Discrete projection: Highly correlated arrays and ghosts for image reconstruction

Matthew Ceko

Supervisors: Imants Svalbe, Timothy Petersen

Monash University
School of Physics & Astronomy

A thesis submitted for the degree of Doctor of Philosophy 2020

Abstract

Whenever measurements of a physical system are made, the quantization implicit in all detection processes forces us to consider a discretisation of the problem. Hence it is important to investigate and understand these discrete systems. The work in this thesis is centred around constructing discrete arrays that have use in discrete tomography for image reconstruction as well as in broader communications applications.

We will use discrete projections to construct large families of so-called perfect arrays, which have ideal correlation properties. This is performed using the finite Radon transform, which allows for lossless transformation between image and projection space. The resultant arrays match perfectly only when exactly aligned, and are poor matches to all other arrays in their families. As we will demonstrate, these arrays are highly valuable in many applications such as watermarking, encryption and communications technologies.

We then construct maximal ghosts, which are arrays consisting of 2^N connected points that have zero sum discrete projections across N directions. In discrete tomography, these points define locations of indeterminate parts of the reconstructed image. Therefore, when discrete tomography is used in digital communications schemes, it is favourable for as many points to be obscured as possible if the security of stored projection data is breached. Maximal ghosts obscure a maximal amount of information. These maximal ghosts can then be used to construct boundary ghosts, which consist only of a thin boundary of ghost points. The errors in boundary ghosts are placed close to the border of the reconstructed image, which is often where less important information lies. Hence, boundary ghosts are ideal for tomographic applications where insufficient information is provided to fully reconstruct an image.

The problem of constructing these ghosts is approached via a recursive tiling. This places self similar discrete tiles at non-degenerate relative shifts to each other, to maximise the length of the contact boundary between adjacent tiles. This maximal joining can be viewed as an energy minimisation, and therefore this tiling may provide insight into physical systems such as the flocculation of lattice structures. The tiling is initially performed in two dimensions, and is then generalised to higher dimensions.

Finally, a discrete reconstruction algorithm is presented which operates in linear time. This algorithm efficiently reconstructs a function from its line sums, provided that the projections are noise free and the function takes values in a unique factorization domain, such as the real or integer numbers. When this algorithm is applied using the projection set which defines a boundary ghost, it prescribes an efficient solution to find all values inside and outside the ghost domain.

Declaration

This thesis contains no material which has been accepted for the award of any other degree or diploma at any university or equivalent institution and that, to the best of my knowledge and belief, this thesis contains no material previously published or written by another person, except where due reference is made in the text of the thesis.

Acknowledgements

First and foremost I would like to thank my supervisors, Imants Svalbe and Tim Petersen. Their leadership and tutelage has made this experience both enjoyable and enlightening. I cannot express enough appreciation for their dedication of time and knowledge towards myself and this project. This would not have been possible without their continuous and open support. I have learned and grown so much over the past few years through them.

I would like to express gratitude towards Rob Tijdeman for his willingness to collaborate, and devote so much of his time to this work. It has been a great honour to learn from such an esteemed researcher in a field that was outside of the focus of our group at Monash. I greatly value his de facto supervision throughout my candidature. I look forward to being able to collaborate in person in the near future.

I wish to also thank Silvia Pagani for her collaboration and encouragement. She has greatly helped deepen my understanding of discrete tomography. I am grateful to have been a part of this research group, and I hope we will have the chance to collaborate further.

This research was supported by an Australian Government Research Training Program (RTP) Scholarship and the J. L. William Scholarship from the School of Physics and Astronomy at Monash University.

Contents

CHAPTER 1

Introduction

Over the past century, methods of collecting and communicating data have changed drastically. The precision of information that can be collected in addition to the speed and bandwidth with which it can be transmitted has grown exponentially. This is due to developments in both the physical technology, as well as the theoretical methods used. A major contributor to these technologies has been the field of tomographic reconstruction. Tomography involves reconstructing an object from its projections at a range of viewing angles. While the mathematical foundation for tomography was established in 1917, it would take until 1971 for the first patient brain scan to be performed. The variations and applications of tomography have been widespread, from medical diagnostics and materials analysis to communications technology and encryption. Today, tomographic imaging is a staple in medical examination as well as scientific research due to its accuracy and non-invasive methodology. Particular implementations of tomography have very close links to algebra and number theory, which lead to tomographic methods being employed in communications and for encryption.

The work of this thesis focuses on tomography over a discrete domain. The problem of determining matrices from their row and column sums has long been considered in mathematics. This problem would grow to the study of discrete tomography we know today. It would seem natural to derive results for discrete tomography purely from tomography over a continuous domain, but as we shall explore, it can be useful to approach tomography using the foundation of discrete mathematics and discrete geometry. There are many interesting results that come from approaching discrete tomography in this manner that would not be possible using continuous tomography. We will look at constructing arrays using discrete tomography techniques that have a wide range of uses such as communications, watermarking, encryption, and of course, imaging. An overview of this thesis is as follows:

Chapter 2 will provide the preliminary theory on discrete tomography that shall be used throughout the rest of this work. Here, we first take a look at the origins of continuous tomography and its limitations. We then show how this problem can be discretised, and present some of the advantages in doing so. The notion of a discrete lattice is formalised, followed by discrete projections. An overview of algebraic methods for reconstruction using discrete tomography is then introduced. Finally, this information is tied together in presenting an example of a discrete tomographic scheme known as the Mojette transform. We shall outline some uses of discrete tomography that not only apply to image reconstruction, but many aspects of communication technologies.

Chapter 3 uses a special formalism of discrete tomography to build structured arrays. Here, we use a scheme that implements periodic projections, which wrap around the lattice to ensure optimal sampling with no redundancy. Under this tomographic transform, we can move between the image space and projection space losslessly. By starting in the projection space, we can use property preserving aspects of discrete projection to build arrays that look self-similar when aligned, but completely dissimilar if they are misaligned at all. We can construct many of these arrays to form a family which all have the property of being perfectly self-similar, but very dissimilar to all other family members. This property makes them highly valuable for watermarking, or scrambling keys for encryption. By viewing these arrays as quadratic residue sequences, we are able to extend these arrays to any number of dimensions.

Chapter 4 explores the placement of errors in discrete tomographic reconstructions. The form of these errors are arrays known as ghosts, which are null projection sets that are invisible when viewed along a given set of projection directions. To begin, we construct ghosts which define errors to maximally obfuscate reconstructions. This is performed through a recursive lattice tiling. Due to the structure of these patterns, we can remove interior ghost points to leave only a boundary of errors that will only impact the outer edges of a reconstruction. Using these projection directions, the goal is to have reconstructions where the errors are placed in less important parts of an image, leaving the object clear and accurately reconstructed. A new result is presented that links the convex hull area of a ghost to the number of projection bins it occupies, which can be viewed as a version of Pick's theorem for discrete projection. This relationship is used to give some properties and bounds on the size of these ghost patterns.

The construction of maximal and boundary ghosts is then extended to three dimensions, and higher in Chapter 5. We show that the properties derived in two dimensions naturally extend to higher dimensions. The additional degrees of freedom in higher dimensions give rise to a wider variety of possible ghost shapes. Computations are given which show that the geometric placement of maximal ghosts tiles inherently maximises the joining boundary of the shapes. Hence, these ghost tiles may provide insight into the formation of some lattice structures.

Finally, the problem of reconstruction methods is considered in Chapter 6. Here, we present an efficient algorithm which reconstructs an unknown function from its line sums in linear time with respect to the grid size and number of projection directions. This is possible whenever the line sums are consistent and the function takes values in a unique factorization domain. This algorithm can be extended to higher dimensions if the set of projection directions satisfy certain conditions.

CHAPTER 2

Background

Tomography is a ubiquitous method of imaging in both the medical and scientific world today. Only a century ago, obtaining information about the internal parts of an object was, by necessity, an invasive and often destructive procedure. This began to change in 1917, when Johann Radon showed that a function can be represented by the set of its integral projections [58, 59]. As we will show, the transform that Radon provided was ill-posed, as it requires an infinite set of projections for exact inversion. In 1937, Stefan Kaczmarz showed the convergence of an algorithm for approximate solutions to systems of linear equations, which meant that the Radon transform could be used to approximate a solution from a finite number of projections [46]. This new insight, combined with the experimental work of Allan McLeod Cormack led to the invention of the first commercial computerized tomography (CT) scanner by Sir Godfrey Hounsfield, and the first patient brain-scan performed in 1971 [20, 21]. This method has a wide range of applications, from medical scans to geology and astrophysics [45, 50]. The first results in discrete tomography were published in 1957 by Ryser, who gave sufficient and necessary conditions for the reconstruction of binary matrices from their row and column sums, and an algorithm for performing these reconstructions [61]. The field has evolved greatly since then, and many different discrete tomographic situations have been studied. In this chapter, we will outline the mathematical basis for tomography, and then restrict the domain to a discrete lattice.

2.1 The Radon Transform

The Radon transform underpins all of tomography. In this section we provide an overview of the Radon transform in two dimensions, which could be considered as a slice in 3D. The goal of tomography is to reconstruct a 2D function $f : \mathbb{R}^2 \mapsto \mathbb{R}$ from a set of 1D projections. The projections are integrals along

the set of lines defined by angle $\theta \in [0, \pi)$ and translation $\rho \in \mathbb{R}$. These lines have the equation

$$(x(s), y(s)) = (\rho\cos(\theta) - s\sin(\theta), \rho\cos(\theta) + s\sin(\theta)) \tag{2.1}$$

for $s \in \mathbb{R}$. The Radon transform of function $f(\rho, \theta)$ is given by

$$\mathcal{R}[f(\rho, \theta)] = \int_{-\infty}^{\infty} f(\rho\cos(\theta) - s\sin(\theta), \rho\cos(\theta) + s\sin(\theta))\, ds. \tag{2.2}$$

Note that $\mathcal{R}[f(\rho, \theta)] = \mathcal{R}[f(-\rho, \theta + \pi)]$, which gives the restriction on θ. The function f is assumed to be finite and have compact support. A function with compact support has zero value outside of a compact set, which is a reasonable constraint to impose when imaging an object. In the case of medical imaging, for example, we are not interested in the area outside of the patient.

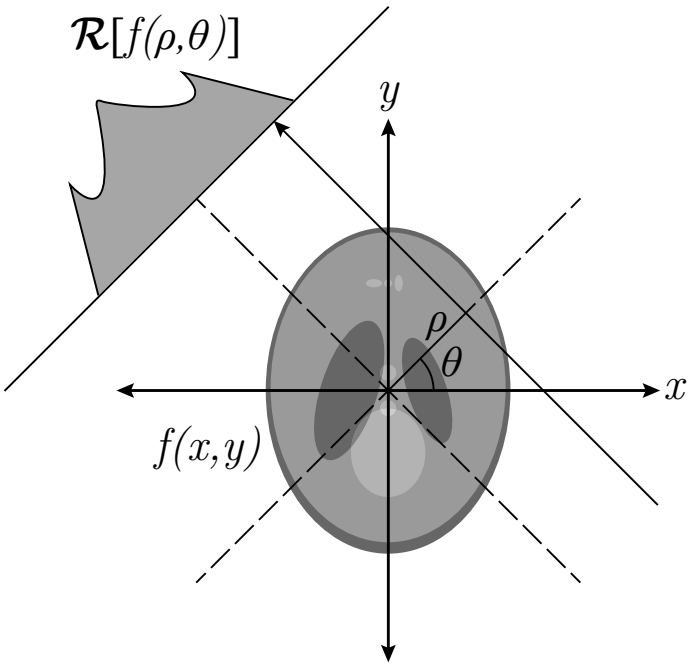


Figure 2.1: Radon transform geometry.

We step away briefly from the Radon transform to present the Fourier transform and show that there is a strong link between them that is essential to tomography. The Fourier transform decomposes a function into its component sine and cosine frequencies. This gives the frequency spectrum of the function. The 2D Fourier transform on function f is given by

$$\mathcal{F}[f(u, v)] = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x, y)e^{-i2\pi(ux+vy)}\, dx\, dy. \tag{2.3}$$

The function f can be recovered from its Fourier transform provided it is continuous and absolutely integrable.

$$f(x, y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \mathcal{F}[f(u, v)]e^{i2\pi(ux+vy)}\, du\, dv \tag{2.4}$$

To show the link between the Radon and Fourier transforms, consider the projection for angle $\theta = 0$. In this case, the Radon projection reduces to

$$\mathcal{R}[f(\rho, 0)] = \int_{-\infty}^{\infty} f(\rho, y)\, dy. \tag{2.5}$$

We now take the Fourier transform of both sides of (2.5).

$$\begin{aligned} \mathcal{F}[\mathcal{R}[f(\rho,0)]] &= \int_{-\infty}^{\infty} \left[\int_{-\infty}^{\infty} f(\rho, y)\, dy \right] e^{-i2\pi ux}\, dx \\ &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) e^{-i2\pi ux}\, dx\, dy \\ &= \mathcal{F}[f(u, 0)] \end{aligned} \tag{2.6}$$

Since the Fourier transform is invariant under rotation, this holds for all θ. This is known as the Fourier Slice Theorem, which states that the 1D Fourier transform of a projection at angle θ is equal to the slice of $\mathcal{F}[f(u, v)]$ that passes through the origin at angle θ. Therefore, each projection (given by the rows of the sinogram in Fig. 2.2b) allows us to recover a slice in Fourier space (Fig. 2.2a).

