(x) = Cae^{ax}$$

and the energy-efficiency hop length is

$$\lambda^* = \frac{1}{a}.$$

The inverse function is

$$f^{-1}(y) = \frac{\ln\left(\frac{y}{C}\right)}{a}.$$

The hypotenuse function is

$$h(x) = x - \ln 2.$$

$\delta_2(x)$ is

$$\delta_2(x) = \sqrt{(x - \ln 2)^2 - \frac{x^2}{4}}$$

with the following limit

$$\lim_{x\to\infty} \frac{\sqrt{3}}{2}$$

so that

$$\lim_{x\to\infty} \frac{\mathcal{D}(x; \delta_1(x), \delta_2(x)}{x^2} = \frac{\sqrt{3}}{2},$$

and thus $f(x)$ is strongly relay-efficient.

4.5.6.3 Pseudo-Huber functions

As a mathematical exercise to see (a) a case where f is relay-efficient but not strongly relay efficient and (b) another case where f is not relay-efficient, we briefly describe two functions whose slope is linear in the limit as x gets large.

The first pseudo-Huber function

$$f_1(x) = \sqrt{1 + x^2} - 1$$

is continuous, convex, nondecreasing, and satisfies both $\lim_{x\to\infty} \frac{f_1(x)}{x} = 1$ and $f(0) = 0$. Thus, it is relay efficient by Fact 35, but it is not strongly relay efficient by Fact 41.

On the other hand, the (positive-valued) pseudo-Huber function

$$f_2(x) = \sqrt{1 + x^2}$$

does not satisfy $f(0) = 0$ and is not relay-efficient by Fact 35.

4.5.6.4 Initially-linear energy function

For another mathematical exercise, we briefly demonstrate a function that is relay-efficient, but no unique value satisfies $f(x) = r(x)$ due to the fact that it is initially a linear function. Consider the continuous, convex, nondecreasing function

$$f(x) = \begin{cases} x, & 0 \le x \le 1 \\ \frac{1}{2}x^2 + \frac{1}{2}, & x > 1 \\ 0, & \text{otherwise.} \end{cases}$$

Note that this function satisfies $f(x) = r(x)$ for all $x \in [0, 1]$, and $r(x) < f(x)$ for all $x > 1$. Thus, it is relay-efficient with efficiency threshold $x^* = 1$ since this is the largest value of x such that $f(x) = r(x)$. It is easy to check that this function is strongly relay-efficient since it follows a power law for large x.

4.5.7 Conclusions

We analyze continuous, convex, nondecreasing functions that are used to model energy transmission in a sensor network. For distances smaller than x^*, relaying will never be better than direct transmission. Sensors separated by distances larger than x^* can reduce the total energy consumption of the network if a relay sensor exists in the pair's relay region. Outer and inner bounds were derived for the relay region for large distances x. Furthermore, a savvy network designer than reduce the overall energy consumption by ensuring that neighboring sensors are spaced λ^*. Finally, we considered sensor models with a maximum transmission distance, and also provided several examples of energy functions $f(x)$ while calculating and relevant quantities.

4.6 Efficient Communication on a Lattice Sensor Network

In the previous section, we studied properties of transmission energy functions, and explored what it meant for such a function to be relay-efficient, including bounds on the relay region. In this section, we will explore efficient communication between sensors on a lattice sensor network. Given an transmission energy function f and a lattice, the goal in this section is to find (a) efficient paths through the lattice sensor network and different angles θ, and (b) find closed-form expressions for minimal energy costs of such paths as a function of their direction θ. It is anticipated that future work will extend this analysis to Manhattan networks, cutset networks, and other periodic sensor networks.

To begin, we denote our set of sensor locations/sites/nodes as $V = \{v_i\} \subset \mathbb{R}^2$, where V is a lattice formed by a span of linearly-independent basis vectors $[u_1, u_2]$. We define the following terms for describing communication along paths in a lattice network:

- Hop: A pair of nodes, $\widetilde{h} = (s_1, s_2)$, $s_1, s_2 \in V$.
- Hop Type: The vector difference $h = s_2 - s_1$ between two nodes in a hop, where $h = (s_1, s_2)$ and $s_1, s_2 \in V$. Note that hop types must also be elements of the lattice $h \in V$ due to group properties of the lattice.
- Path (Definition 1): A sequence of nodes $p = (s_0, \cdots, s_n)$, where each $s_i \in V$
- Path (Definition 2): An initial site and a sequence of hops $p = (s_0, \widetilde{h}_1, \cdots, \widetilde{h}_n)$ where a hop $\widetilde{h}_i = (s_i, s_{i-1})$ and each $h_i \in V$ by definition of lattice.
- Path (Definition 3): An initial site and a sequence of hop types $p = (s_0, h_1, \cdots, h_n)$.
- Net Progress of path p is
$$s_n - s_0 = \sum_{i=1}^{n} h_i = \sum_{v \in V} n_v v \tag{4.42}$$
where n_v is the integer number of times hop type v is used.
- Angle of path p is
$$\angle(s_n - s_0) = \angle \left\{ \sum_{i=1}^{n} h_i = \sum_{v \in V} n_v v \right\}, \tag{4.43}$$
where n_v is the integer number of times hop type v is used.

- Net length of p is
$$\|s_n - s_0\| = \left\| \sum_{i=1}^{n} h_i \right\| = \left\| \sum_{v \in V} n_v v \right\|. \tag{4.44}$$
where n_v is the integer number of times hop type v is used.

- Energy of path p is
$$e(p) = \sum_{i=1}^{n} f(\|h_i\|) = \sum_{v \in V} n_v f(\|v\|) \tag{4.45}$$
where n_v is the number of times hop type v is used.

- Energy from x to y via path p is
$$e(x, y, p) = f(\|x - s_0\|) + \sum_{v \in V} n_v f(\|v\|) + f\left(\left\| y - s_0 - \sum_{v \in V} n_v v \right\| \right), \tag{4.46}$$
where n_v is the integer number of times hop type v is used.

- The minimal energy from x to y is
$$e^*(x, y) = \min_{p \in P_{x,y}} e(x, y, p) = \min_{s_0 \in V, \{n_v\}} e(x, y, p) \tag{4.47}$$
where $P_{x,y}$ is the set of all possible paths from x to y in V.

- The *path type* is $\{n_v\}_{v \in V}$, i.e., the set of hop type counts.

- Let $(s_0, \{n_v\})$ be a path defined by a starting node and a path type. The optimal path type is any $\{n_v\}$ satisfying
$$e^*(x, y) = e(x, y; (s_0, \{n_v\})) = \min_{s_0 \in V, \{n_v\}} e(x, y, (s_0, \{n_v\})). \tag{4.48}$$

The high level goal of this section is to find an approximation $\mathcal{E}(x, y)$ for $e^*(x, y)$ that is a function of the distance $\|y - x\|$ and the angle $\angle(y - x)$. In particular, we believe that it will decompose as
$$\mathcal{E}(x, y) = \|x - y\| g(\angle(y - x))$$
such that the energy-per-distance will only depend on the angle through some function $g(\theta)$. Note that the net progress, angle, net length, and energy are invariant to permutations of the hop types.

4.6.1 Minimum Energy Paths

Given an energy transmission function $f(x)$ and a lattice V, for any angle θ we would like to try to estimate the minimum energy-per-distance required to transmit a message a long distance through our network at angle θ. Since any path through the network can be described by its initial site s_0 and its set of hop types $\{n_v\}$, we would like to see how the *distribution of hop types* behave for optimal, i.e., minimal energy, paths for long distances. Therefore, consider the following constrained optimization problem:

Problem 1: The Normalized Hop Type Efficiency Problem

$$g(\theta) = \min_{\{\boldsymbol{\alpha}_v\}} J(\boldsymbol{\alpha};\theta) = \min_{\{\boldsymbol{\alpha}_v\}} \frac{\sum_{v\in V} \alpha_v f(\|v\|)}{\left\|\sum_{v'\in V} \alpha_{v'} v'\right\|},$$

such that

$$\alpha_v \geq 0 \quad \forall v \in V,$$

$$\sum_{v\in V} \alpha_v = 1,$$

$$\angle\left\{\sum_{v\in V} \alpha_v v\right\} = \theta.$$

In this problem, the we would like to find a set of α_v's that describe the fraction of times that each hop types v is used in an optimal path to travel at angle θ through the network. Note that the objective function $J(\boldsymbol{\alpha};\theta)$ is in energy-per-distance. This is a useful way to view the problem of finding efficient paths for long-distance communication, but it will be easier to work with an unnormalized version, and show that the unnormalized version is equivalent to the normalized version. Thus, instead of solving Problem 1 for a fixed angle θ, consider another problem where we fix a vector u and minimize the energy-per-distance needed to traverse a net distance of $\|u\|$ at angle $\angle u$:

Problem 2: The Unnormalized Hop Type Efficiency Problem

$$\widetilde{g}(u) = \min_{\{\mathbf{a}_v\}} \widetilde{J}(\mathbf{a};u) = \min_{\{\mathbf{a}_v\}} \sum_{v\in V} a_v f(\|v\|),$$

such that

$$a_v \geq 0 \quad \forall v \in V,$$

$$\sum_{v\in V} a_v v = u.$$

In fact, it is easy to check that if $\theta = \angle u$, then Problems 1 and 2 are equivalent, i.e., a solution to one yields a solution to the other. If $\theta = \angle u$, then a solution to Problem 1 can be used to obtain a solution to Problem 2 using the change-of-variables

$$a_v = \frac{\alpha_v}{\left\| \sum_{v' \in V} \alpha_{v'} v' \right\|} \|u\|. \tag{4.49}$$

Moreover,

$$\widetilde{g}(u) = \|u\| g(\angle u).$$

Likewise, if $\theta = \angle u$, then a solution to Problem 2 can be used to obtain a solution to Problem 1 using the change-of-variables

$$\alpha_v = \frac{a_v}{\sum_{v' \in V} a_{v'}}. \tag{4.50}$$

In general, it is typically easier to prove results for Problem 2.

Now, solving Problems 1 and 2 can be difficult, since the summations in the objective functions are over infinite sets. We now show that it is only necessary to search over a smaller set of lattice points V^* instead of the infinite set V. In a later fact, we will demonstrate that V^* is finite for energy-efficient functions.

Definition 46 (Energy efficient hop types)**.** *Let the set of* energy efficient hop types *be*

$$V^* \triangleq \{v \in V : \boldsymbol{\alpha} = \mathbf{e}_v \text{ solves Problem 1 for } \theta = \angle v\}. \tag{4.51}$$

where $\mathbf{e}$ *is an indicator vector where the* v*'th element is 1 and the rest are zero. Furthermore, the equivalency of Problems 1 and 2 implies* V^* *is can also be written as*

$$V^* = \{v \in V : \mathbf{a} = \mathbf{e}_v \text{ solves Problem 2 for } u = v\}. \tag{4.52}$$

We now demonstrate that we need only search over elements of V^* to solve Problems 1 and 2.

Fact 47 (Solutions to Problems 1 and 2 use only elements of V^*)**.** *If* $\mathbf{a}$ *solves Problem 2 for some* u*, then* $v \in V^*$ *for each nonzero coefficient* $a_v > 0$*. Equivalently, if* $\boldsymbol{\alpha}$ *solves Problem 1, then* $v \in V^*$ *for every nonzero coefficient* $\alpha_v > 0$*.*

Proof. Proof by contradiction. Suppose $\mathbf{a}$ solves Problem 2 for some fixed u and there exists some $v \in V$ such that $a_v > 0$, but $v \notin V^*$. By definition of V^*, the fact that $v \notin V^*$ implies that the indicator vector $\mathbf{e}_v$ does not minimize the cost function in Problem 2 for v. Therefore, there must exist some other vector $\mathbf{b} \neq \mathbf{e}_v$ solving Problem 2 for $u = v$, i.e.,

$\sum_{v' \in V} b_{v'} v' = v$, $b_{v'} \geq 0$. Additionally, since $\mathbf{b}$ minimizes the cost function $\widetilde{J}(\cdot; v)$, we must have the inequality $\widetilde{J}(\mathbf{b}; v) < \widetilde{J}(\mathbf{e}_v; v)$. This last statement implies

$$\sum_{v' \in V} b_{v'} f(\|v'\|) < f(\|v\|).$$

We now show that this generates a contradiction by violating the optimality of $\mathbf{a}$. Substituting this inequality into the definition of $J(\mathbf{a}; v)$ generates the contradiction

$$\begin{aligned}
\widetilde{J}(\mathbf{a}; v) &= \sum_{v' \in V} a_{v'} f(\|v'\|) \\
&= a_v f(\|v\|) + \sum_{v' \neq v} a_{v'} f(\|v'\|) \\
&> a_v \sum_{v' \in V} b_{v'} f(\|v'\|) + \sum_{v' \neq v} a_{v'} f(\|v'\|) \\
&= \sum_{v' \in V} c_{v'} f(\|v'\|) \\
&= \widetilde{J}(\mathbf{c}; v),
\end{aligned}$$

where $\mathbf{c} = \mathbf{a} - a_v \mathbf{e}_v + a_v \mathbf{b}$. We see that the optimality of $\mathbf{a}$ is violated by $\mathbf{c} \neq \mathbf{a}$ since it $\mathbf{c}$ generates a lower cost $\widetilde{J}(\mathbf{c}; v)$.

Finally, since Problems 1 and 2 are equivalent, this result must also hold for any $\boldsymbol{\alpha}$ solving Problem 1 for some fixed θ. □

It is obvious that V^* is a smaller set than V. However, it would be useful to show that V^* is actually a finite set. V^* finite combined with Fact 47 implies that Problems 1 and 2 reduce to straightforward linear programs, and thus can be solved by normal means. In the next section, we determine a distance $\widetilde{d}^*$ such that $\|v\| > \widetilde{d}^*$ implies that v cannot be in V^*. Thus, the set $\widetilde{V} = \{v \in V : \|v\| \leq \widetilde{d}^*\}$ forms a *finite* superset of V^*. Later, we will show that each element of $\widetilde{V}$ can be checked individually to determine if it is an element of V^*.

4.6.2 Sensor separation guaranteeing relay region contains lattice point

In this section, we find bound an upper bound on x such that a lattice V intersects the relay region $\widetilde{\mathcal{R}}(x)$, thereby giving an upper bound on the distance over which direct transmission can be efficient for a given lattice. As described at the end of the previous section, this will also eventually imply that the set V^* is finite when sensor communication is modeled with relay-efficient functions.

Let $V + \mathbf{c}$ be a shifted lattice defined by the basis vectors $[u_1, u_2]$ and shift vector $\mathbf{c}$, where $u_1, u_2, \mathbf{c} \in \mathbb{R}^2$, and u_1 and u_2 are linearly independent. We want to find a distance d^* such that two sensors separated by $x > d^*$ are guaranteed to have an element of $\mathbf{v} \in V + \mathbf{c}$ contained in the relay region $\widetilde{\mathcal{R}}(x)$ for any shift $\mathbf{c}$ of the lattice V. This will mean that for distances greater than d^*, direct transmission will never be optimal since there exists a relay sensor within the relay region $\widetilde{\mathcal{R}}(x)$. We begin by defining d^*.

Definition 48. *Let $f : \mathbb{R}^+ \to \mathbb{R}^+$ be a continuous, convex, nondecreasing function, and suppose f is relay-efficient with efficiency threshold x^*. Let V be a lattice defined by the linearly independent basis vectors $[u_1, u_2]$. Define d^* to be the smallest distance such that the relay region is guaranteed to intersect any shift $\mathbf{c}$ of the lattice V. That is,*

$$d^* \triangleq \inf \left\{ x \geq 0 : \widetilde{\mathcal{R}}(x) \cap (V + \mathbf{c}) \neq \emptyset, \quad \forall \mathbf{c} \in \mathbb{R}^2 \right\}. \tag{4.53}$$

We briefly note the following two facts about d^*:

1. $d^* \geq x^*$, since x must be large enough for the relay region $\widetilde{\mathcal{R}}(x)$ to be nonempty.
2. If $x^* = \infty$, then $d^* = \infty$.

In general, finding an exact value for d^* is difficult. However, we can use our diamond-shaped region $\mathcal{D}(x; \delta_1(x), \delta_2(x))$ to find an upper bound. Our strategy will be to inscribe a circle within $\mathcal{D}(x; \delta_1(x), \delta_2(x))$ and compare it to the basis vectors that generate V. Then we will show that this circle must intersect V for a sufficiently large radius that grows with x. If V intersects this circle, then it must also intersect $\widetilde{\mathcal{R}}(x)$.

Fact 49. *Let $u_1, u_2 \in \mathbb{R}^2$ be two linearly independent vectors that form the basis of a lattice. A closed circle centered at any $c_0 \in \mathbb{R}^2$ whose radius r satisfies $r > \frac{1}{2}\max\{\|u_1 - u_2\|, \|u_1 + u_2\|\}$ must contain an element of V in its interior.*

Proof. Let $\{0, u_1, u_2, u_1 + u_2\}$ define the vertices of a parallelogram. The length of the longest diagonal of the parallelogram has length $\frac{1}{2}\max\{\|u_1 - u_2\|, \|u_1 + u_2\|\}$. By drawing a circumcircle around this parallelogram with radius $\frac{1}{2}\max\{\|u_1 - u_2\|, \|u_1 + u_2\|\}$, it is easy to see that any closed circle in $\mathbb{R}^2$ with radius satisfying $r > \frac{1}{2}\max\{\|u_1 - u_2\|, \|u_1 + u_2\|\}$ must contain at least one lattice point. □

We now inscribe a circle within $\mathcal{D}(x; \delta_1(x), \delta_2(x))$; its radius is

$$\eta(x) = \frac{\delta_1(x)\delta_2(x)}{\sqrt{\delta_1^2(x) + \delta_2^2(x)}},$$

which is equivalent to

$$\eta(x) = \sqrt{\delta_1^2(x)||\delta_2^2(x)},$$

where $a||b$ is defined to be

$$a||b = \frac{ab}{a+b}.$$

If $\eta(x)$ satisfies $\eta(x) > \frac{1}{2}\max\{\|u_1 - u_2\|, \|u_1 + u_2\|\}$, then the relay region will not be empty. From the previous sections, we have closed-form expressions for $\delta_1(x)$ and $\delta_2(x)$ and they are both increasing in x. Thus, $\eta(x)$ is also increasing in x. It is easy to see this since $\eta(x)$ can also be rewritten as

$$\eta(x) = \frac{1}{\sqrt{\frac{1}{\delta_1^2(x)} + \frac{1}{\delta_2^2(x)}}}$$

and any increase in $\delta_1(x)$ or $\delta_2(x)$ will cause $\eta(x)$ to increase. Thus, a computer can be used to find a $\widetilde{d}^*$ solving

$$\eta(\widetilde{d}^*) = \frac{1}{2}\max\{\|u_1 - u_2\|, \|u_1 + u_2\|\}.$$

Note that $\widetilde{d}^*$ has the property that any $x > \widetilde{d}^*$ implies $\eta(x) > \max\{\|u_1 - u_2\|, \|u_1 + u_2\|\}$. Thus, $\widetilde{d}^*$ is an upper bound

$$d^* \leq \widetilde{d}^*.$$

We also note that $a||b \geq \min\{a, b\}$, and we can find another upper bound $\widehat{d}^*$ by finding the unique $\widehat{d}^*$ such that

$$\min\{\delta_1(\widehat{d}^*), \delta_2(\widehat{d}^*)\} = \frac{1}{2}\max\{\|u_1 - u_2\|, \|u_1 + u_2\|\}.$$

The motivation is that if an analytic expression is desired, it may be possible to solve this by hand instead of solving $\eta(\widetilde{d}^*) = \frac{1}{2}\max\{\|u_1 - u_2\|, \|u_1 + u_2\|\}$. Again, note that d^* is upper bounded

$$d^* \leq \widehat{d}^*.$$

We now use these results to show that V^* is finite for relay-efficient functions.