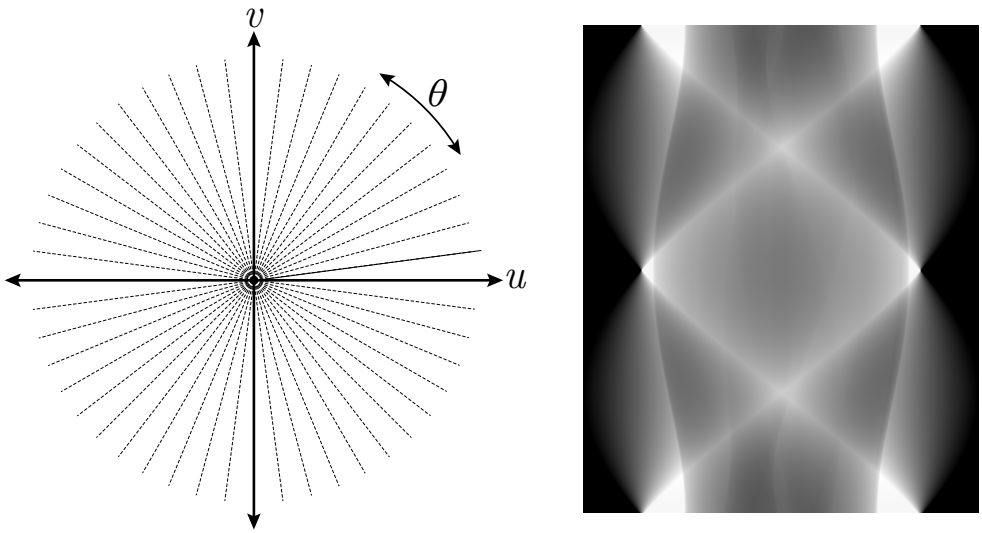


Figure 2.2: a) Each dashed line, or slice, in Fourier space is given by a projection (row) of the sinogram (b).

This result highlights a clear problem in the direct application of the Radon transform. That is, information is only available along radial lines of Fourier space. Hence, to recover all points in Fourier space, we require an infinite number of projections and the space is not uniformly sampled. While many different methods exist to deal with this problem, this work uses discrete projections.

2.2 Algebraic Reconstruction Techniques

In any practical imaging system, the problem of discretisation must be considered. Whether it be the finite number of bins that are used to take an image, the finite levels of intensities being measured, or even a priori assumptions about the densities of objects being imaged, there are always discrete levels when measurements are made. Here, we take a look at algebraic reconstruction techniques, first presented in [38].

We define a $P \times Q$ grid $\mathcal{A} = \{(i,j) \in \mathbb{Z}^2 : 0 \leq i < P, 0 \leq j < Q\}$ and a function f with PQ unknown values which represents an image. Each image point f_j is referred to as a pixel (or voxel in 3D), that has an associated value within the pixel. Under this scheme, each projection r_i is a linear combination of pixel values f_j, where weights w_{ij} are applied to each pixel based on the fractional area of the intersecting ray. For each detector we have

$$r_i = \sum_j w_{ij} f_j \tag{2.7}$$

for all ray sums over all projections. This defines a set of linear equations which can be written in matrix form as

$$\boldsymbol{W}\boldsymbol{f} = \boldsymbol{r} \tag{2.8}$$

where $\boldsymbol{W}$ is a projection matrix, $\boldsymbol{f}$ is the image in vectorised form and $\boldsymbol{r}$ is a vector of detected projections. In general, the projections will have noise, which we can model through a perturbation term ϵ as $\tilde{r}_i = r_i + \epsilon$. Algebraic reconstruction methods must therefore compute an approximate solution by minimising the cost function:

$$\chi = \min_f \|\boldsymbol{W}\boldsymbol{f} - \tilde{\boldsymbol{r}}\| \tag{2.9}$$

Often, the l^2-norm is chosen to penalise the difference between computed and measured projections. But this can change depending on the application.

There exist many variations to this technique, built upon these basic principles. In the simultaneous iterative reconstructive technique (SIRT), projections are performed simultaneously and then averaged. This usually provides more accurate reconstructions but with slower convergence. The simultaneous algebraic reconstruction technique (SART) improves upon the method by simultaneously applying error correction terms to all rays in a projection, and includes longitudinal weighting of the correction terms for back-projection [3]. A discrete version of SART, termed DART, has also been developed. This algorithm uses a continuous method such as ART, SIRT or SART as a subroutine, with an additional discretization step [7]. Many modern tomography software packages include these algorithms in their framework, such as TomoPy and the ASTRA toolbox [42, 72, 73].

2.3 Discrete Projection

Instead of computing ray sums by weighting pixels with respect to their contact with the ray, we can instead choose to only sum pixels that lie exactly on a line. We define the notion of a lattice direction $v = (p, q)$ where $p, q \in \mathbb{Z}$ and $\gcd(p, q) = 1$. Integers p and q must be co-prime to ensure there are no degenerate projection directions. The lattice direction is used to define a discrete line $jp = iq + b$ across the lattice, where $b \in \mathbb{Z}$. A discrete projection is defined as the set of all line sums along a lattice direction (p, q). An example of a $(2, 1)$ discrete line sum is illustrated in Fig. 2.3. The full projection would then be computed from all parallel $(2, 1)$ line sums.

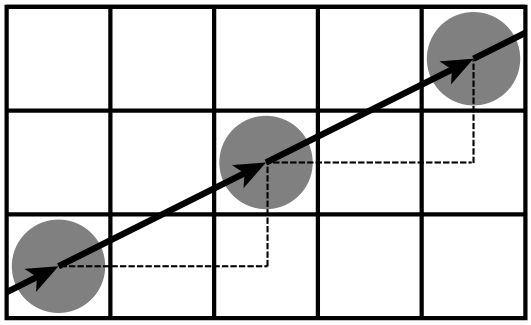


Figure 2.3: Discrete projection for direction $(2, 1)$.

Discrete projections can also be computed with periodic boundary conditions, as shown in Fig. 2.4. While this may not be a practically feasible method yet, we shall see in Chapter 3 that periodic projections are quite useful as they enable projections with zero redundancy when the arrays have a prime number of rows and columns.

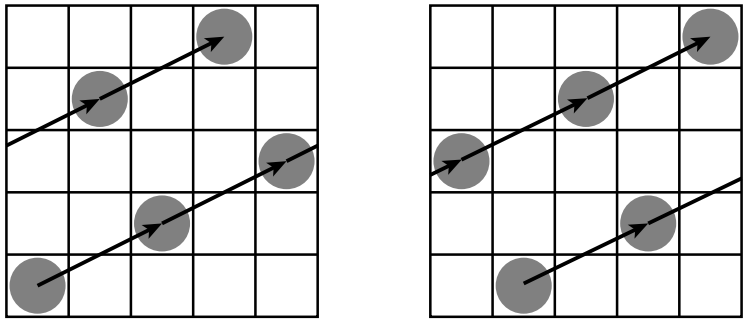


Figure 2.4: Periodic discrete projections for direction $(2, 1)$ on a 5×5 array starting from two different locations.

There are a number of discrete projective transforms that use either periodic or non-periodic projections, although they all define a set of linear equations that can be inverted given adequate projection information.

2.4 The Mojette Transform

The Mojette transform is an example of a discrete Radon transform that uses discrete projections for reconstruction. This can allow for exact tomographic inversion in the absence of noise. Projections under the Mojette transform are discrete, and sum pixel values at sites separated by p steps in the x direction, and q steps in the y direction. Each projection sum is stored in a projection bin. The word "Mojette" comes from a type of bean used to teach children counting and arithmetic, and is a play on the words "bean" and "bin".

Under the Mojette transform, angles are chosen from the discrete set $\theta = \tan^{-1}(q/p)$. Each projection is the sum of the pixel values that intersect the line $b = -qk + pl$. The Dirac-Mojette operator [40] acts on image f by

$$[M_\delta f](b,p,q) = proj_\delta(b,p,q) = \sum_{k=-\infty}^{+\infty} \sum_{l=-\infty}^{+\infty} f(k,l)\delta(b+kq-pl) \qquad (2.10)$$

where $(b,p,q) \in \mathbb{Z}$, and $\delta(b)$ is the Kronecker delta which has value $\delta(b) = 1$ if $b = 0$, otherwise $\delta(b) = 0$. The Kronecker delta is used to sample pixels that exactly fall on the projection ray. A Mojette transform is demonstrated in Fig. 2.5.

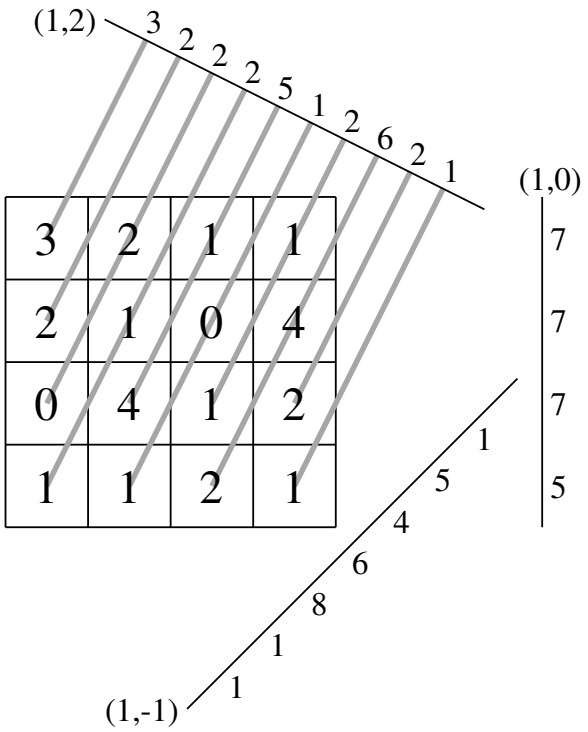


Figure 2.5: Mojette transform with the set of projections $\{(2,1), (1,0), (1,-1)\}$ on a 4×4 array.

The inverse of the Mojette projector M is the Mojette back-projector M^*. In the Dirac-Mojette model, the Mojette back-projector M_δ^* is defined as

$$[M_\delta^* proj_{(p,q)}](k,l) = \sum_{i=-\infty}^{+\infty} \sum_{j=-\infty}^{+\infty} \sum_{b\in B} \sum_{(p,q)\in S} \delta(k-i)\delta(l-j)proj(b,p,q)\delta(b+qi-pj) \tag{2.11}$$

where S is the set of (p,q) directions. Again, the sifting property of the Kronecker delta is implemented to place projections back at the correct pixel location.

For a set of non-degenerate projections $\{(p_i,q_i)\}$ on a $P\times Q$ array, unique reconstruction of any image $I:\mathcal{A}\rightarrow\mathbb{R}$ is possible if and only if the Katz criterion is met [47].

$$\sum|p_i|\geq P \quad \text{or} \quad \sum|q_i|\geq Q \tag{2.12}$$

In this case, the image is uniquely determined by the discrete projections and can be reconstructed efficiently [51, 52]. Geometrically, the Katz criterion states that if the vector $(\sum|p_i|,\sum|q_i|)$ can fit inside the $P\times Q$ array, then it is not possible to reconstruct all pixels from the projection set. For example, if we consider a 4×3 lattice then the set of projections $\{(0,1),(1,1),(1,0)\}$ does not satisfy the reconstructability requirement while $\{(1,0),(1,1),(2,1)\}$ does (Fig. 2.6).

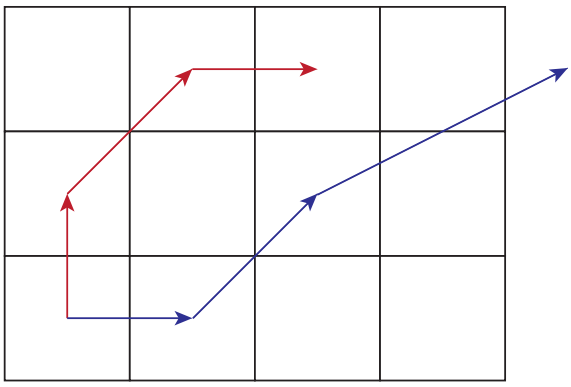


Figure 2.6: Projection vectors $\{(0,1),(1,1),(1,0)\}$ (red) and $\{(1,0),(1,1),(2,1)\}$ (blue) for a 4×3 lattice.