Fact 50. *If the energy cost function f is relay-efficient, then the set of hop types V^* is finite.*

Proof. The following proof can use either upper bound $\widetilde{d}^*$ or $\widehat{d}^*$. Without loss of generality, we use $\widetilde{d}^*$. Following the process above, if $f(x)$ is relay efficient, then a computer can calculate a distance $\widetilde{d}^*$ such that

$$\eta(\widetilde{d}^*) = \frac{1}{2}\max\{\|u_1 - u_2\|, \|u_1 + u_2\|\}.$$

Thus, $\mathcal{D}(\widetilde{d}^*; \delta_1(\widetilde{d}^*), \delta_2(\widetilde{d}^*))$ is nonempty since it contains an inscribed circle of radius $\eta(\widetilde{d}^*)$. By Fact 49 and the fact that $\eta(x)$ is an increasing function of x for $x > \widetilde{d}^*$, the relay region for any two sensors separated by distance x is nonempty and must intersect the lattice V. Thus, take any lattice point $v \in V$ satisfying $\|v\| > \widetilde{d}^*$. $\|v\| > \widetilde{d}^*$ implies that the relay region $\mathcal{R}(0, v)$ intersects V, i.e., there exists some $v' \in V$ such that $v' \in \mathcal{R}(0, v)$. This means that it is cheaper to relay through v' than to transmit directly to v, i.e., $f(\|v'\|)+f(\|v-v'\|) < f(\|v\|)$. This means that $\mathbf{e}_v$ cannot solve Problem 2 for $u = v$, and therefore $v \notin V^*$ by definition of V^*. □

A direct result of Fact 50 is that we can find a finite superset $\widetilde{V}$ of V^*.

Corollary 51. *The set of lattice points whose distance is less than $\widetilde{d}^*$ from the origin*

$$\widetilde{V} = \left\{ v \in V : \|v\| \leq \widetilde{d}^* \right\} \tag{4.54}$$

is a finite superset of V^*, i.e.,

$$V^* \subset \widetilde{V}. \tag{4.55}$$

Proof. Follows directly from proof of Fact 50, since a necessary condition for $v \in V^*$ is that $\|v\| \leq \widetilde{d}^*$. □

Using the fact that V^* is finite, we can find bounds that relate the solutions to Problems 1 and 2 to finding the shortest paths between lattice points in our network; that is, this Fact shows that $g(\theta)$ is indeed the function we seek to describe the minimum cost of long-distance communication in a lattice network.

The following is a main result of Chapter 4. It shows that the output of our linear program is a lower bound to any path of minimal communication energy in a sensor network, and the output of our linear program plus a constant is an upper bound to any path of minimal communication energy in a sensor network. Note that the constant error term is negligible for sufficiently large distances $\|u\|$; therefore, our linear program provides (nearly optimal) energy-efficient paths for long-distance communication.

Fact 52. *Let f be a relay-efficient function so that V^* is finite by Fact 50. Let $u \in V$. Let* **a** *solve Problem 2, and let $\{n_v\}$ minimize $e^*(0, u)$. Then*

$$\widetilde{g}(u) \leq e^*(0, u) \leq \widetilde{g}(u) + f\left(|V^*| \max_{v \in V^*} \|v\| \right).$$

Proof. We begin by showing the lower bound to $e^*(0,u)$. Since $0, u \in V$, clearly any solution to $e^*(0,u)$ must have the "error hop" terms in the definition of $e(0,u,p)$ equal to zero. Thus, solving for the cheapest path $e^*(0,u)$ is identical to Problem 2, except that the solution set is over integers instead of reals. Since the solution of Problem 2 is over a larger set, we must have that

$$\min_{\mathbf{a}} \sum_{v \in V} a_v f(\|v\|) \leq \min_{\mathbf{n}} \sum_{v \in V} n_v f(\|v\|)$$

which implies

$$\widetilde{g}(u) \leq e^*(0,u).$$

For the upper bound, we can use a solution to Problem 2 to generate an approximate solution to $e^*(0,u)$ by selecting

$$n'_v = \lfloor a_v \rfloor$$

for every $v \in V$. Since the $\{n'_v\}$ are integers, they generate a path through the lattice from the origin to some u' where

$$u' = \sum_{v \in V} \lfloor a_v \rfloor v,$$

and this path has cost bounded by

$$\begin{aligned} e(0, u', \{n'_v\}) &= \sum_{v \in V} \lfloor a_v \rfloor f(\|v\|) \\ &\leq \sum_{v \in V} a_v f(\|v\|) \\ &= \widetilde{g}(u). \end{aligned}$$

Define the residual vector between u and u' to be $r = u - u'$. Note that $r \in V$ since both $u \in V$ and $u' \in V$. Since $\sum_{v \in V} a_v v = u$, we must have that the length of r is bounded by

$$\begin{aligned} \|r\| &= \|u - u'\| \\ &= \left\| \sum_{v \in V} (a_v - \lfloor a_v \rfloor) v \right\| \\ &\leq \sum_{v \in V} |\lfloor a_v \rfloor - a_v| \|v\| \\ &\leq \sum_{v \in V^*} \|v\| \\ &\leq |V^*| \max_{v \in V^*} \|v\|. \end{aligned}$$

Thus a loose bound to the cost of relaying from u' to u is given by bounding the cost of direct transmission $f(\|r\|)$ using the above result to obtain

$$e^*(u', u) = e^*(0, r) \leq f(\|r\|) \leq f\left(|V^*| \max_{v \in V} \|v\|\right).$$

Finally, since $e^*(0, u) \leq e^*(0, u') + e^*(u', u)$, we must have that

$$\begin{aligned} e^*(0, u) &\leq e^*(0, u') + e^*(u', u) \\ &= e^*(0, u') + e^*(0, r) \\ &\leq e(0, u'; \{n'_v\}) + f\left(|V^*| \max_{v \in V} \|v\|\right) \\ &= \widetilde{g}(u) + f\left(|V^*| \max_{v \in V} \|v\|\right). \end{aligned}$$

□

The following corollary follows from the fact that $\widetilde{g}(u)$ can be rewritten as $\|u\| g(\angle u)$. This also shows that the quantity $\frac{e^*(0,u)}{\|u\|}$ converges to $g(\theta)$ as $\|u\| \to \infty$.

Corollary 53. *Let f be a relay-efficient function so that V^* is finite by Fact 50. Let $u \in V$. Then for $\theta = \angle u$,*

$$g(\theta) \leq \frac{e^*(0, u)}{\|u\|} \leq g(\theta) + \frac{f\left(|V^*| \max_{v \in V} \|v\|\right)}{\|u\|}.$$

Proof. Let $\mathbf{a}$ solve Problem 2 for $\theta = \angle u$, and let $\{n_v\}$ minimize $e^*(0, u)$. Following our results of Fact 52 and letting $A = \sum_{v \in V} a_v$, and $\boldsymbol{\alpha} = \frac{a_v}{A}$, we must have that

$$\begin{aligned} \frac{\widetilde{g}(u)}{\|u\|} &= \min_{\mathbf{a}} \frac{\sum_{v \in V} a_v f(\|v\|)}{\|u\|} \\ &= \min_{\mathbf{a}} \frac{\sum_{v \in V} a_v f(\|v\|)}{\left\|\sum_{v' \in V} a_{v'} v'\right\|} \\ &= \min_{\mathbf{a}} \frac{\sum_{v \in V} \frac{a_v}{A} f(\|v\|)}{\left\|\sum_{v' \in V} \frac{a_{v'}}{A} v'\right\|} \\ &= \min_{\boldsymbol{\alpha}} \frac{\sum_{v \in V} \alpha_v f(\|v\|)}{\left\|\sum_{v' \in V} \alpha_{v'} v'\right\|} \\ &= g(\theta), \end{aligned}$$

and the result follows by normalizing the inequality in Fact 52 by $\|u\|$. □

4.6.3 Calculating V^*

For a given energy function f and lattice sensor location V, we would like to find a systematic way to generate the finite set V^*. The following algorithm does this:

Method for finding V^*:

1. Given: f, V.
2. Following the method in Section 4.6.2, calculate a distance $\widetilde{d}^*$ such that for all $v \in V$ satisfying $\|v\| > \widetilde{d}^*$, $V \cap \mathcal{R}(0, v) \neq \emptyset$.
3. Generate the finite set $\widetilde{V} = \{v \in V : \|v\| \leq \widetilde{d}^*\}$.
4. Initialize $V^* = \emptyset$.
5. For each $v \in \widetilde{V} \backslash \{(0,0)\}$:
   (a) Solve Problem 2 for $u = v$ by optimizing over $\widetilde{V}$ using standard linear programming tools to obtain coefficient vector $\boldsymbol{\alpha}$.
   (b) Calculate minimum cost $\mathcal{E}^* = \widetilde{J}(\boldsymbol{\alpha}; v)$.
   (c) Calculate cost of using only v, $\mathcal{E}_v = \widetilde{J}(\mathbf{e}_v; v)$.
   (d) If $\mathcal{E}_v = \mathcal{E}^*$, then set $V^* = V^* \cup v$.
6. Return resulting V^*.

Step 5(a) is justified by combining Fact 47 and Corollary 51.

4.6.4 Conjecture: A closed-form expression for $g(\theta)$ and $\widetilde{g}(u)$

In this section, we conjecture that $\widetilde{g}(u)$ can be constructed in closed form. Specifically, we conjecture that the minimum of Problem 2 can be be attained using an $\mathbf{a}$ containing at most two nonzero elements. In the case that $\theta = v$ for some $v \in V^*$, then $\mathbf{a}$ will contain only one nonzero element, namely, its vth element will be nonzero. For any other θ, there will be two nonzero elements corresponding to the elements of V^* that are "closest" in angle on each side, i.e., the closest bounding elements of V^* in angle.

To begin, for any $u \neq (0,0)$, define

$$R_u = \begin{bmatrix} \cos \angle u & \sin \angle u \\ -\sin \angle u & \cos \angle u \end{bmatrix}$$

to be a rotation matrix that rotates a vector $\angle u$ clockwise about the origin. We can define the set of vectors that lie in the half-plane "above" u to be

$$\overline{V}^*(u) = \{v \in V^* : e_2 R_u v > 0\},$$

and the set of vectors that lie in the half-plan "below" u to be

$$\underline{V}^*(u) = \{v \in V^* : e_2 R_u v < 0\}.$$

Assume without loss of generality that all $v \in V^*$ have unique angles. Given some arbitrary u, two vectors in V^* that most closely "bound" u are defined to be

$$\overline{v}^*(u) = \operatorname*{argmax}_{v \in \overline{V}^*(u)} v^T u$$

and

$$\underline{v}^*(u) = \operatorname*{argmax}_{v \in \underline{V}^*(u)} v^T u.$$

Our conjecture is as follows: For any u, if $\angle u = \angle v$ for some $v \in V^*$, then the only nonzero coefficient will be a_v and it will be equal to $a_v = \frac{\|u\|}{\|v\|}$. Otherwise, we choose the two "closest bounding" vectors $\overline{v}^*(u)$ and $\underline{v}^*(u)$, and set $a_{\overline{v}^*(u)}$ and $a_{\underline{v}^*(u)}$ to be the unique coefficients solving the linear equation

$$\begin{bmatrix} \overline{v}^*(u) & \underline{v}^*(u) \end{bmatrix} \begin{bmatrix} a_{\overline{v}^*(u)} \\ a_{\underline{v}^*(u)} \end{bmatrix} = u.$$

Thus, our prediction $\widehat{g}(u)$ of $\widetilde{g}(u)$ can be expressed as

$$\widehat{g}(u) = \begin{cases} \frac{\|u\|}{\|v\|} f(\|v\|), & \text{if } \angle u = \angle v \text{ for some } v \in V^*, \\ \begin{bmatrix} f(\|\overline{v}^*(u)\|) \\ f(\|\underline{v}^*(u)\|) \end{bmatrix}^T \begin{bmatrix} \overline{v}^*(u) & \underline{v}^*(u) \end{bmatrix}^{-1} u & \text{otherwise.} \end{cases}$$

In the case that all $v \in V^*$ do not have unique angles, then we can arbitrarily choose vectors that satisfy the above definitions and the resulting value of $\widehat{g}(u)$ will be the same. For example, if $\angle v_i = \angle v_j$, choose the vector with smallest length.

In a similar manner, we can calculate a normalized function $\widehat{g}(\theta)$ by either (a) choosing $u = [\cos\theta, \sin\theta]$ in the above expressions, or (b) choosing $\widehat{g}(\theta) = \frac{\widehat{g}(u)}{\|u\|}$ for any u satisfying $\angle u = \theta$.

In the next section, we run simulations in which we compare these closed form predictions to $e^*(0,u)$ and $\widetilde{g}(u)$.

4.6.5 Lattice Communication Experiments

For large u and for communication on a lattice network using a power law model (4.41), the goal of this section is determine how close $e^*(0,u)$ is to $\widetilde{g}(u)$, and also to test our conjecture that $\widehat{g}(u)$ is a good predictor of $e^*(0,u)$. Our experimental setup, which is shown in Figure 4.11, consists of arranging 2500 sensors spaced 1 meter apart in a square region, with 50 sensors arranged along each side. The sensors denoted by blue circles lie in an annulus of inner radius 23 meters and outer radius 24 meters, and our goal will be to calculate the energy required for each of these sensors to communicate with the center sensor (green triangle). Specifically, for each sensor u in this annulus, Dijkstra's algorithm will be used to calculate $e^*(0,u)$, which is the total path cost from the center to each of the blue circles. This value will then be compared to $\widetilde{g}(u)$, which can be calculated by first finding V^* using the method in Section 4.6.3, and then solving Problem 2 for u using standard linear programming methods (we used the Python method `scipy.optimize.linprog`). Finally, we will calculate $\widehat{g}(u)$ using the method in Section 4.6.4. Given these three quantities, we will also compare the distance-normalized quantities $\frac{e^*(0,u)}{\|u\|}$, $g(\angle u)$ and $\widehat{g}(\angle u)$. We did not find it necessary to plot the upper bound of Fact 52 because (a) it was one or two orders of magnitude larger than the other plotted quantities in these experiments, and (b) our experimental results were already either exact or tight. The tightness of our results suggests that this upper bound may be very loose; tightening this bound could be the subject of future work.

4.6.5.1 Power law with no constant overhead ($c = 0$)

In our first experiment, whose results are shown in Figure 4.12, we consider the case where

$$f(x) = x^3$$

which is an "ideal" case where there is no constant over head for transmission, since $f(0) = 0$. This model has an efficiency threshold of $x^* = 0$ m, λ^* does not exist, $\widetilde{d^*} = 1.8218$ m, and V^* consists of four vectors $V^* = \{(\pm 1, 0), (0, \pm 1)\}$. In Figure 4.12(a), we see the total energy calculated by all three methods, and we see that they all generate the same values for each experiment. Thus, we see that solving Problem 2 yields "good" paths. Figure 4.12(a) also supports our conjecture that $\widetilde{g}(u) = \widehat{g}(u)$, i.e. that the "closest bounding" vectors $\overline{v}^*(u)$ and $\underline{v}^*(u)$ are the only two vectors used in the solution to Problem 2. Figure 4.12(b) shows the

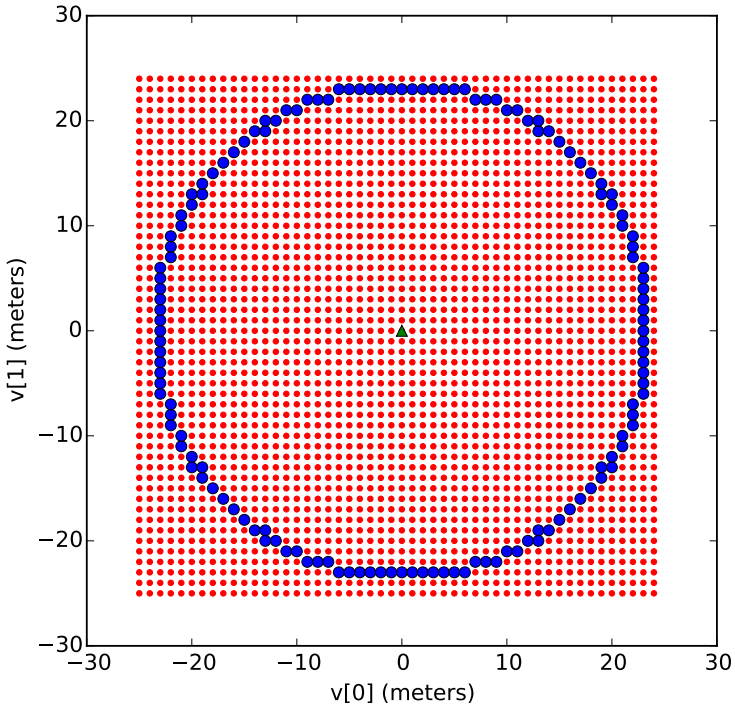


Figure 4.11: Sensor layout for lattice communication experiments. $e^*(0,u)$, $\widetilde{g}(u)$ and $\widehat{g}(u)$ were calculated for paths connecting the green triangle sensor (center) to the blue circles.

output of the coefficients when solving the linear program for Problem 2, and again we see that there are at most two nonzero coefficients, which correspond to the "closest bounding" vectors. Figures 4.12(c) and 4.12(d) show the distance-normalized quantities of (a) and (b).

Finally, we note that one of the reasons for the perfect alignment of these three quantities is that V^* happens to equal the basis vectors $V^* = \{\pm u_1, \pm u_2\}$. As we will see, when this is not the case, there exist elements of V that cannot be reached using the span of V^*, and thus other hops that are not in V^* will be necessary. Also, for any given u, there are only two shortest v's that have positive inner product with u. These shortest v's suffice because the $x^* = 0$.

4.6.5.2 Power law with "small" constant overhead ($c = 5$)

In our second experiment, whose results shown in Figure 4.13, we consider the case where

$$f(x) = x^3 + 5$$

which is a case where there is is a small constant over head for transmission, since $c = 5 = f(0)$. We say that this is a "small" overhead because V^* consists of the 8 vectors