When the Katz condition is not met reconstruction is still possible, but some values in the reconstructed array do not have a unique solution. The locations of these indeterminate values will be discussed in depth in Chapter 4. Multiple iterations of inversion methods can be applied by setting arbitrary values to indeterminate pixels. Numerical inversion using a procedure such as the conjugate gradient method [62] will yield a similar result.

Under the Mojette transform, pixels are sampled multiple times when there is more than one projection. This redundancy in the projection information

was initially viewed as a problem. More recently however, it was discovered that the redundancy could be taken advantage of in many communications and security applications [17]. We shall outline a few of these applications briefly to demonstrate that discrete tomography has a wide range of uses far outside the context of imaging.

When information is communicated via the internet, it is done through packets of bytes under some agreed upon protocol. Partitioning large amounts of information into smaller packets allows for more efficient and robust communication. Each packet can be communicated through the most efficient route at the time it is being sent. And if any packet is lost, having redundant information can ensure all information is correctly delivered to the destination. Projections of the Mojette transform naturally creates packets of information, and the redundancy in the projections can be used to repair damaged or lost bits of information. Also, the fact that each projection is associated with a direction adds an additional layer of security, as this link must be known for the projection to be useful.

The instability of the inverse Mojette transform can be exploited for the purpose of security and encryption [5]. Erroneous projection bins propagate misinformation quickly through the inverse Mojette transform, which is favourable for encryption. Encryption keys can be generated to corrupt specified bins, and without knowledge of the key, the reconstructed data is rendered useless. Similarly, shared keys can be used to generate Mojette based watermarks for images. In this scheme, phantoms are added to an image with zero projections for a given set of directions. Each user has knowledge of the directions used to generate the phantom, and its position.

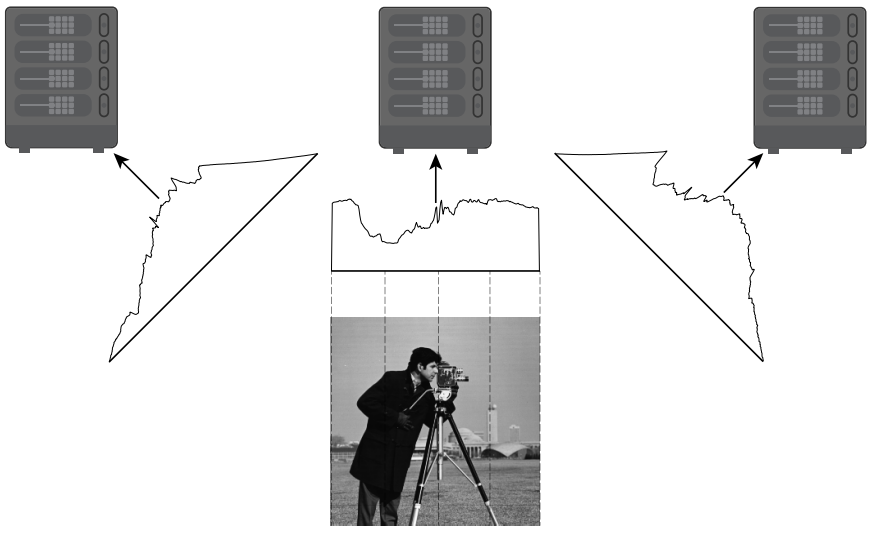


Figure 2.7: An image being split into projections to be stored on separate, remote servers. Data from a subset of servers can be used to recover the original information.

The Mojette transform has been implemented into a secure data file system known as RozoFS [54]. For this application, discrete projections are used to split a set of data to be stored on multiple remote servers. This method is more efficient than simply replicating data, which reduces the cost of data storage. If the security of some (but not all) of the servers becomes compromised, we want to ensure the least amount of information is recoverable. However, an end user can recover their data perfectly by accessing a minimum subset of required servers. Since the information can be recovered from a subset of the total projections, server load can be reduced by allocating multiple users to different servers. Careful choice in the optimum sets of projection angles used is a topic of this thesis.

CHAPTER 3

Perfect Arrays

Large sets of distinct arrays of variable size that possess both strong auto-correlation and weak cross-correlation properties are highly valuable in many imaging and communications applications. In this chapter, we use the discrete finite Radon transform to construct $p \times p$ arrays with "perfect" correlation properties, for any prime p. Array elements are restricted to the integers $\{0, \pm 1, +2\}$. Each array exhibits perfect periodic auto-correlation, having peak correlation value p^2, with all off-peak values being exactly zero. Each array contains just $3(p-1)/2$ zero elements, the minimum number possible using this alphabet. Large families with size $M = p^2 - 1$ of such arrays can be constructed. Each of the $M(M-1)/2$ intra-family periodic cross-correlations is guaranteed to have one of the three lowest possible merit factors. This family of size M can be extended to multiples of p^2-1 if we permit more than the three lowest cross-correlation levels. Using Legendre quadratic residue sequences, we generalize the idea to nD and build families of perfect arrays that have prime side lengths p, with lower cross-correlation. Affine transformations extend the perfect nD Legendre array family size to at least $(p-1)^n$.

Large families of arrays with perfect correlation properties are highly desirable for any application that calls for many different arrays that can be used as unique identifiers or masks. In this chapter, we consider families of $p \times p$ arrays, where p is prime. The periodic cross-correlation between two arrays f and g (written $f \otimes g$) is the set of correlations $\{C_{f,g}(r,s) | 0 \le r \le p-1, 0 \le s \le p-1\}$ where

$$C_{f,g}(r,s) = \sum_{x=0}^{p-1} \sum_{y=0}^{p-1} f(\langle x+r \rangle_p, \langle y+s \rangle_p) \cdot g(x,y) \tag{3.1}$$

and $\langle j \rangle_p$ denotes j modulo p. Then the aperiodic cross-correlation is the set $\{C_{f,g}(r,s) \; | 1-p \leq r \leq p-1, 1-p \leq s \leq p-1\}$ where $x+r$, and $y+s$ are not computed modulo p and values outside of $0 \leq x+r \leq p$ and $0 \leq y+s \leq p$ are assumed to be zero. When $f = g$, the cross-correlation is referred to as the auto-correlation. When we speak of perfect arrays, we require ideal auto-correlation, $C_{f,f}(r,s) = 0$ for all $r, s \neq 0$, and low cross-correlation values between all family members. We use the merit factor (MF), defined as the square of the peak value divided by the sum of all squared off-peak values, to characterise the cross-correlations. For cross-correlations, we want MF $\ll 1$, and for ideal auto-correlation the MF is infinite. Examples of two 59×59 arrays are displayed in Fig. 3.1, which exhibit perfect auto-correlation and optimally low cross-correlation. In Fig. 3.1a,b, the values $\{-1, 0, 1, 2\}$ have been mapped to greyscale where -1 appears as black and $+2$ as white. Throughout this chapter, we consider the top-left of the array as the origin.

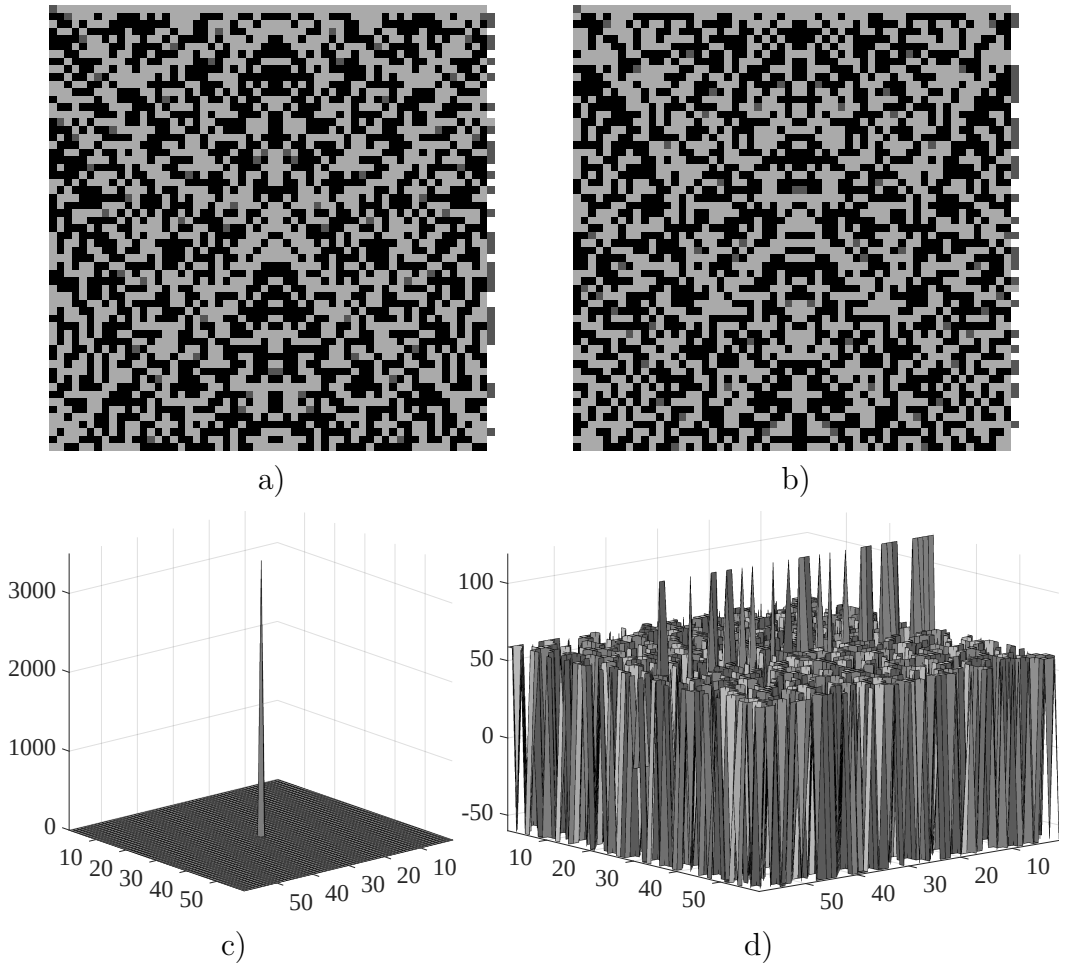


Figure 3.1: a,b) Two $p = 59$ perfect arrays where the values $\{-1,0,1,2\}$ have been shifted to greyscale. c) Ideal perfect periodic auto-correlation of (a) and (b) with peak value of p^2. d) Optimally low cross-correlation between (a) and (b). Cross-correlation values are $-p, 0, p, 2p$.

Large families of arrays with such low correlations are required for many communications, cryptography, and imaging applications. In code-division multiple-access communication systems, each user is prescribed such an array to distinguish themselves from other users while allowing them to communicate simultaneously on the same frequency band [34, 78]. Arrays with high auto- and low cross-correlation properties are also useful as coded apertures for 2D tracking, radar and sonar [35, 39]. They are also used as watermarks for authentication as a means of copyright protection. They can be embedded and hidden in digital media such as audio, images or video through various methods, which can then be retrieved through correlation [22, 69]. There is a vast literature on constructing binary sequences for such purposes (see [57]), however the strict limit of values also restricts the number of possible sequences. So in this work, we slightly relax this restraint in order to construct much larger families of arrays.

Previous work [70] used the finite Radon transform (FRT) to construct $p \times p$ pseudo-noise arrays in families of size $M = p$ (where p is a $4N - 1$ prime) that had optimal periodic auto-correlation and cross-correlation, that meet the Welch correlation bounds [79, 80]. Welch's lower bound for the maximum cross correlation will be detailed later in this chapter. These families of (Legendre) arrays have an alphabet of a single zero element with the remainder being equal numbers of ± 1 elements. Orthogonal families of arrays exist in any even number of dimensions, where they can be balanced. "Grey" versions of these array families were also constructed that have integer alphabets (with integer values ranging between $\pm\sqrt{p}$). Recovery of a grey array, A, embedded in grey data, B, can be advantageous, as $A \otimes (A + B) \approx 2A \otimes A$ if we choose to embed A in those parts of B where $A \approx B$.

Subsequent work [68] extended the size M of these array families to multiples of p, typically $M \approx 3p$. This was done by blending the original array family with distinct arrays either derived from the original array auto-correlations, or with new arrays, also built using the FRT, but with their families generated using different (but equivalent) Hadamard matrices. The only concession made when extending the family size beyond p is that the strength of each cross-correlation now lies in a range of statistically predictable values, at or just above the lowest possible level. However, a large fraction of these array pairings exhibit the lowest possible cross-correlation, with the remaining cross-correlations being close to this lowest value.

Further extension of the size of a family of arrays well beyond p becomes increasingly difficult as it is hard to constrain the range of cross-correlation values. The rapidity of this rise is, in part, due to the depth of the array alphabet. A binary array (or any array with mostly ± 1 values) has a limited number of combinations that can simultaneously sustain high auto- and low cross-correlation. The grey versions of the $p \times p$ Legendre arrays constructed in [70] can support a much larger and more diverse range of well-correlated structures. The combinatorial diversity of grey perfect arrays makes them

significantly more secure and resistant to hacking.

The balance theorem ensures that the sum of the array values dictates the sum over all correlation values [34]. This ensures that alphabets spanning a wide range of greys also require more zero elements in those perfect arrays (see Section 3.2.3). The number of zero elements in a perfect-correlation array increases with the square of the values of each non-zero element. Arrays containing a large number of zero elements have reduced operational efficiency, as the zero terms make no change to the signal in which it is embedded.

In this chapter, we construct families of arrays with the restricted alphabet $\{0, \pm 1, +2\}$, as published in [16]. These arrays have a fixed histogram, where the number of +2s were minimised to $(p-1)/2$, thus requiring $3(p-1)/2$ zero terms in each array. The balance of these array values (always a clear majority) are either ± 1. The presence of a relatively few extra zeroes reduces the efficiency of these arrays, by $\mathcal{O}(1/p)$, but this becomes less significant for large p. Affine transformations are then used as array shuffling operations to produce many complementary perfect arrays. We show here that the pooling of arrays can directly produce a family of size $p^2 - 1$ by selecting a fixed, complementary set of rotation and skew angles, thus avoiding the need to check any cross-correlations. This significantly reduces the time and complexity to build families for large p, which is of the order $\mathcal{O}(p^4)$.

We also present additional seed families, each of size $p^2 - 1$. Including these new families enlarges the final family size to multiples of $p^2 - 1$, provided the threshold for the lowest permitted cross-correlation is relaxed. These arrays can be interpreted as shifted quadratic residue sequences. This formulation allows the construction of perfect arrays to an arbitrary number of dimensions, withstanding a minor concession to the alphabet used.

In Section 3.1, we outline operations that preserve correlation properties. This allows us to start with a sequence that is known to have ideal auto-correlation and to create new arrays that maintain this property. Section 3.2 presents an algorithm that can be used to create a family of $M = p^2 - 1$ arrays which all have ideal auto-correlation, and guaranteed low cross-correlation between family members. The histogram of these arrays is analysed to predict the different cross-correlation levels that can occur. In Section 3.3, we look at ways to extend the family size beyond $p^2 - 1$ by relaxing the histogram values, and consequently a slightly wider spread of cross-correlation levels. Section 3.4 offers some applications for such arrays to detail the scope of their use. We implement them into basic watermarking and encryption schemes, as well as explain their use for multiple access communications. Finally, we use quadratic residue sequences as an alternative construction for perfect arrays in Section 3.5. This new perspective allows us to generalise perfect arrays to any dimension n, and then by using affine transformations we can extend the size of a perfect nD perfect array family to at least $(p-1)^n$.

3.1 Correlation Preserving Operations

3.1.1 Discrete Projection

A discrete projection sums the values located at all points on a lattice that intersect a given straight line. We consider projections of a $p \times p$ image $I(x, y)$ as the sum of all grid points that intersect the line

$$x = \langle t + my \rangle_p \tag{3.2}$$

where $m, t \in \mathbb{Z}$. The discrete central slice theorem [40, 55] states that the $(n-1)$D projection corresponds to a central section (slice) of the nD Fourier transform orthogonal to that direction. Hence the $(n-1)$ dimensional projected views of a distribution preserve the nD Fourier transform of the distribution. As a corollary of the central slice theorem, moments and correlations of a distribution are also preserved under projection. This means, for example, that the auto-correlation of a 1D projection of an image is equal to the 1D projection of the 2D autocorrelation. The projections of any distribution inherit that distribution's correlation properties [68]. The reverse is also true, meaning that the back projection of 1D sequences with perfect auto-correlation would yield a 2D array with the same properties.

3.1.2 Finite Radon Transform

The finite Radon transform [49] is a discrete projective transform used to provide a unique and exact reconstruction of any 2D $p \times p$ object from its $p+1$ 1D cyclically wrapped projected views. These projections are stored as rows of a $p \times (p+1)$ array in Radon space. Since p is prime, the periodic projections are uncoupled, as each projection fully tiles a $p \times p$ array, exactly once, at all positions, in a distinct pattern. This provides optimal sampling, as there is no redundancy in the projection information. The FRT is exact, which allows for lossless transformations between image and projection space. The FRT is similar to the Mojette transform introduced in Section 2.4, but with periodic projections that ensure there is no redundancy.

Projection $R(t, m)$ of an image $I(x, y)$ starts from translate t, $0 \leq t \leq p-1$. By convention, $t = 0$ is the first of p parallel rays in a projection, where each ray begins from a position on the top row of a $p \times p$ array (Fig. 3.2). Each 1D projection is comprised of p parallel rays, where each ray sums p pixel values in $I(x, y)$ that are located at p steps beginning from t, each step being m pixels across and one pixel down, wrapping periodically around the ray as required, where $0 \leq m \leq p$ (Fig. 3.3). Projections 0 and p are column and row sums of $I(x, y)$, respectively. The FRT of image I is therefore written as:

$$\mathrm{FRT}(I(x,y)) = R(t,m) = \begin{cases} \sum_{y=0}^{p-1} I(\langle t + my \rangle_p, y) & \text{for } 0 \leq m \leq p-1 \\ \sum_{x=0}^{p-1} I(x,t) & \text{for } m = p \end{cases} \tag{3.3}$$

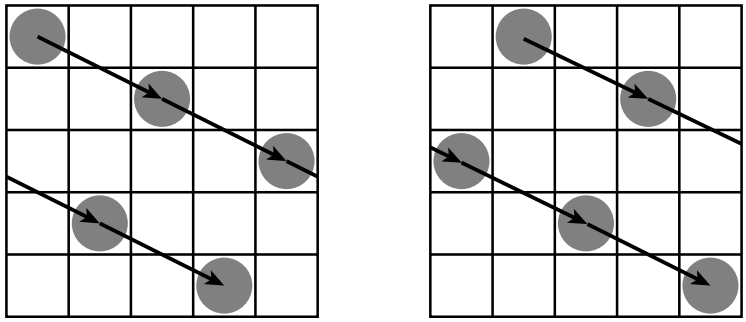


Figure 3.2: FRT projections with $m = 2$ for $t = 0$ and $t = 1$.

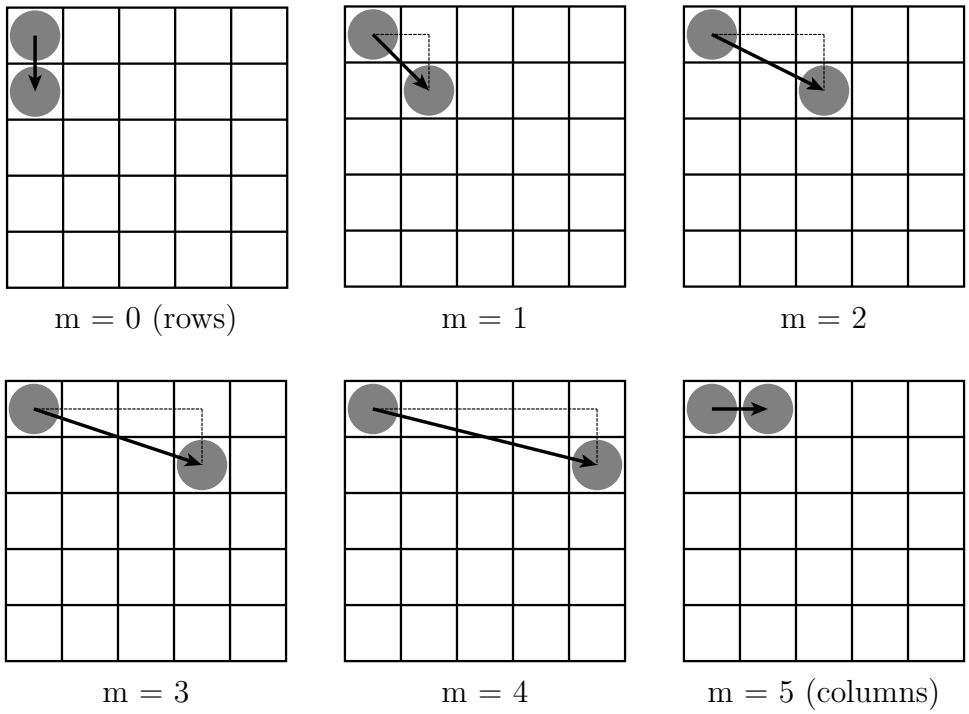


Figure 3.3: FRT projections for a 5×5 array.

Each projection m maps to a unique discrete angle (i, j), where $i, j \in \mathbb{Z}$ and $\gcd(i, j) = 1$, such that the set of $p + 1$ angles is fixed for each array size p [64]. Back-projecting each of the $p + 1$ 1D projections across a zeroed $p \times p$ array at the complemented angles and normalising the result recovers, exactly, the 2D data that was projected. Each value is equally weighted in the projection, and each pixel contributes exactly once to each projection. This allows for both exact and efficient inversion. This means no back-projection filter is required as opposed to many other tomographic methods.

We use $\text{FRT}^{-1}(R(t,m))$ to signify the inverse FRT of $R(t,m)$.

$$\text{FRT}^{-1}(R(t,m)) = I(x,y) = \frac{1}{p}\left[R(y,p) - I_{sum} + \sum_{m=0}^{p-1} R(\langle x - ym\rangle_p, m)\right] \tag{3.4}$$

where I_{sum} is the sum of all values in I. I_{sum} can be found by the sum of any row in R, as each FRT projection samples every array point exactly once. An example of the FRT is illustrated for an arbitrary 3×3 array in Fig 3.4.

A	B	C
D	E	F
G	H	I

$I(x,y)$

$\longleftrightarrow$

A+D+G	B+E+H	C+F+I
A+E+I	B+F+G	C+D+H
A+F+H	B+D+I	C+E+G
A+B+C	D+E+F	G+H+I

$R(t,m)$

Figure 3.4: An image $I(x,y)$ of size $p \times p$ has an FRT $R(t,m)$ of size $(p+1) \times p$.

The FRT represents a 2D image exactly by its 1D projections, and thereby obeys the discrete central slice theorem. As a result, each slice is the 1D discrete Fourier transform of the periodic projection. Hence, a 2D array with perfect auto-correlation properties can be constructed by projecting a set of 1D FRT projections, with each projection having perfect auto-correlation. The reconstructed 2D object will inherit the correlation properties of the ensemble of 1D arrays [68, 70].

This correlation property is displayed in Fig. 3.5. In Radon space, we have shown two arrays built from $p+1$ shifted Kronecker deltas of amplitude p, indicated by the black locations in Fig. 3.5c. Each row in Radon space represents a projection of the image. These are back-projected through the inverse FRT to produce arrays in image space with values $\{0, \pm 1, +2\}$, which have been mapped to greyscale values in this Fig. 3.5b. Due to each array having discrete FRT projections with ideal auto-correlation, the 2D images will also have ideal auto-correlation. The cross-correlation in 2D can be computed equivalently in FRT space as the 1D cross-correlations of each row. Image space and FRT space are also linked through Fourier space (Fig. 3.5a). The 2D Fourier transform of each image can also be computed through the inverse FRT of the 1D Fourier transforms for each row in FRT space. The cross-correlation can be computed via a pointwise multiplication in Fourier space. In Fig. 3.5a,d, the greyscale values correspond to the phase of the FFT, as the amplitude of each frequency is always constant for both 1D and 2D cases.

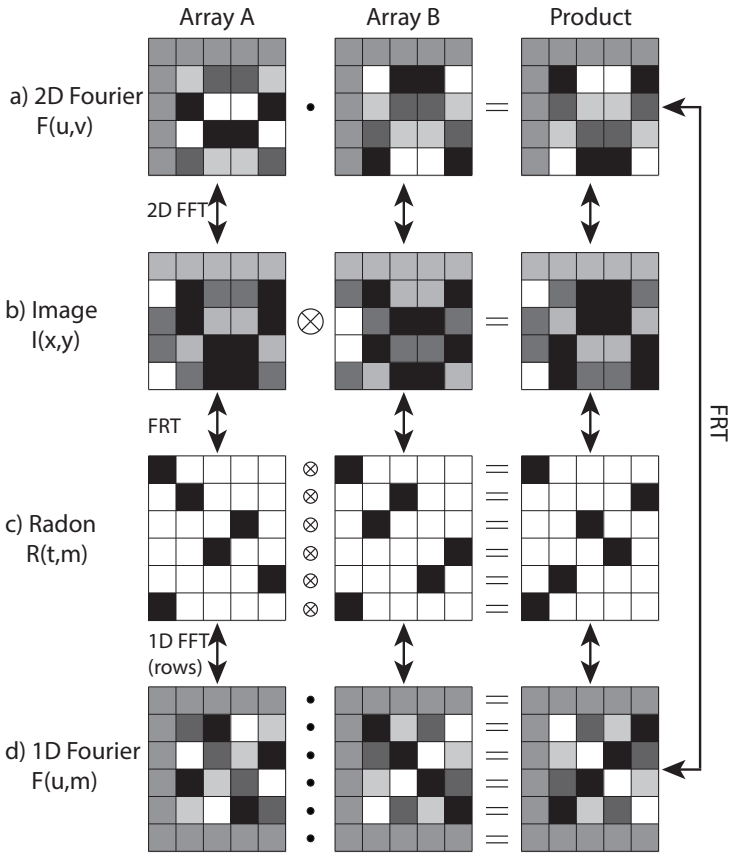


Figure 3.5: Cross-correlation of $p = 5$ perfect arrays. a) Cross-correlation in image space is equal to the pointwise product in Fourier space. Each value shows the Fourier phase. b) Cross-correlation of two perfect arrays, resulting in another perfect array. c) Perfect arrays in FRT space, which are shifted Kronecker deltas of amplitude p. The convolution of each row corresponds to the difference in shift. d) Row-wise 1D FFT of each shifted delta function. Grey values represent phase.