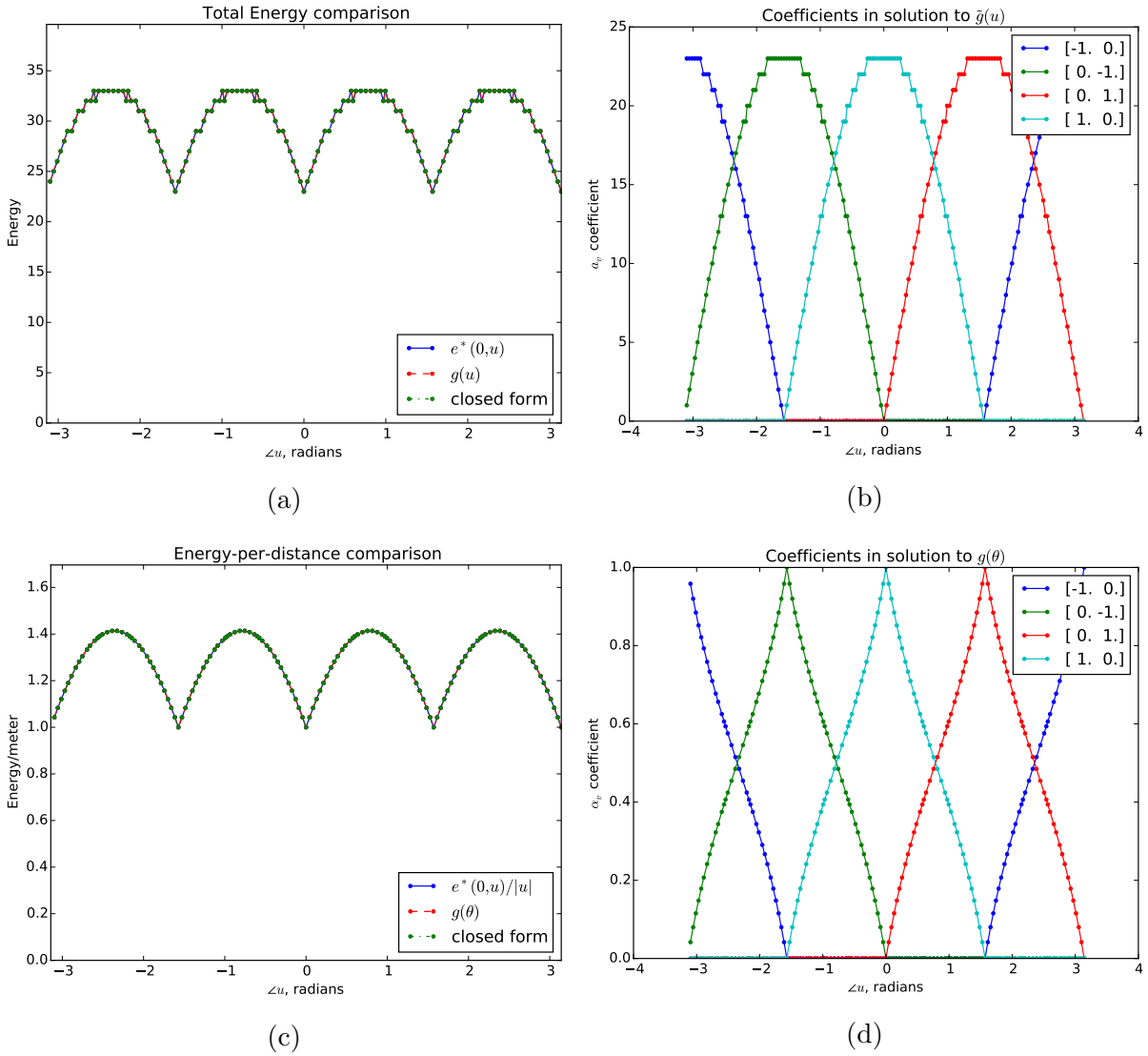


Figure 4.12: Comparison of shortest path energies to predicted energies using elements of V^* for $f(x) = x^3$ (no constant overhead). Normalized coefficients to output of $g(u)$ (i.e. $g(\theta)$) are also shown.

$V^* = \{(\pm 1, 0), (0, \pm 1), (\pm 1, \pm 1)\}$, which consist of the "8 neighbors" of the origin. This is contrasted to a "large" over head that causes sensors to be "skipped" over, as we will see in our final experiment. This model has an efficiency threshold of $x^* = 1.8821$ m, $\lambda^* = 1.3572$ m, and $\widetilde{d}^* = 8.9690$ m. Note that the lengths of each $v \in V^*$ are close in value to λ^*; indeed, it is easy to see that all other lattice points are not as close in value to λ^* as those in V^*.

In Figure 4.13(a), we once again see that all three methods generate the same values for each experiment. Again, we hypothesize that this is the case because the span of V^* is equal to the span of the lattice basis vectors $\{u_1, u_2\}$. Figure 4.13(b) shows the output of the

coefficients when solving the linear program for Problem 2, and again we see that there are at most two nonzero coefficients, corresponding to the "closest bounding" vectors. Figures 4.13(c) and 4.13(d) show the distance-normalized quantities of (a) and (b).

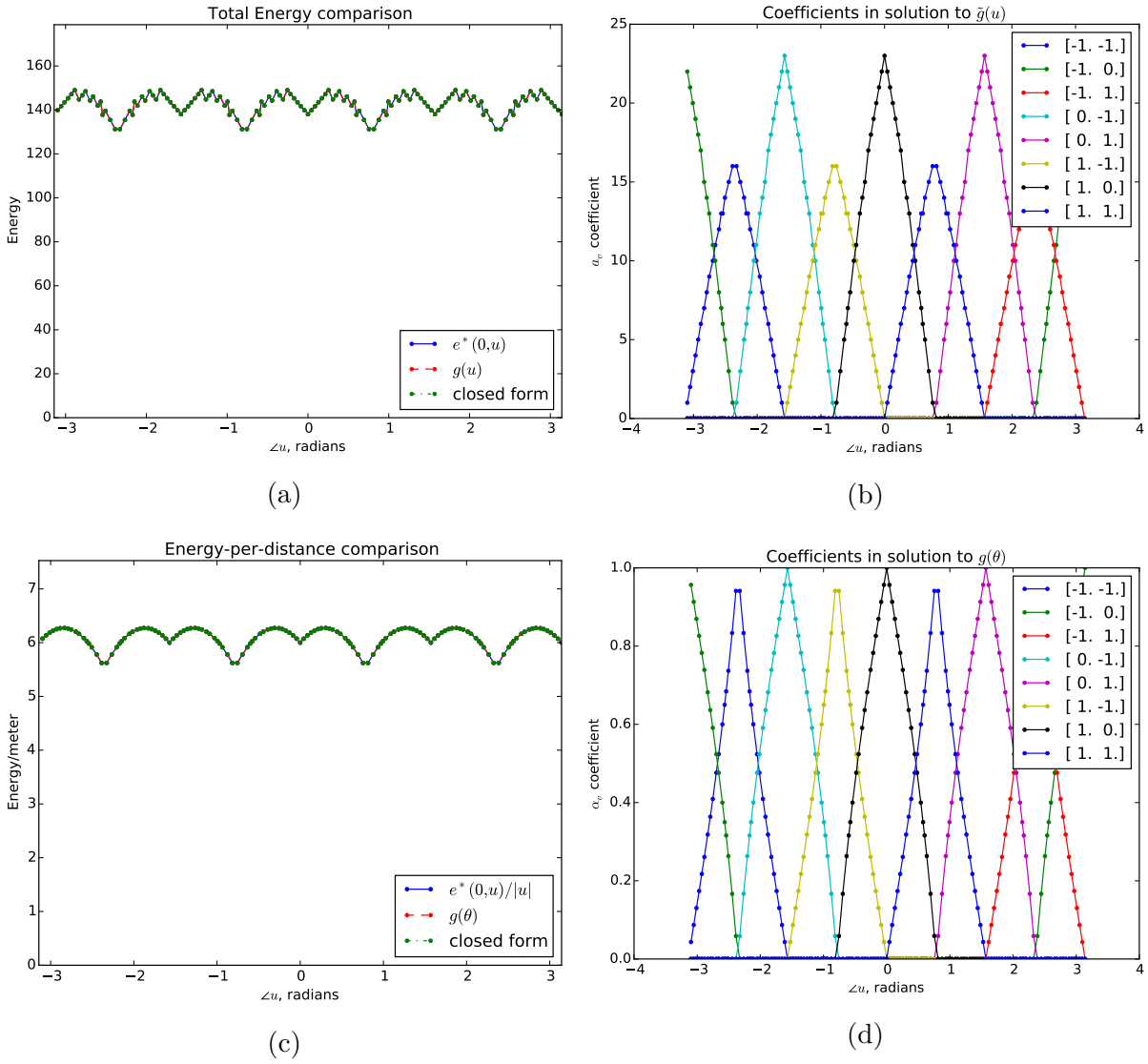


Figure 4.13: Comparison of shortest path energies to predicted energies using elements of V^* for $f(x) = x^3 + 5$ (small constant overhead). Normalized coefficients to output of $g(u)$ (i.e. $g(\theta)$) are also shown.

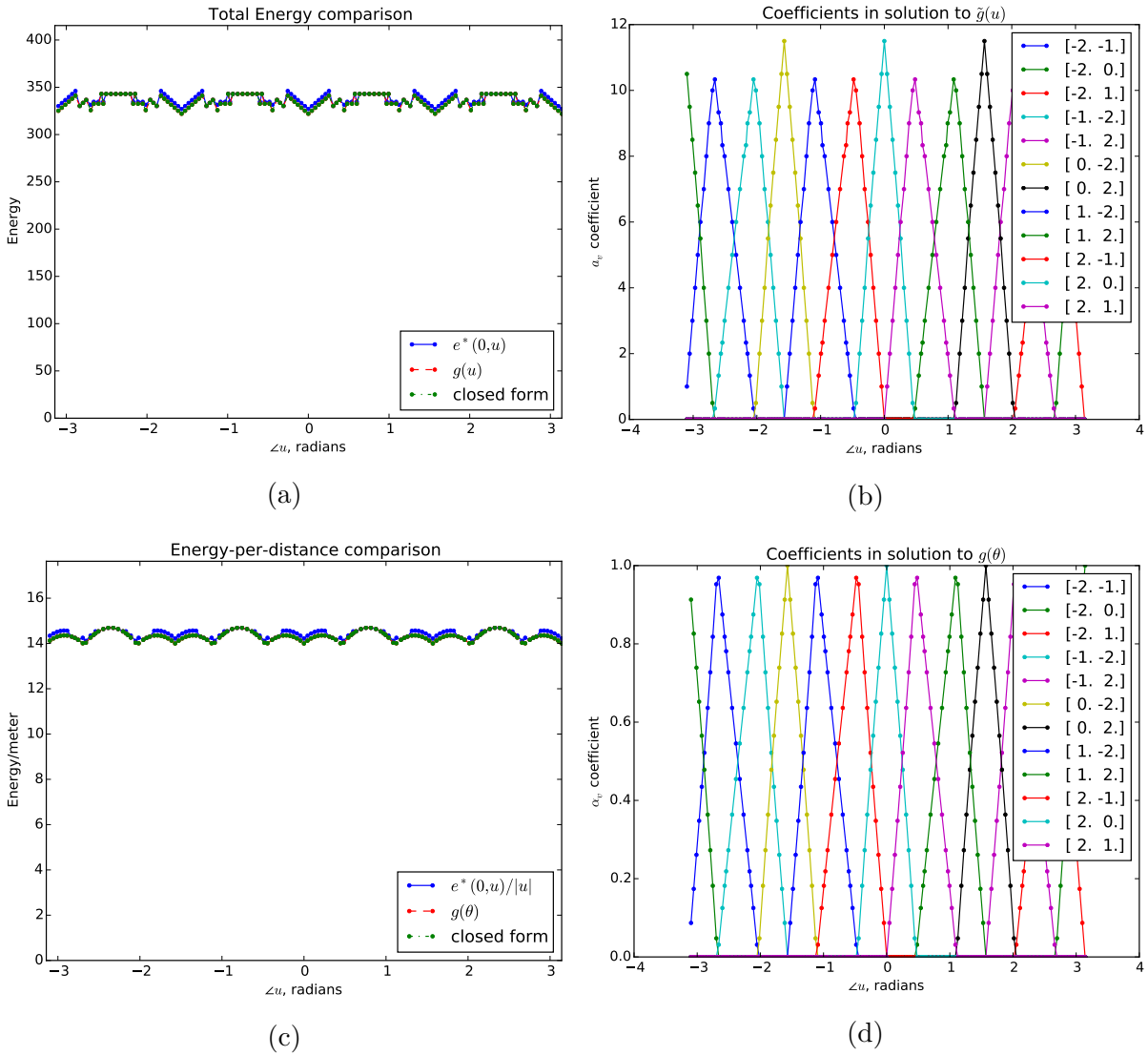


Figure 4.14: Comparison of shortest path energies to predicted energies using elements of V^* for $f(x) = x^3 + 20$ (large constant overhead). Normalized coefficients to output of $g(u)$ (i.e. $g(\theta)$) are also shown.