3.1.3 Affine Transformation

A 2D affine transformation (reversibly) maps each pixel (x, y) of a prime $p \times p$ 2D image to a unique new location (x', y'). The new coordinates are obtained via matrix multiplication in homogeneous coordinates (modulo p):

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & e \\ c & d & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{3.5}$$

where the values, $0 \leq a, b, c, d, e, f \leq p - 1$, are arbitrary integer transform coefficients, provided that the upper matrix $[a\ b; c\ d]$ has non-zero determinant (modulo p). The coefficients e and f serve as a discrete translation vector. Periodic translations are trivial operations for periodic arrays, and therefore coefficients e and f are set to zero to avoid producing degenerate copies of arrays. When $[a\ b; c\ d] = [j\ -i; i\ j]$, the affine transform rotates the array by

the discrete angle (i, j). When $[a\ b;\ c\ d] = [j\ i;\ i\ j]$, the affine transform skews the array by vector (i, j).

These operations are exact modulo mappings within the array, thereby constituting a unique and reversible reshuffling of array elements. Due to the homogeneity of the affine transformation, the auto-correlation values of an array remain unchanged, their positions are simply permuted. The values outside the central peak are shifted under the same affine transformation, with the same peak at the origin. For perfect arrays, all auto-correlation values for non-zero shifts are zero, hence the auto-correlation of a perfect array is invariant under affine transformation.

For $p \times p$ arrays, we know in advance the exact set of angles (i, j) that correspond to the complete set of $p + 1$ discrete projections of the FRT for a $p \times p$ array [64]. If we avoid the simple axial rotations (1,0) (90°) and (0,1) (0°), we can, without redundancy, rotate each original array A by affine coefficients (i, j) to obtain $p - 1$ distinct copies A' of each A, whilst preserving the original correlation properties. For a $p \times p$ array, an affine transformation with zero translation coefficients can be implemented as presented in Algorithm 1.

Algorithm 1 Affine transformation of a $p \times p$ array A.

```
1: function AFFINE(A, a, b, c, d)
2:     for x := 0 to p do
3:         for y := 0 to p do
4:             T(⟨ax + by⟩_p, ⟨cx + dy⟩_p) ← A(x, y)
       return T
```

3.2 Families of Perfect Arrays

In this section, we detail the construction of families of $M = p^2 - 1$ perfect $p \times p$ arrays using the alphabet $\{-1, 0, +1, +2\}$. Section 3.2.1 employs the FRT to construct perfect arrays using distinct cyclic shifts of a delta function, where the translate t in $R(t, m)$ is given by $t = \langle m^n \rangle_p$ for selected values of n. Section 3.2.2 shows how affine transformations can then be applied to extend the size of the array family, while preserving the ideal auto-correlation properties of each new array. We then explore the histogram of array values in Section 3.2.3 and compare the correlation values to a known lower bound in 3.2.4.

3.2.1 Perfect Array Construction

A 1D discrete Kronecker delta (unit impulse), has perfect 1D auto-correlation. Since correlation properties are preserved under projection, we can assign $p+1$ impulses of length p to each row of an array in projection space. The $p + 1$ rows of an array in FRT space correspond exactly to a discrete projection of a 2D array in image space. The inverse FRT of this $(p + 1) \times p$ array produces

a $p \times p$ array with perfect 2D auto-correlation as it will inherit the correlation properties of the projections (as shown in Fig. 3.5).

However, if all of the 1D impulses are positioned at the same location, the back-projected reconstruction trivially produces a 2D impulse. Here we construct arrays by back-projecting delta functions with amplitude p, making $I_{sum} = p$. From equation (3.4), the value of a given array site is given by subtracting one from the number of delta function peaks along its back-projected path. Each delta function peak will contribute p to the back projection, from which we subtract $I_{sum} = p$ and then normalise by dividing by p.

Array entries of $+2$ correspond to positions in the image where back-projected rays from three different projection angles intersect, as $(3p - p)/p = 2$. To minimise the number of 2s in the array, the impulses on each row of the FRT must be translated to minimise the number of triple intersection points between all back-projected rays. This will, in turn, minimise the number of 2s and hence the amount of zeros in the final array. We must also ensure that we are not only producing arrays with perfect auto-correlations, but low cross-correlations also. By randomly shifting delta functions between rows, it is simple to produce a huge family of arrays with perfect auto-correlations, but it is likely that the cross-correlation between family members will be poor when their FRT patterns are too similar.

Controlled shifting of delta functions to ensure low cross correlation of perfect arrays can be achieved by considering the intersection of back projected rays. To this end, shifts of the form $\langle \alpha m^n \rangle_p$ can be applied to each row m with $\alpha, n \in \mathbb{Z}$. We restrict $1 \leq \alpha \leq p - 1$ and $(1 - p)/2 \leq n \leq (p - 1)/2$ to avoid repetition modulo p. For $n < 0$, we compute shift t for row m such that the congruence $mt \equiv 1 \pmod{p}$ is satisfied. The arrays in Radon space can then be written as

$$S_{p,\alpha,n}(t, m) = p\delta_{t,\langle \alpha m^n \rangle_p} \tag{3.6}$$

where $\delta_{i,j}$ is the Kronecker delta. This results in an impulse of height p in FRT column $t = \langle \alpha m^n \rangle_p$ for each row m. The inverse FRT of equation (3.6) will then produce a perfect array in image space $\mathrm{FRT}^{-1}(S_{p,\alpha,n})$. Algorithm 2 outlines the method for producing $p-1$ perfect arrays of size $p \times p$, using shifts of the form $t = \langle \alpha m^n \rangle_p$.

We wish to minimise the number of zeros in the final array for any prime p. This, in turn, will restrict the range of array values. Algorithm 2 constructs arrays limited to the alphabet $\{-1, 0, 1, 2\}$ for $n = -1$ and $n = 2$. In the FRT, each ray (m, t) is back-projected [49] as the line that passes through the image points (x, y), where

$$t = \langle x - my \rangle_p \tag{3.7}$$

for $1 \leq m \leq p - 1$. We want to ensure that the delta impulse from projection m_1 intersects with the delta impulse from projection m_2 at a distinct point (x, y) for each pair m_1, m_2. We let $n = 2$, and substitute $t = \alpha m^2$ into (3.7), which gives

$$x = \alpha m_1^2 + m_1 y = \alpha m_2^2 + m_2 y$$

Algorithm 2 Build $p-1$ seed arrays.

```
1: function BuildSeedSet(p, n)
2:     for α := 1 to p − 1 do
3:         for t := 0 to p − 1 do
4:             for m := 0 to p do
5:                 if t = ⟨αm^n⟩_p then
6:                     S_α(t, m) ← p
7:                 else
8:                     S_α(t, m) ← 0
9:         A_α ← FRT^{−1}(S_α)
       return A
```

which intersects at

$$(x, y) = (-\alpha m_1 m_2, -\alpha(m_1 + m_2)) \,. \tag{3.8}$$

Alternatively we can take $n = -1$ and substitute $t = \alpha m^{-1}$ into (3.7) to produce

$$x = \frac{\alpha}{m_1} + m_1 y = \frac{\alpha}{m_2} + m_2 y$$

which intersects at

$$(x, y) = \left(\frac{\alpha}{m_1} + \frac{\alpha}{m_2}, \frac{\alpha}{m_1 m_2} \right) . \tag{3.9}$$

This shows that we get unique intersections between any pair of rays m_1 and m_2 for all non-degenerate values of $\alpha \pmod p$.

Suppose there exists another ray, m_3. We can substitute $t = \alpha m_3^2$ into (3.7) along with (3.8), or $t = \alpha m_3^{-1}$ and (3.9) to give (3.10) and (3.11) respectively.

$$\alpha m_3^2 + \alpha m_1 m_2 + \alpha m_3 (m_1 + m_2) = 0 \tag{3.10}$$

$$\frac{\alpha}{m_3} - \frac{\alpha}{m_1} - \frac{\alpha}{m_2} + \frac{\alpha}{m_1 m_2} = 0 \tag{3.11}$$

Both (3.10) and (3.11) have solutions $m_3 = m_1$ and $m_3 = m_2$, which contradicts the existence of a direction m_3 distinct from m_1 and m_2. Therefore the values for $1 \le x \le p-1$ and $1 \le y \le p-1$ are from the set $\{-1, 0, 1\}$.

For $x = 0$, $n = 2$ gives $\langle m_1 m_2 \rangle_p = 0$ from (3.8). This equation cannot be satisfied for $1 \le m \le p-1$. Using (3.9), $n = -1$ results in $\langle \langle 1/m_1 \rangle_p + \langle 1/m_1 \rangle_p \rangle_p = 0$, which has $(p-1)/2$ solutions. Along with $R(0,0) = p$, this back-projection therefore has sum $3p$, and results in a $+2$ in the final array.

For $y = 0$, $n = 2$ gives $\langle m_1 + m_2 \rangle_p = 0$ from (3.8). There are $(p-1)/2$ solutions which results in $(p-1)/2$ $+2$ values. For $n = -1$, using (3.9) we have $\langle 1/(m_1 m_2) \rangle_p = 0$, which has no solution. When $x = 0$ and $y = 0$, the final image value for $A(0,0)$ is 1 as we assigned $t = 0$ for both $m = 0$ and

$m = p$. Therefore the arrays constructed through Algorithm 2 are restricted to the alphabet $\{-1, 0, 1, 2\}$ for $n = -1$ and $n = 2$.

Assigning projection m to have cyclic shift $t = m^2$ or $t = m^{-1}$ imposes a strong symmetry on the FRT matrix, as each projection m has a negative counterpart $m' = p - m = -m \pmod{p}$, and then $m^2 = m'^2$, and $1/m = -1/m'$. The "near-orthogonality" of these shift assignments is evident in the "pseudo-Hadamard" matrix constructed from the $p \times p$ matrix product of the shifted delta functions of the FRT, S, with the shifted impulses of its transpose, shown as SS^T, in Fig. 3.6 a) for $t = m^2$ and b) for $t = m^{-1}$. We use the term near-orthogonal to reference the fact that there are few non-zero elements that lie off the diagonal of SS^T.

$t \rightarrow$

m ↓

$$\begin{array}{|ccccccc|} \hline p & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & p & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & p & 0 & 0 \\ 0 & 0 & p & 0 & 0 & 0 & 0 \\ 0 & 0 & p & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & p & 0 & 0 \\ 0 & p & 0 & 0 & 0 & 0 & 0 \\ p & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline \end{array}$$

a) $S_{7,1,2}$

$$\begin{array}{|cccccccc|} \hline p^2 & 0 & 0 & 0 & 0 & 0 & 0 & p^2 \\ 0 & p^2 & 0 & 0 & 0 & 0 & p^2 & 0 \\ 0 & 0 & p^2 & 0 & 0 & p^2 & 0 & 0 \\ 0 & 0 & 0 & p^2 & p^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & p^2 & p^2 & 0 & 0 & 0 \\ 0 & 0 & p^2 & 0 & 0 & p^2 & 0 & 0 \\ 0 & p^2 & 0 & 0 & 0 & 0 & p^2 & 0 \\ p^2 & 0 & 0 & 0 & 0 & 0 & 0 & p^2 \\ \hline \end{array}$$

$S_{7,1,2}S_{7,1,2}^T$

$t \rightarrow$

m ↓

$$\begin{array}{|ccccccc|} \hline p & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & p & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & p & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & p & 0 \\ 0 & 0 & p & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & p & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & p \\ p & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline \end{array}$$

b) $S_{7,1,-1}$

$$\begin{array}{|cccccccc|} \hline p^2 & 0 & 0 & 0 & 0 & 0 & 0 & p^2 \\ 0 & p^2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & p^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & p^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & p^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & p^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & p^2 & 0 \\ p^2 & 0 & 0 & 0 & 0 & 0 & 0 & p^2 \\ \hline \end{array}$$

$S_{7,1,-1}S_{7,1,-1}^T$

Figure 3.6: FRT projection matrices for a 2D array built from 1D phase shifted delta functions, for $p = 7$. a) $n = 2$. b) $n = -1$. The pseudo-orthogonality of these phase shifts is shown on the right via the matrix product of their 2D FRT arrays.

Notice that all non-zero elements lie on either the diagonal or anti-diagonal of SS^T. This means that only rows m and $p - m$ can have the same shifts. The arrays constructed from $S_{7,1,2}$ and $S_{7,1,-1}$ are shown in Fig. 3.7, along with their auto- and cross-correlation.

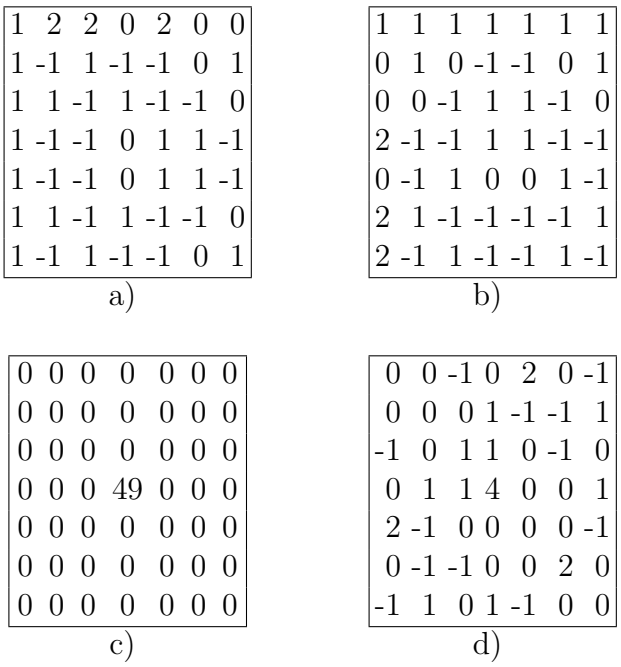


Figure 3.7: 7×7 arrays reconstructed from: a) $\mathrm{FRT}^{-1}(S_{7,1,2})$, b) $\mathrm{FRT}^{-1}(S_{7,1,-1})$. c) Auto-correlation of (a) and (b). d) Cross-correlation between (a) and (b).

3.2.2 Algorithm for Building Array Families

Families of prime $p \times p$ arrays with family size $M = p^2 - 1$ can be built by extending seed sets through affine transformations. Seed sets consisting of $p-1$ perfect arrays are constructed using equation (3.6). We use $p-1$ distinct linear multiples α of the shifts $t = \langle \alpha m^2 \rangle_p$ and $t = \langle \alpha m^{-1} \rangle_p$ to construct two seed sets of arrays, A_1 and A_2.

$$A_1 = \{\mathrm{FRT}^{-1}(S_{p,\alpha,2}) | 1 \leq \alpha \leq p-1\}$$
$$A_2 = \{\mathrm{FRT}^{-1}(S_{p,\alpha,-1}) | 1 \leq \alpha \leq p-1\}$$

Each of these arrays display perfect auto-correlation, and optimally low cross-correlation between arrays within their respective seed sets. It is seen empirically that combining these seed sets results in additional cross-correlation levels, depending on p, that are slightly higher than the crosses within each set (but still has MF $\ll 1$).

Affine transformations are then applied to the seed set A_1 for each of the $p-1$ FRT angles $\theta_{i,j} = \{(1,q) | 1 \leq q \leq p-1\}$ to form the set A_T. For angles where $(q^2+1) \mod p \neq 0$, the affine transform coefficients are set to $[a\ b;\ c\ d] = [j\ -i;\ i\ j]$, which corresponds to an affine rotation. When $q^2+1 \mod p = 0$, the affine rotation matrix has a zero determinant (modulo p), so an affine skew is applied by setting $[a\ b;\ c\ d] = [j\ i;\ i\ j]$.

Finally, the array sets A_1, A_2 and A_T are pooled together to form a set A that will consist of $M = (p-1)+(p-1)+(p-1)(p-1) = p^2-1$ arrays. Each of these arrays will have perfect auto-correlation, and low cross-correlation between all other family members.

Algorithm 3 Build family of $p \times p$ perfect arrays.

```
1: function PerfectArrayFamily(p)
2:     A1 ← BuildSeedSet(p, −1)
3:     A2 ← BuildSeedSet(p, 2)
4:     n ← 1
5:     for α := 1 to p − 1 do
6:         for m := 1 to p − 1 do
7:             if ⟨m² + 1⟩_p = 0 then
8:                 AT_α ← Affine(A1_α, m, 1, 1, m)
9:             else
10:                AT_α ← Affine(A1_α, m, −1, 1, m)
       return A ← {A1, A2, AT}
```

A set of p^2-1 equivalent arrays (up to translation) could be constructed by affine transforming A_2 instead of A_1, or skewing all FRT angles for which $(q^2-1) \mod p \neq 0$ and rotating when $(q^2-1) \mod p = 0$. The choice here is arbitrary, and has no effect on the correlation properties of the final family of arrays. However, applying affine transformations to both seed sets, or using affine rotations and skews for all angles with a non-zero determinant will result in duplicate arrays being produced in the family. This will result in poor cross-correlation between a small fraction family members, where the MF is infinite for identical arrays. These arrays would need to be removed through laborious cross-correlation checking, as done in [66].

A distinguishing property of these arrays are the locations of the +2 values, as they occur least frequently. For seed set arrays, the 2s lie along the first row and column for $n = 2$ and $n = -1$ respectively, as seen in Fig. 3.7a,b. Under the affine transformations applied to extend the family of arrays, the lines of 2s are shifted. They are either rotated or skewed (modulo p) and will lie along a discrete line that wraps around the array. However, visually the +2 elements appear to be randomly positioned under most affine transformations.

The computational complexity of constructing the seed arrays, as shown in algorithm 2, is dominated by taking the inverse FRT of $2(p-1)$ arrays. For each of these arrays, the inverse FRT is of order $\mathcal{O}(p^2 \log p)$ [18], and therefore the overall complexity of building seed arrays is $\mathcal{O}(p^3 \log p)$. To extend the seed arrays, an affine transformation, $\mathcal{O}(p^2)$, is applied to each of the $2(p-1)$ arrays for $(p-1)$ FRT angles. Hence the computational complexity of extending the seed arrays to form a family of p^2-1 arrays is $\mathcal{O}(p^4)$. Algorithm 3, which builds a large family of perfect arrays, consists of building seed sets and then applying affine transformations to them for each FRT angle, therefore has

computational complexity of order $\mathcal{O}(p^4)$. This is a significant improvement on the $\mathcal{O}(p^6)$ complexity in [66], where cross-correlation checking between all pairs of the pooled arrays was required.

3.2.3 Histogram of Perfect Array Values

The structure of the perfect arrays in FRT space fixes the histogram of values in image space. We assign variables h_-, h_0, h_+ and h_2 to the number of grey elements -1, 0, $+1$ and $+2$ respectively in any $p \times p$ array. Using these histogram counts we can write the following set of equations. Given a $p \times p$ array, we know there are p^2 elements in each array.

$$h_- + h_0 + h_+ + h_2 = p^2 \tag{3.12}$$

Each row in FRT space has sum p, therefore the array sum is always p.

$$(-1)h_- + (0)h_0 + (1)h_+ + (2)h_2 = p \tag{3.13}$$

The peak of the auto-correlation is p^2 by the balance theorem [34].

$$(-1)^2h_- + (0)^2h_0 + (+1)^2h_+ + (+2)^2h_2 = p^2 \tag{3.14}$$

Equations (3.12), (3.13) and (3.14) can be solved for h_-, h_0 and h_+ in terms of h_2 and p to give the histogram counts for -1, 0 and $+1$ values.

$$h_- = 1/2(p^2 - p - 2h_2) \tag{3.15}$$
$$h_0 = 3h_2 \tag{3.16}$$
$$h_+ = 1/2(p^2 + p - 6h_2) \tag{3.17}$$

The FRT translates, t, arrange the $(p-1)$ projections to give one triple intersection per m and $p-m$, yielding $(p-1)/2$ elements with value $+2$ in the final array, thus fixing $h_2 = (p-1)/2$. Hence any $p \times p$ array (with p prime) made using the FRT with these values of t, where $t = \langle m^2 \rangle_p$ or $t = \langle m^{-1} \rangle_p$, will have a fixed histogram for its element values, -1, 0, $+1$, $+2$, as $\{(p-1)^2/2$, $3(p-1)/2$, $(p-1)^2/2+1$, $(p-1)/2\}$. Affine transformations leave this histogram unchanged as it is a unique reshuffling of the same array values.

The fixed histogram of element values permits quantification of the merit factors for the periodic cross-correlation of these arrays. The merit factor (MF) is defined as the square of the peak correlation value divided by the sum of all $p^2 - 1$ squared off-peak values. Perfect arrays are, by definition, spectrally flat. All cross-correlations between pairs of spectrally flat arrays are also spectrally flat by the convolution theorem, and hence those cross-correlations are also perfect arrays themselves. The Fourier amplitudes are constant at each frequency, hence they only differ by their phase.

The cross-correlation of two 2D arrays can be equally computed as the cross-correlation of each 1D projection (row) in FRT space, as demonstrated

in Fig. 3.5. The cross-correlation of two delta functions produces another delta function, with its position shifted by the difference of the two original delta functions. Since perfect arrays are constructed from 1D delta functions, the cross-correlation of two arrays in FRT space is simply the difference of the corresponding delta function shifts in each row. Therefore, the cross-correlation of two perfect arrays is also a perfect array. However, it is not guaranteed to consist of the same alphabet.

The lowest possible maximum value of any (normalised) cross-correlation is 2, as it is a perfect array and it was shown in the construction of these arrays that there is a back-projection will intersect at least 3 delta function peaks. The sum of all array terms squared is always p^2, hence μ_0, the lowest possible MF value, is given by $\mu_0 = 2^2/(p^2 - 2^2) = 4/(p^2 - 4)$. The next possible cross levels, μ_v, correspond to the resultant perfect array having a maximum value of $v+2$, or having a back-projection that intersects $v+3$ delta functions. Hence, the MF value is given by

$$\mu_v = \frac{(v+2)^2}{p^2 - (v+2)^2}. \tag{3.18}$$

In practice, it is observed that the bulk of the intra-family cross-correlation values are mainly of type μ_1 and at most μ_4.

3.2.4 Cross-Correlation Lower Bound

In [80], Welch established a lower bound on the maximum cross-correlation value amongst a family of M sequences of length L with ideal auto-correlation. The result aides the design of perfect array families, as it states how small the cross-correlation and auto-correlation can simultaneously be. The maximum cross-correlation between intra-family arrays is defined as

$$C_{\max} = \max_{f \neq g} \left| C_{f,g}(r,s) \right|. \tag{3.19}$$

Welch treated correlations as a form of inner product in a multi-dimensional vector space. The Cauchy–Schwarz inequality is invoked to produce the first Welch bound, which is relevant to our application.

$$C_{\max} \geq \sqrt{\frac{M-1}{ML-1}} \tag{3.20}$$

The families of $M = p^2 - 1$ perfect arrays can be viewed as sequences of length $L = p^2$ to use Welch's lower bound, and yield the following result.

$$C_{\max} \geq \sqrt{\frac{p^2-2}{p^4-p^2-1}} \tag{3.21}$$

This bound assumes the auto-correlation of each array has a unitary peak, $C_{f,f}(0,0) = 1$. Without normalisation, perfect arrays have p^2 auto-correlation

peak, and vp cross correlation peak. Thus, adjusting the cross-correlation MF levels from Sect. 3.2.3 requires vp/p^2 to apply the Welch bound, so we write the cross-correlation values are written as v/p for $v = 2, 3, 4$. Figure 3.8 plots v/p against the Welch bound to show the difference between the cross-correlation of these arrays and the theoretical limit. These families of arrays are above the theoretical lower bound, but steadily approach it for large p.

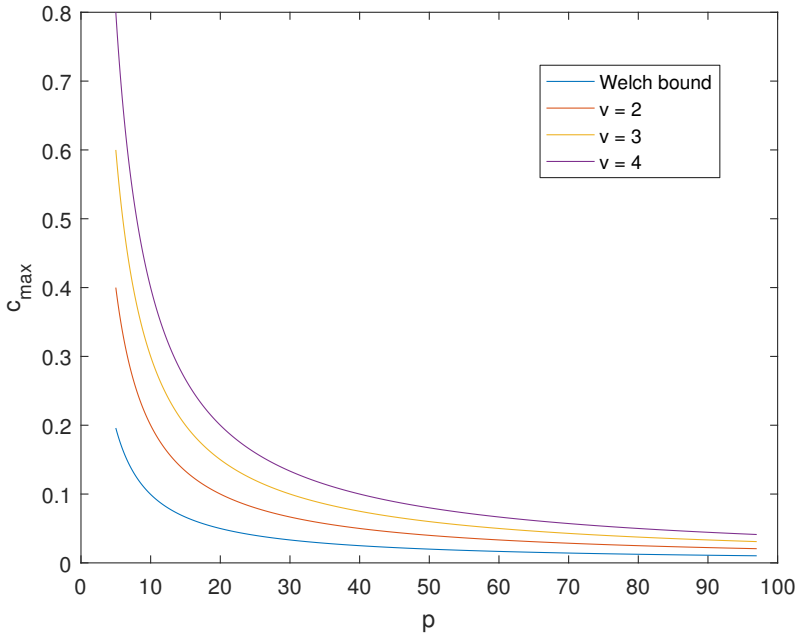


Figure 3.8: Welch bound with $M = p^2 - 1$ and $L = p^2$ plotted against maximum correlation values v/p for $v = 2, 3, 4$ for $p = 5$ to 97.