4.6.5.3 Power law with "large" constant overhead ($c = 20$)

In our third and final experiment, with results shown in Figure 4.14, we consider the case where

$$f(x) = x^3 + 20$$

which is a case where there is is a large constant overhead ($c = 20$) for transmission. We say that this is a "large" overhead because V^* consists of the 12 vectors

$$V^* = \{(\pm 2, 0), (0, \pm 2), (\pm 1, \pm 2), (\pm 2, \pm 1)\}.$$

Unlike the first two experiments, these are not lattice points that "neighbor" the origin. This model has an efficiency threshold of $x^* = 2.9876$ m, $\lambda^* = 2.1544$ m, and $\widetilde{d}^* = 13.3733$ m. Once again, the lengths of each $v \in V^*$ are close in value to λ^*.

In Figure 4.14(a), we now see that all three methods do not generate the same values for each experiment. In particular, $e^*(0, u)$ is lower bounded by both $\widetilde{g}(u)$ and $\widehat{g}(u)$. However, $\widetilde{g}(u)$ and $\widehat{g}(u)$ still match, which again supports our conjecture that the closed form approach of Section 4.6.4 is equivalent to solving the linear program in Problem 2. Also observe that the span of V^* is not equal to the span of the lattice basis vectors $\{u_1, u_2\}$, which results in the shortest paths using "error" hops that are not elements of V^*.

Figure 4.14(b) shows the output of the coefficients when solving the linear program for Problem 2, and again we see that there are at most two nonzero coefficients, corresponding to the "closest bounding" vectors. Figures 4.14(c) and 4.13(d) show the distance-normalized quantities of (a) and (b).

4.7 Conclusions

In this chapter, we examined energy-performance tradeoffs in wireless sensor networks; specifically, we considered "cutset networks" where sensors were deployed along straight line segments. We considered the problem of source localization on these networks in both centralized and decentralized scenarios. In the centralized scenario, we considered Manhattan, Triangular, and Honeycomb cutset networks, and found that they offered reduced energy costs at the expense of lower estimation accuracy. In the decentralized scenario, we provided the "Midpoint Algorithm" for efficient communication on a Manhattan sensor network, which offered cheaper communication over competing localization methods at the same accuracy. We also took a broader look at functions that model sensor communication, and characterized how these functions determine whether or not relaying through a sensor network can be used to reduce the overall energy cost. These observations were tested in the context of a lattice sensor network, where it was found that for relay-efficient functions, there only exist one or two hop types that are required for efficient communication through the network.

CHAPTER 5

Conclusions

We now summarize our results by listing the main contributions of each chapter.

Chapter 2 introduced Manhattan sampling in two and higher dimensions, and proved sampling theorems. In particular, Section 2.3 presented two-dimensional Manhattan sampling, which consists of sampling an image on equally spaced rows and columns, formed by the union of two lattices. It was shown that images bandlimited to the union of Nyquist regions corresponding to these lattices can be perfectly reconstructed from their Manhattan samples. The higher dimensional analogues of these result were given in Section 2.4.

Chapter 3 presents three new methods for reconstructing images from their Manhattan samples: The Piecewise-Planar method, the Orthogonal Gradient algorithm, and the Locally Orthogonal Orientation Penalization algorithm. Additionally, despite being designed with cutset sampling in mind, the OG and LOOP algorithms can also be applied to reconstructing images from arbitrary sampling patterns, such as traditional lattice sampling. Of the three methods, the LOOP algorithm performs the best, both in terms of mean-squared error and subjective image quality. It was also shown that for lattice interpolation, the LOOP algorithm outperforms bicubic interpolation, and also performs competitively against a recent interpolation method [11].

Finally, Chapter 4 investigates energy-performance tradeoffs in cutset wireless sensor networks. These were introduced in Section 4.1, where it was shown that under a power-law energy model with no constant overhead, cutset networks require less energy for communication than lattice networks. In Section 4.3, we found that for the problem of centralized source localization, cutset networks offered significant increases in energy efficiency over random networks and lattice networks without surrendering much accuracy. Furthermore, in Section 4.4, we explored the decentralized source localization problem and presented the Midpoint Algorithm for source localization on a Manhattan grid. We found that the Midpoint Algorithm used less energy than a competing POCS algorithm at certain fixed accuracies. Finally, in Sections 4.5, we investigated functions that are used to model the energy required

for data transmission in a sensor network. In particular, we defined a *relay efficient function* to be a function who generates a nonempty *relay region* for sufficiently large distances, i.e., a region where a relay sensor can be placed to reduce the overall transmission energy between two sensors. After finding sufficient conditions for relay-efficiency, we applied our findings to communication on a lattice network in Section 4.6. We then found that if a function is relay-efficient, there are typically only one or two "hop types" needed to construct a path of minimal energy consumption when transmitting messages over long distances in a lattice sensor network. We also found that these hop types can be found by solving a linear programming problem, and we conjectured that this linear program can also be solved in closed form once the most efficient "hop types" were known.

5.1 Future Work

It would be nice to reformulate the OG and LOOP algorithms in such a way so that convergence is theoretically guaranteed, hopefully without sacrificing much performance. Furthermore, it is possible that there exists a complete probability model whose MAP solution can be found with a (modified version) of these two algorithms.

With regard to Chapter 4, the idea of a relay region discussed in Section 4.5 can be used in the context of geometric graph theory. For example, a *Gabriel graph* [68] is a geometric graph where the nodes are locations in the plane, and there is an edge between two nodes if and only if the circle with diameter linking the two nodes does not contain another node. This "circle" is exactly the relay region generated by the quadratic power law $f(x) = Px^2$. We believe it can be shown that the Gabriel Graph generated by a set of sensor locations is exactly the graph containing all shortest communication paths in the case of $f(x) = Px^2$. A similar connection lies with *Nearest Neighbors* graphs, which can be formed by drawing an edge between two nodes if there exists no other sensor in the "lune" connecting the pair of nodes (Figure 4.9).

Additionally, Section 4.6 provides a framework that can be extended to periodic sensor networks, such as cutset networks. In particular, any periodic sensor network can be defined using a set of basis vectors and a finite set of shift vectors. By considering each periods of the sensor network to be a "cell," it may be possible to formulate the problem of efficient long-distance communication in the network by only considering efficient paths between cells These paths will then be analogous to the set of efficient lattice hop types V^* explored in Section 4.6.

APPENDIX A

Interior labeling algorithm for $k = 3$

Some definitions: A *run* is a connected border segment of the same class (label)i, i.e. it contains no odd bonds. The *length* of the run is the number of (consecutive) nodes in a run. The *majority class* is the class with the most presence on the border.

1. If 0 odd bonds, fill interior with present class (same as Reyes segmentation)
2. If 2 odd bonds, connect locations of odd bonds with line and fill accordingly (same as Reyes segmentation)
3. If 3 odd bonds, there must be three classes present, i.e., three different runs:
   (a) If each run contains a corner, one class must connect the two odd bonds containing the longest run, fill the region with the color of the longest run, and drop a perpendicular line from the remaining odd bond to this line to complete the partition.
   (b) If any class has all four corners, fill block with that class and ignore the other two classes.
   (c) Otherwise, one class contains no corner; We arbitrarily merge this "no corner" class with one of the other two runs and treat as a 2 odd bond case. We chose to merge with closest run in counter-clockwise direction around the block.
4. If 4 odd bonds:
   (a) If all three classes are present:
      i. If any class contains zero corners :
         A. If "no corner" class has two runs, fill the interior with that class.
         B. Otherwise, ignore "no corner" class, connect endpoints of the runs of the other two classes, and fill remaining interior with majority class.

ii. Otherwise one class has two corners, and two classes have one corner:

A. If two corner class has two runs, connect endpoints of one-corner classes and fill accordingly.

B. Otherwise, some other class has two runs; then connect endpoints of one-run classes.

(b) If only two classes are present, calculate the STRENGTH of each odd bond by comparing the difference in pixel intensities to a predetermined THRESHOLD. A STRONG bond has a difference above the threshold, and is WEAK otherwise.

i. If a pair of adjacent odd bonds are either both strong or both weak connect STRONG to STRONG and WEAK to WEAK.

ii. If both classes have two corners, connect endpoints of runs of the class with the most color on the border (the majority class)

iii. Otherwise, connect endpoints of class with most corners

5. If greater than 4 odd bonds, we find the smallest runs and merge them if bounded by same class on either side. This process continues until the number of odd bonds reduces to 4 or below. If this does not work, we continually "mode filter" the border with a growing window size until there are 4 odd bonds or fewer.