3.3 Alternative Seed Sets

Algorithm 3 uses shifts of the form $t = m^2$ and $t = m^{-1}$ to construct families of perfect arrays. These shifts were chosen as they minimise the number of triple intersections between back-projected rays, thus minimising the number of zeros in the final array. Other values of n can be used as the exponent for the scaled $t = m^n$ shifts of impulses. $S_{p,\alpha,n}$ can be constructed using any $n \in \mathbb{Z}$, although setting n arbitrarily will not maintain the small alphabet of $\{0, \pm 1, +2\}$ after back-projection. Therefore, we restrict n to cases where all back-projected rays intersect no more than 3 of the shifted delta functions. This ensures the maximum value of the back-projected array is no more than 2. Values of n that satisfy this property change for any given p, as projections wrap cyclically, modulo p. Satisfactory exponents n are found by checking the histogram of any sample array (as affine transformations do not alter the histogram). An array with shifts $t = \langle m^n \rangle_p$ can be back-projected and checked relatively quickly to determine if the resultant array consists of values from the

limited alphabet. If n is found to be a valid exponent, then all shifts $t = \alpha\langle m^n\rangle_p$ are valid.

Despite having the same alphabet as the $n = 2$ and $n = -1$ cases for shifts of $t = m^n$, identically constructed arrays for other values of n are not guaranteed to have the same histogram. As the shifts for $n = 2$ and $n = -1$ were chosen to have a minimal number of zeros, other values of n will produce more $+2$ elements and hence more sparse arrays. Just as the histogram for $n = 2$ is always the same as that of $n = -1$, that of n is the same as for $1 - n$. If we look at (3.2) for exponents n and $n' = 1 - n$, then for

$$\begin{array}{llllll} n=0, & x = 1 + my & \Rightarrow & n' = 1, & x & = m + my \\ n=1, & x = m + my & \Rightarrow & n' = 0, & x & = 1 + my \\ n=2, & x = m^2 + my & \Rightarrow & n' = -1, & mx & = 1 + m^2 y \\ n=3, & x = m^3 + my & \Rightarrow & n' = -2, & m^2 x & = 1 + m^3 y \\ n=4, & x = m^4 + my & \Rightarrow & n' = -3, & m^3 x & = 1 + m^4 y \end{array}$$

We are interested in the number of intersections at (x, y) in a $p \times p$ array. Then, up to a cyclic shift, both sides of these equations will always be the same (mod p), if we let the number of solutions for (x, y) interchange with (y, x). We know that $n = 0$ or 1 will never satisfy the back-projection constraint, and $n = 2$ always produces suitable arrays. Since the translates are symmetric after $n = (p-1)/2$, we only need to look for exponents in the range $3 \le n \le (p-1)/2$.

Figure 3.9 shows an example for $p = 11$, which has allowed shifts for $n = 2, 3$ and 4 (and therefore $n = -1, -2$ and -3). The cross-correlation between $\mathrm{FRT}^{-1}(S_{11,1,2})$ and $\mathrm{FRT}^{-1}(S_{11,1,3})$ is displayed in Fig. 3.9d, which is of type μ_1. It should be noted that the properties for $|n| > 2$ are experimentally observed. Similar results to the $p = 11$ example are also experimentally observed for all tested primes, but this remains to be proved.

The histogram counts for the values $\{-1, 0, +1, +2\}$ are $\{50, 15, 51, 5\}$ and $\{40, 45, 21, 15\}$ for $n = 2$ and 3 respectively. For $p = 11$, $n = 3$ and 4 have the same histograms, and therefore so too will $n = -2$ and -3. From Fig. 3.9, it can be seen that there are fewer 0 and 2 elements for $n = 2$ as predicted. However, the array constructed by $S_{11,1,3}$ is far from sparse, as the majority of values are still non-zero. We have not yet found any array constructed using this method that is dominated by zero values.

These alternate seed sets can also be extended using affine transformations to give an additional $p^2 - 1$ arrays for each $(n, 1 - n)$ pair. These alternate families introduce higher cross-correlation levels to those shown in Sect. 3.3, as their histograms differ. Using $p = 11$ as an example, the family generated by $S_{11,\alpha,2}$ and $S_{11,\alpha,-1}$ has 3 cross levels: $\mu_0 = 0.0342$, $\mu_1 = 0.0804$ and $\mu_2 = 0.1524$. The family from $S_{11,\alpha,4}$ and $S_{11,\alpha,-3}$ has 4 cross levels: μ_0, μ_1, μ_2 and $\mu_3 = 5^2/(p^2 - 5^2) = 0.2604$. Using $S_{11,\alpha,3}$ and $S_{11,\alpha,-2}$ as seed sets, the constructed family also has 4 cross levels: μ_0, μ_1, μ_2 and $\mu_4 = 6^2/(p^2 - 6^2) = 0.4235$. These families can be pooled to produce a larger family of $3(p^2 - 1) = 360$ arrays with perfect auto-correlation and 5 cross-correlation levels. Table

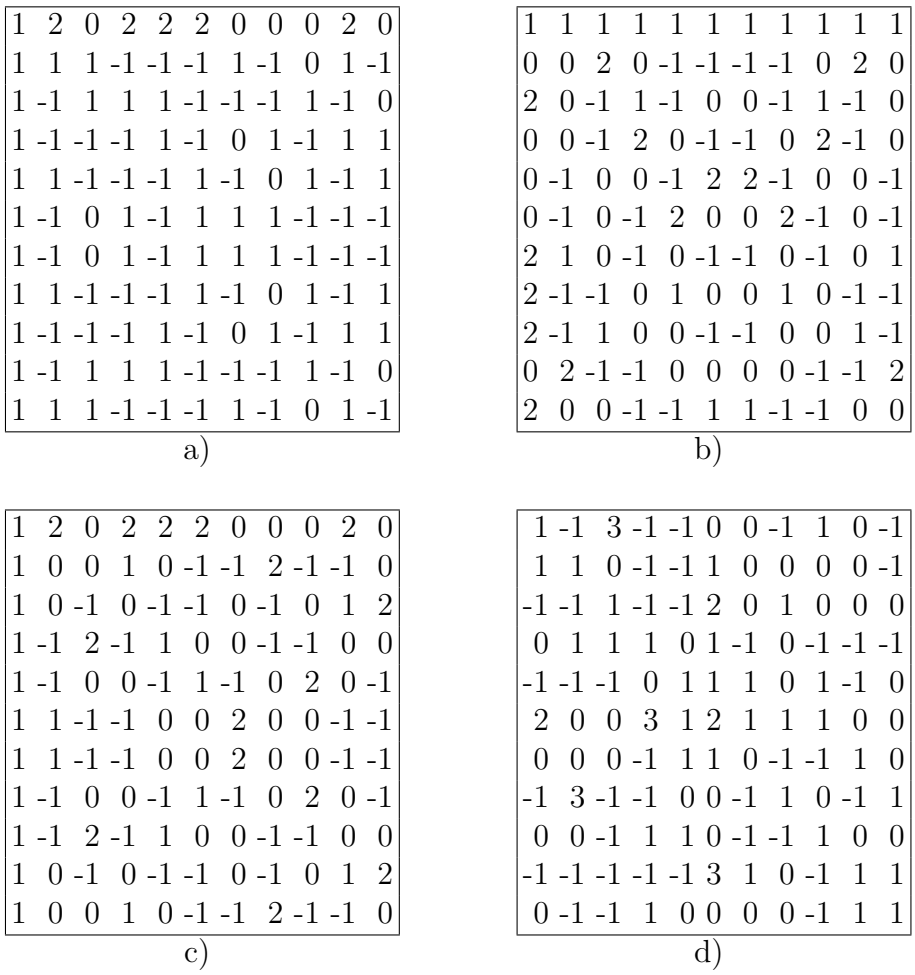


Figure 3.9: 11×11 arrays. a) $\mathrm{FRT}^{-1}(S_{11,1,2})$ b) $\mathrm{FRT}^{-1}(S_{11,1,3})$ c) $\mathrm{FRT}^{-1}(S_{11,1,4})$ d) Cross-correlation between (b) and (c).

3.1 shows the fraction of each cross level in all cross-correlations between family members. This is done for each of the three $(n, 1-n)$ pairs of seed sets, and then the combined family of $3(p^2-1)$ arrays for $p = 11$.

The mean cross-correlation for each of these families is 0.0846, 0.0998, 0.0879 and 0.1049, respectively. Therefore, their average performance is similar, but a marginally wider spread of possible cross-correlation values arises from seed families arising from $(n, 1-n)$ pairs, where $|n| > 2$.

3.3.1 $4N-1$ and $6N-1$ Primes

When a prime p can be written as $4N-1$, where $N \in \mathbb{N}$, the family of arrays produced by $S_{p,\alpha,4}$ along with $S_{p,\alpha,-3}$ has been seen experimentally to always satisfy the back-projection constraint to produce arrays with the alphabet $\{0, \pm 1, +2\}$. Similarly, when p is a $6N-1$ prime, the pair $S_{p,\alpha,3}$ and $S_{p,\alpha,-2}$ also produce a family of arrays with the required restricted alphabet. This was

Table 3.1: Cross-correlation levels and their fractional frequencies in seed sets for $p = 11$.

Seed Sets	μ_0	μ_1	μ_2	μ_3	μ_4
	0.0342	0.0804	0.1524	0.2604	0.4235
$S_{11,\alpha,\{2,-1\}}$	0.0924	0.7899	0.1176	0	0
$S_{11,\alpha,\{3,-2\}}$	0.1765	0.5042	0.3025	0	0.0168
$S_{11,\alpha,\{4,-3\}}$	0.2437	0.5714	0.1345	0.0504	0
Pooled Sets	0.0752	0.5980	0.2934	0.0279	0.0056

seen in the example for $p = 11$, as this is both a $4N - 1$ and $6N - 1$ prime.

Table 3.2: Array histograms for $4N - 1$ and $6N - 1$ primes.

	$n = 2$	$n = 3$	$n = \frac{(p+1)}{4}$	$n = \frac{(p+1)}{4}$
		$n = \frac{(p+1)}{3}$	$n = \frac{(p-3)}{2}$	$n = \frac{(p-3)}{2}$
	for all p	$p = 6N - 1$	$p = 8N - 1$	$p = 8N - 5$
h_-	$\frac{(p-1)^2}{2}$	$\frac{(p^2-1)}{3}$	$\frac{3(p^2-1)}{8}$	$\frac{(3p-1)(p-1)}{8}$
h_0	$\frac{3(p-1)}{2}$	$\frac{(p-1)(p-2)}{2}$	$\frac{3(p-1)(p-3)}{8}$	$\frac{3(p^2-1)}{8}$
h_+	$\frac{(p-1)^2}{2} + 1$	$2p - 1$	$\frac{(p^2+16p-9)}{8}$	$\frac{(p+1)(p+3)}{8}$
h_2	$\frac{(p-1)}{2}$	$\frac{(p-1)(p-2)}{6}$	$\frac{(p-1)(p-3)}{8}$	$\frac{(p^2-1)}{8}$

In these cases, the arrays also have predictable histograms for particular exponents m^n. For $n = 3$ and $n = (p+1)/3$, which requires $p = 6N-1$ so that $p+1$ is always divisible by 3, we find $h_2 = (p-1)(p-2)/6$. For $n = (p+1)/4$ and $n = (p-3)/2$ we have two cases of $p = 4N - 1$ primes, $p = 8N - 1$ and $p = 8N - 5$ primes. Primes of the form $p = 8N - 1$ give $h_2 = (p-1)(p-3)/8$ and $p = 8N - 5$ primes result in $h_2 = (p^2 - 1)/8$. Equations (3.15), (3.16) and (3.17) are used to obtain the histogram counts for h_-, h_0 and h_+, as in Table 3.2.