APPENDIX B

Expected value of C

We follow the notation of Section 3.4.1, letting $u = [u_x, u_y]^T$, and let $A_i = [g_j]_{j \in \mathcal{N}_i}$ be our data matrix of noisy gradient samples. Since each g_j is distributed as $\mathcal{N}(\gamma u, \sigma^2 I_{2\times 2})$, we can consider the univariate random variables $g_{x,j} = \gamma u_x + \eta_{j,x}$ and $g_{y,j} = \gamma u_y + \eta_{j,y}$ where $\eta_{j,x}, \eta_{j,y} \sim \mathcal{N}(0, \sigma^2)$. We have that

$$\frac{1}{n}\mathbb{E}\left[A_i A_i^T\right] = \begin{bmatrix} \frac{1}{n}\sum_{j=1}^n \mathbb{E}\left[g_{x,j}^2\right] & \frac{1}{n}\sum_{j=1}^n \mathbb{E}\left[g_{x,j}g_{y,j}\right] \\ \frac{1}{n}\sum_{j=1}^n \mathbb{E}\left[g_{x,j}g_{y,j}\right] & \frac{1}{n}\sum_{j=1}^n \mathbb{E}\left[g_{y,j}^2\right] \end{bmatrix}$$

The top-right element can be evaluated as

$$\begin{aligned} \frac{1}{n}\sum_{j=1}^n \mathbb{E}\left[g_{x,j}^2\right] &= \frac{1}{n}\sum_{j=1}^n \mathbb{E}\left[(\gamma u_x + \eta_{j,x})^2)\right] \\ &= \frac{1}{n}\sum_{j=1}^n \left[\gamma^2 u_x^2 + 2u_x \mathbb{E}[\eta_{j,x}] = \mathbb{E}[\eta_{j,x}^2]\right] \\ &= \frac{1}{n}\sum_{j=1}^n \left[\gamma^2 u_x^2 + \sigma^2\right] \\ &= \gamma^2 u_x^2 + \sigma^2 \end{aligned}$$

Similarly, it can be shown that

$$\frac{1}{n}\sum_{j=1}^n \mathbb{E}\left[g_{y,j}^2\right] = \gamma^2 u_y^2 + \sigma^2.$$

For the cross-terms, we have

$$\begin{aligned}\frac{1}{n}\sum_{j=1}^{n}\mathbb{E}\left[g_{x,j}g_{y,j}\right] &= \frac{1}{n}\sum_{j=1}^{n}\mathbb{E}\left[(\gamma u_x+\eta_{j,x})(\gamma u_y+\eta_{j,y})\right] \\ &= \frac{1}{n}\sum_{j=1}^{n}\left[\gamma^2 u_x u_y-\gamma u_x\mathbb{E}[\eta_{j,y}]-\gamma u_y\mathbb{E}[\eta_{j,x}]+\mathbb{E}[\eta_{j,x}\eta_{j,y}]\right] \quad = \frac{1}{n}\sum_{j=1}^{n}\gamma^2 u_x u_y \\ &= \gamma^2 u_x u_y.\end{aligned}$$

Combining these results yields

$$\frac{1}{n}\mathbb{E}\left[A_i A_i^T\right] = \begin{bmatrix}\gamma^2u_x^2+\sigma^2 & \gamma^2u_xu_y \\ \gamma^2u_xu_y & \gamma^2u_x^2+\sigma^2\end{bmatrix}.$$

We would like to find an eigenvalue decomposition of this matrix. Recall that $\|u\|=1$, so that $u_x^2+u_y^2=1$. Using this fact, we can calculate the characteristic polynomial and setting it equal to zero to obtain

$$\begin{aligned}\begin{vmatrix}\gamma^2u_x^2+\sigma^2-\lambda & \gamma^2u_xu_y \\ \gamma^2u_xu_y & \gamma^2u_x^2+\sigma^2-\lambda\end{vmatrix} &= (\gamma^2u_x^2+\sigma^2-\lambda)(\gamma^2u_x^2+\sigma^2-\lambda)-\gamma^4u_x^2u_y^2 \\ &= \gamma^4u_x^2+\gamma^2\sigma^2u_x^2-\gamma^2u_x^2\lambda+\gamma^2\sigma^2u_y^2+\sigma^4-\sigma^2\lambda \\ &\quad -\gamma^2u_y^2\lambda-\sigma^2\lambda+\lambda^2-\gamma^4u_x^2u_y^2 \\ &= \lambda^2-(\gamma^2u_x^2+\gamma^2u_y^2+2\sigma^2)\lambda+(\gamma^2\sigma^2u_x^2+\gamma^2\sigma^2u_y^2+\sigma^4 \\ &= \lambda^2-(\gamma^2+2\sigma^2)\lambda+\sigma^2(\gamma^2+\sigma^2) \\ &= (\lambda-\sigma^2-\gamma^2)(\lambda-\sigma^2)=0.\end{aligned}$$

Thus, our matrix has eigenvalues

$$\begin{aligned}\lambda_1 &= \gamma^2+\sigma^2 \\ \lambda_2 &= \sigma^2.\end{aligned}$$

One can check that u is an eigenvector corresponding to the largest eigenvalue $\lambda_1 = \gamma^2 + \sigma^2$ since

$$\begin{aligned}\begin{bmatrix} \gamma^2 u_x^2 + \sigma^2 & \gamma^2 u_x u_y \\ \gamma^2 u_x u_y & \gamma^2 u_x^2 + \sigma^2 \end{bmatrix} \cdot \begin{bmatrix} u_x \\ u_y \end{bmatrix} &= \begin{bmatrix} u_x(\gamma^2 u_x^2 + \sigma^2 + \gamma^2 u_y^2) \\ u_y(\gamma^2 u_x^2 + \sigma^2 + \gamma^2 u_y^2) \end{bmatrix} \\ &= (\gamma^2 + \sigma^2) \begin{bmatrix} u_x \\ u_y \end{bmatrix}.\end{aligned}$$

The second eigenvector $u^\perp$ can be chosen by taking either of the two possible $\frac{\pi}{2}$ rotations of v.

APPENDIX C

Minimum path cost calculations

We assume that energy-efficient paths in a cutset network travel along the cutset boundaries, i.e., edges of the tessellation generating the cutset pattern. We also assume that communication along these paths only occurs between neighboring sensors. Since each hop is the same length λ, the minimum path cost $c(r, \phi)$ can be decomposed into the number of hops $n(r, \phi)$ required to travel distance r at angle ϕ times the hop cost λ^α, where α is the communication energy path-loss exponent. Specifically,

$$c(r, \phi) = \lambda^\alpha n(r, \phi).$$

Let $d(r, \phi)$ be the distance traveled along the cutset boundaries to arrive at a point at distance r and angle ϕ. Then the number of hops is simply the total distance along the path divided by the hop length $n(r, \phi) = \frac{d(r,\phi)}{\lambda}$, and we have that

$$c(r, \phi) = \lambda^{\alpha-1} d(r, \phi).$$

All that remains is to find $d(r, \phi)$ for Manhattan, Triangular, and Honeycomb networks.

C.1 Manhattan network case

For a Manhattan network, to travel a net distance of r requires a total distance of

$$d(r, \phi) = r|\sin\phi| + r|\cos\phi|,$$

and thus we have that

$$c(r, \phi) = \lambda^{\alpha-1} r \left(|\cos\phi| + |\sin\phi|\right).$$

C.2 Triangular network case

For a triangular network, by symmetry, it suffices to calculate $d(r, \phi)$ for $|\phi| \leq \frac{\pi}{6}$. To travel a net distance of r requires a total distance of

$$d(r, \phi) = r \cos \phi + r \frac{|\sin \phi|}{\sqrt{3}}, \quad |\phi| \leq \frac{\pi}{6}.$$

and thus we have that

$$c(r, \phi) = \lambda^{\alpha-1} r \left(\cos \phi + \frac{|\sin \phi|}{\sqrt{3}} \right), \quad |\phi| \leq \frac{\pi}{6}.$$

C.3 Honeycomb network case

For a Honeycomb network, by symmetry, it suffices to calculate $d(r, \phi)$ for $|\phi| \leq \frac{\pi}{6}$. To travel a net distance of r requires a total distance of

$$d(r, \phi) = \frac{4}{3} r \cos \phi, \quad |\phi| \leq \frac{\pi}{6},$$

and thus we have that

$$c(r, \phi) = \frac{4}{3} \lambda^{\alpha-1} r \cos \phi, \quad |\phi| \leq \frac{\pi}{6}.$$

BIBLIOGRAPHY

[1] A. Farmer, A. Josan, M. Prelee, D. Neuhoff, and T. Pappas, "Cutset sampling and reconstruction of images," in *Proc. IEEE ICIP*, Brussels, Sept. 2011, pp. 1909–1912.

[2] D. Blatt and A. Hero, "Energy-based sensor network source localization via projection onto convex sets," *IEEE Trans. Sig. Proc.*, vol. 54, no. 9, pp. 3614–3619, 2006.

[3] D. Obenour, A. Michalak, Y. Zhou, and D. Scavia, "Quantifying the impacts of stratification and nutrient loading on hypoxia in the northern Gulf of Mexico," *Environmental science & technology*, vol. 46, no. 10, pp. 5489–5496, 2012.

[4] M. Reyes, X. Zhao, D. Neuhoff, and T. Pappas, "Lossy compression of bilevel images based on Markov random fields," in *Proc. IEEE ICIP*, vol. 2, San Antonio, Sept. 2007, pp. II–373.

[5] M. Reyes and D. Neuhoff, "Arithmetic encoding of Markov random fields," in *Proc. IEEE Int. Symp. Inform. Thy.*, Seoul, June 2009, pp. 532–536.

[6] ——, "Lossless reduced cutset coding of Markov random fields," in *Proc. Data Compression Conf.* Snowbird, Mar. 2010, pp. 386–395.

[7] M. Reyes, "Cutset based processing and compression of Markov random fields," Ph.D. dissertation, Univ. of Mich., 2011.

[8] H. Landau, "Necessary density conditions for sampling and interpolation of certain entire functions," *Acta Math.*, vol. 117, no. 1, pp. 37–52, 1967.

[9] M. Afonso, J. Bioucas-Dias, and M. Figueiredo, "Fast image recovery using variable splitting and constrained optimization," *IEEE Trans. Image Proc.*, vol. 19, no. 9, pp. 2345–2356, 2010.

[10] H. Takeda, S. Farsiu, and P. Milanfar, "Kernel regression for image processing and reconstruction," *Image Processing, IEEE Transactions on*, vol. 16, no. 2, pp. 349–366, 2007.

[11] X. Zhang and X. Wu, "Image interpolation by adaptive 2-D autoregressive modeling and soft-decision estimation," *Image Processing, IEEE Transactions on*, vol. 17, no. 6, pp. 887–896, 2008.

[12] M. Prelee and D. Neuhoff, "A sampling theorem for Manhattan grids," in *Proc. IEEE ICASSP*, Kyoto, Mar. 2012, pp. 3797–3800.

[13] M. Prelee, D. Neuhoff, and T. Pappas, "Image reconstruction from a Manhattan grid via piecewise plane fitting and gaussian markov random fields," in *Proc. IEEE ICIP*, Orlando, FL, Sept. 2012, pp. 2061–2064.

[14] M. Prelee and D. Neuhoff, "Energy efficient source localization on a Manhattan grid wireless sensor network," in *Proc. IEEE ICASSP*, Vancouver, May 2013, pp. 4266–4270.

[15] ——, "Performance-energy tradeoffs in cutset wireless sensor networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, May 2014, pp. 7585–7589.

[16] J. Unnikrishnan and M. Vetterli, "Sampling high-dimensional bandlimited fields on low-dimensional manifolds," *IEEE Trans. Info. Thy.*, vol. 59, no. 4, pp. 2103–2127, 2013.

[17] ——, "Sampling and reconstruction of spatial fields using mobile sensors," *IEEE Trans. Sig. Proc*, vol. 61, no. 9, pp. 2328–2340, 2013.

[18] M. Prelee, D. L. Neuhoff *et al.*, "Image interpolation from Manhattan cutset samples via orthogonal gradient method," in *Image Processing (ICIP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 1842–1846.

[19] D. Petersen and D. Middleton, "Sampling and reconstruction of wave-number-limited functions in n-dimensional euclidean spaces," *Inf. Control*, vol. 5, no. 4, pp. 279–323, 1962.

[20] A. Rosenfeld and A. Kak, *Digital picture processing*. New York, NY: Academic Press, 1976.

[21] A. Tekalp and A. Tekalp, *Digital video processing*. Prentice Hall PTR Upper Saddle river, NJ, 1995, vol. 1.

[22] D. Dudgeon and R. Mersereau, "Multidimensional digital signal processing," *Prentice-Hall Signal Processing Series, Englewood Cliffs: Prentice-Hall*, vol. 1, 1984.

[23] N. T. Gaarder, "A note on the multidimensional sampling theorem," *Proc. IEEE*, vol. 60, no. 2, pp. 247–248, 1972.

[24] I. Marks and J. Robert, "Multidimensional-signal sample dependency at nyquist densities," *J. Opt. Soc. Amer.*, vol. 3, no. 2, pp. 268–273, 1986.

[25] K. F. Cheung, R. J. Marks II *et al.*, "Imaging sampling below the nyquist density without aliasing," *J. Opt. Soc. Amer.*, vol. 7, no. 1, pp. 92–105, Jan. 1990.

[26] K. F. Cheung, "A multidimensional extension of papoulis generalized sampling expansion with the application in minimum density sampling," in *Advanced Topics in Shannon Sampling and Interpolation Theory*. Springer, 1993, pp. 85–119.

[27] A. Faridani, "An application of a multidimensional sampling theorem to computed tomography," *Contemp. Math.*, vol. 113, pp. 65–79, 1990.

[28] ——, "A generalized sampling theorem for locally compact abelian groups," *Math. Comp.*, vol. 63, no. 207, pp. 307–327, 1994.

[29] D. Walnut, "Nonperiodic sampling of bandlimited functions on unions of rectangular lattices," *J. Fourier Anal. and Appl.*, vol. 2, no. 5, pp. 435–452, 1996.

[30] R. Venkataramani and Y. Bresler, "Perfect reconstruction formulas and bounds on aliasing error in sub-Nyquist nonuniform sampling of multiband signals," *IEEE Trans. Inform. Thy.*, vol. 46, no. 6, pp. 2173–2183, Sept. 2000.

[31] ——, "Optimal sub-Nyquist nonuniform sampling and reconstruction for multiband signals," *IEEE Trans. Inform. Thy.*, vol. 49, no. 10, pp. 2301–2313, Oct. 2001.

[32] H. Behmard and A. Faridani, "Sampling of bandlimited functions on unions of shifted lattices," *J. of Fourier Anal. Appl.*, vol. 8, no. 1, pp. 43–58, 2002.

[33] H. Behmard, "Efficient reconstruction algorithms using shifted lattices," *IEEE Trans. Sig. Proc.*, vol. 57, no. 7, pp. 2548–2557, July 2009.

[34] J. Unnikrishnan and M. Vetterli, "On sampling a high-dimensional bandlimited field on a union of shifted lattices," in *Proc. IEEE ISIT*, 2012, pp. 1468–1472.

[35] A. Papoulis, "Generalized sampling expansion," *IEEE Trans. Circuits and Systems*, vol. 24, no. 11, pp. 652–654, 1977.

[36] P. Willis and Y. Bresler, "Optimal scan for time-varying tomography. I. theoretical analysis and fundamental limitations," *IEEE Trans. Image Proc.*, vol. 4, no. 5, pp. 642–653, May 1995.

[37] H. Behmard, "Reconstruction of 2D signals from unions of shifted lattices," in *Proc. IEEE ICASSP*, vol. 4, 2005, pp. iv–197.

[38] G. Rilling, Y. Tao, I. Marshall, and M. E. Davies, "Multilattice sampling strategies for region of interest dynamic mri," *Magnetic Resonance in Medicine*, vol. 70, no. 2, pp. 392–403, 2013.

[39] T. N. Pappas, "An adaptive clustering algorithm for image segmentation," *IEEE Trans. Sig. Proc.*, vol. 40, no. 4, pp. 901–914, 1992.

[40] S. Z. Li, *Markov random field modeling in computer vision.* Springer Science & Business Media, 2012.

[41] S. Boyd and L. Vandenberghe, *Convex Optimization.* New York, NY, USA: Cambridge University Press, 2004.

[42] J. B. Rosen, "The gradient projection method for nonlinear programming. part I. linear constraints," *Journal of the Society for Industrial and Applied Mathematics*, vol. 8, no. 1, pp. pp. 181–217, 1960.

[43] N. Parikh and S. Boyd, "Proximal algorithms," *Foundations and Trends in Optimization*, pp. 1–96, 2013.

[44] A. Bovik, *Handbook of Image and Video Processing*. Elsevier Science, 2010.

[45] X. Feng and P. Milanfar, "Multiscale principal components analysis for image local orientation estimation," in *Signals, Systems and Computers, 2002. Conference Record of the Thirty-Sixth Asilomar Conference on*, vol. 1. IEEE, 2002, pp. 478–482.

[46] K. V. Mardia, J. T. Kent, and J. M. Bibby, *Multivariate analysis*. Academic press, 1979.

[47] M. Afonso, J. Bioucas-Dias, and M. Figueiredo, "(C)SALSA: A solver for convex optimization problems in image recovery," http://cascais.lx.it.pt/~mafonso/salsa.html, accessed: Jan. 16, 2014.

[48] H. Takeda, S. Farsiu, and P. Milanfar, "Kernel regression-based image processing toolbox for MATLAB," http://alumni.soe.ucsc.edu/~htakeda/KernelToolBox.htm, accessed: Nov. 2015.

[49] X. Zhang and X. Wu, "Image interpolation by adaptive 2D autogregressive modeling and soft-decision estimation." http://www.ece.mcmaster.ca/~xwu/executables/ARInterpolation.rar, accessed: Nov. 2015.

[50] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *Proc. IEEE Annu. Hawaii Int. Conf. Sys. Sci.*, Jan. 2000, pp. 1–10.

[51] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient mac protocol for wireless sensor networks," in *IEEE INFOCOM*, June 2002, pp. 1567–1576.

[52] T. Van Dam and K. Langendoen, "An adaptive energy-efficient mac protocol for wireless sensor networks," in *Proc. ACM Int. Conf. Embedded Networked Sensor Systems*, Nov. 2003, pp. 171–180.

[53] M. Rabbat and R. Nowak, "Decentralized source localization and tracking," in *IEEE ICASSP*, Montreal, May 2004, pp. iii–921.

[54] A. Salhieh, J. Weinmann, M. Kochhal, and L. Schwiebert, "Power efficient topologies for wireless sensor networks," in *IEEE Int. Conf. Parallel Proc.*, Sept. 2001, pp. 156–163.

[55] P. Cheng, C. Chuah, and X. Liu, "Energy-aware node placement in wireless sensor networks," in *IEEE GLOBECOM*, Nov. 2004, pp. 3210–3214.

[56] E. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[57] T. Hales, "The honeycomb conjecture," *Discrete & Computational Geometry*, vol. 25, no. 1, pp. 1–22, 2001.

[58] N. Patwari, J. Ash, S. Kyperountas, A. Hero III, R. Moses, and N. Correal, "Locating the nodes: cooperative localization in wireless sensor networks," *IEEE Sig. Proc. Magazine*, vol. 22, no. 4, pp. 54–69, 2005.

[59] X. Sheng and Y. Hu, "Maximum likelihood multiple-source localization using acoustic energy measurements with wireless sensor networks," *IEEE Trans. Sig. Proc.*, vol. 53, no. 1, pp. 44–53, 2005.

[60] N. Patwari, R. O'Dea, and Y. Wang, "Relative location in wireless networks," in *Proc. IEEE VTC*, vol. 2, May 2001, pp. 1149–1153.

[61] T. Rappaport, *Wireless Communications: Principles and Practice*, 2nd ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2001.

[62] H. Hashemi, "The indoor radio propagation channel," *Proc. IEEE*, vol. 81, no. 7, pp. 943–968, 1993.

[63] N. Patwari, A. Hero, M. Perkins, N. Correal, and R. O'Dea, "Relative location estimation in wireless sensor networks," *IEEE Trans. Sig. Proc.*, vol. 51, no. 8, pp. 2137–2148, 2003.

[64] A. Coulson, A. Williamson, and R. Vaughan, "A statistical basis for lognormal shadowing effects in multipath fading channels," *IEEE Trans. Comm.*, vol. 46, no. 4, pp. 494–502, 1998.

[65] M. Rabbat and R. Nowak, "Quantized incremental algorithms for distributed optimization," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 4, pp. 798–808, 2005.

[66] D. Li and Y. Hu, "Energy-based collaborative source localization using acoustic microsensor array," *EURASIP Journal on Advances in Signal Processing*, vol. 2003, no. 4, pp. 321–337, 2003.

[67] M. Rabbat, R. Nowak, and J. Bucklew, "Robust decentralized source localization via averaging," in *IEEE ICASSP*, vol. 5, 2005, pp. v–1057.

[68] D. W. Matula and R. R. Sokal, "Properties of Gabriel graphs relevant to geographic variation research and the clustering of points in the plane," *Geographical analysis*, vol. 12, no. 3, pp. 205–222, 1980.