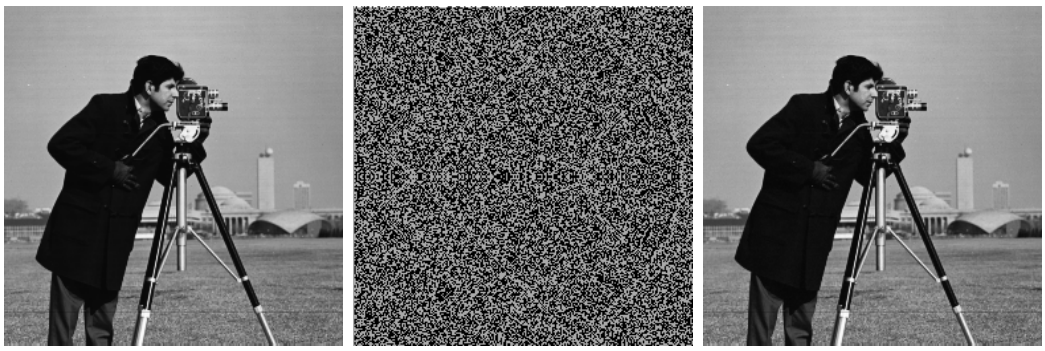
In general, the $n = 3$ case has the largest number of zero elements in the array because there are $(p+1)/3$ triple intersects, being less for $(p+1)/4$ or any higher n. This is always higher than the $n = 2$ case, as there are intersections of three rays occurring inside the array as well as along the top row. There are less than 50% zeros in these arrays for any size p as the maximum fraction of zero elements is $(p-2)(p-1)/(2p^2) < \frac{1}{2}$.

3.4 Perfect Array Applications

In this section, we outline some applications for our perfect arrays. We focus on three broad areas of application: watermarking, encryption and communications. Since this work does not focus on methods for implementing sequences with low correlation, we shall use rather naive methods as a proof of concept.

3.4.1 Watermarking

Watermarking involves embedding markers into media, usually for the purpose of identification. The first examples of digital watermarks were presented in [71, 75]. Highly correlated arrays with a small alphabet are perfect for this application, as they minimally perturb the signal they are embedded into, and are easy to identify. Ideally, watermarks should not be visually identifiable, as to not disrupt consumption. In the case of images, humans are not capable of visually detecting slight changes in the pixel values unless the image is very flat. For most images, this is not an issue. In Fig. 3.10, we have taken a 256×256 image and added a 251×251 perfect array into it. It is nearly impossible to visually detect the change.



a) b) c)

Figure 3.10: a) Original 256×256 cameraman image. b) 251×251 perfect array. c) Cameraman test image with perfect array added.

The watermark can easily be identified using a simple cross-correlation operation. In Fig. 3.11, we have taken the cross-correlation of the 251×251 perfect array, and the embedded image. The graphs represent line-traces through the cross-correlated images. A sharp peak is revealed with a value of 121412. The next highest value is 61995, making the centre of the watermark easily identifiable. For reference, the aperiodic auto-correlation of the perfect array is also shown. Here, the watermark was embedded directly into the image additively. However, this could be embedded through other methods, such as blocks in Fourier space. It is also not necessary to have the array be the same size as the image. Although a larger array will yield a stronger peak.

It is highly favourable to have a large family of watermarks that can each be identified on their own, but bear little resemblance to each other. Each array can then be used to mark its own image, and each array cannot find the embedded watermarks in other images. If a different array in the family was correlated with our embedded image, we would not see the peak in Fig. 3.11b.

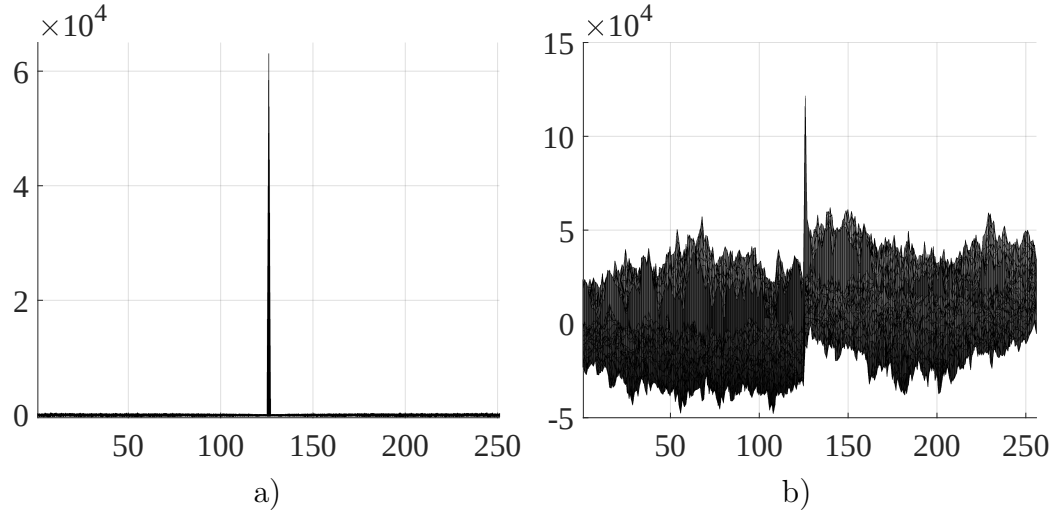


Figure 3.11: a) Aperiodic auto-correlation of perfect array. b) Cross-correlation of perfect array and cameraman image with embedded watermark.

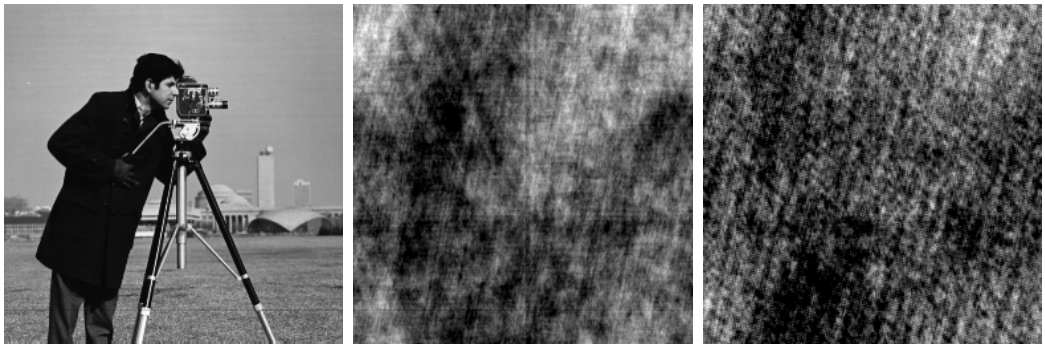
3.4.2 Encryption

Perfect arrays can also be used for the purpose of encryption. These arrays can act as keys used to scramble and then decipher data. Consider again the cameraman image. If we take the cross correlation of the image with a perfect array, we produce an incomprehensible image with useless information (Fig. 3.12b). Applying the inverse operation with the same key yields the original image (i.e. a replica of Fig. 3.12a). However, if another key is applied, a similarly incoherent image is produced, shown in Fig. 3.12c.

The dissimilarity of each array in a family means that every other key will return similarly useless data. Similar to the implementation of watermarks, the process can be carried out in other domains of the image, such as Fourier space. In practice, these arrays may form a part of a larger encryption scheme.

3.4.3 Multiple Access Communication

Consider a wireless communications system with multiple users communicating simultaneously. We need to ensure that all information arrives correctly at its intended destination, and not get mixed up with other users' signals. One way of doing this is to assign communicating users their own frequency band.



a) b) c)

Figure 3.12: a) Original 256×256 cameraman image. b) 251×251 perfect array. c) Cameraman test image with perfect array added.

However, this option is rather limited by the number of possible frequency bands.

Multiplexing is a broad term used to describe any system that allows combined signals to be transferred over the same medium. Many multiplexing technologies use a similar approach, which is to assign each user a unique code, which is then used to modulate the original data. Once this data has been modulated by the code, it can be combined with data from other users for communication. The receiver can then use the code to extract the relevant information from the combined signal.

If this data is packaged and sent simultaneously, the design of the codes can be simple. However, in the case of mobile phones communicating for example, users are not synchronising the time at which they make calls. Therefore, we need a way to identify the which portion of the signal is relevant before decoding. This is where sequences with low correlation are vital. If a sequence with low-off peak correlation is used to modulate the data, a cross correlation can be performed to find the start of the relevant part of the signal. A large number of low-correlation sequences is required, each bearing little similarity, so that the cross-correlation does not mistakenly identify the wrong signal, such as those presented in [37]. The families of arrays presented in this work are highly suited to this application due the large number of possibilities and small alphabet.

3.5 Legendre Arrays

Up to this point, we have considered families of arrays which were constructed by back-projecting delta functions through the FRT. For the original and highest performing case of $n = 2$, the back-projected values depend on whether or not the ray passes through a point with a quadratic shift. Therefore, the arrays that are made up of quadratic residue sequences are various cyclic shifts. In this section, we use this formulation of perfect array families to generalise the

construction to any number of dimensions.

3.5.1 Legendre Sequences

The Legendre symbol, written $\left(\frac{a}{p}\right)$, is a function with values -1, 0, 1, depending on whether a is a quadratic residue of prime number p.

$$\left(\frac{a}{p}\right) = \begin{cases} 0 & \text{if } p|a \\ +1 & \text{if } a \text{ is a quadratic residue} \mod p \\ -1 & \text{if } a \text{ is a quadratic non-residue} \mod p \end{cases} \tag{3.22}$$

These values are periodic, and repeat with period p. This sequence of length p is called the Legendre sequences. The Legendre sequence can alternatively be written as

$$l(k) = k^{(p-1)/2} \mod p \tag{3.23}$$

where we take the quadratic residue of smallest magnitude. Below are two examples of Legendre sequences for $p = 7$ and $p = 11$ respectively. These are the sequences that make up each row of the $n = 2$ perfect array seed sets in Figs. 3.7 and 3.9.

$$l_7 = \{0, 1, 1, -1, 1, -1, -1\}$$

$$l_{11} = \{0, 1, -1, 1, 1, 1, -1, -1, -1, 1, -1\}$$

We now present some properties of Legendre sequences which will be used in constructing perfect arrays. The Legendre sequence has an offset delta function auto-correlation, with peak $p-1$ for zero relative shift and -1 elsewhere. The auto-correlation $c(t)$ is given by

$$\begin{aligned} c(t) &= \sum_{n=0}^{p-1} l(n)l(n+t) \\ &= p\delta(t) - 1 \end{aligned} \tag{3.24}$$

Legendre sequences are comprised of $(p-1)/2$ instances of both $+1$ and -1, and one zero value. Hence, Legendre sequences must sum to zero.

$$\sum_{n=0}^{p-1} l(n) = 0 \tag{3.25}$$

Using this, we can find the cross-correlation of a Legendre sequence with itself shifted by a constant a.

$$c(t) = \sum_{n=0}^{p-1} l(n)(l(n+t) + a) \tag{3.26}$$

$$
\begin{aligned}
&= \sum_{n=0}^{p-1} l(n)l(n+t) + a\sum_{n=0}^{p-1} l(n) \\
&= p\delta(t) - 1
\end{aligned}
$$

Hence the result is the same as the auto-correlation. If we add this constant to both sequences, we get

$$
\begin{aligned}
c(t) &= \sum_{n=0}^{p-1} (l(n+t)+a)(l(n+t)+a) \\
&= \sum_{n=0}^{p-1} (l(n)l(n+t) + al(n) + al(n+t) + a^2) \\
&= p\delta(t) - 1 + pa^2
\end{aligned} \tag{3.27}
$$

The resultant sequence is equal to the auto-correlation shifted by the constant term pa^2. We can now use these properties to investigate how the Legendre sequence can be generalised to higher dimensions.

3.5.2 2D Legendre Arrays

We define the 2D Legendre array as

$$
l(m,n) = (n - m^2)^{(p-1)/2} \mod p + \delta(m) \tag{3.28}
$$

This can also be written in terms of a Legendre sequence.

$$
l(m,n) = l(n - m^2) + \delta(m) \tag{3.29}
$$

Therefore, in 2D, we have arrays where each row is shifted quadratically, and +1 is added to the first row. This gives us the same perfect arrays as constructed by the FRT. Given that we can write these 2D arrays in terms of the 1D sequences, we use the properties given in Section 3.5.1 to show that the 2D auto-correlation is

$$
\begin{aligned}
c(s,t) &= \sum_{m=0}^{p-1}\sum_{n=0}^{p-1} l(m,n)l(m+s,n+t) \\
&= \sum_{m=0}^{p-1}\sum_{n=0}^{p-1} (l(n-m^2)+\delta(m))(l(n+t-(m+s)^2)+\delta(m+s)) \\
&= \sum_{m=0}^{p-1}\Bigg[\sum_{n=0}^{p-1} l(n-m^2)l(n+t-(m+s)^2) + \sum_{n=0}^{p-1}\delta(m)\delta(m+s) \\
&\qquad + \sum_{n=0}^{p-1}\delta(m)l(n+t-(m+s)^2) + \sum_{n=0}^{p-1}\delta(m+s)l(n-m^2)\Bigg]
\end{aligned} \tag{3.30}
$$

$$= \sum_{m=0}^{p-1} \left[p\delta(m)\delta(s) + \sum_{n=0}^{p-1} l(n-m^2)l(n-m^2+t-2ms-s^2) \right]$$
$$= p\delta(s) + \sum_{m=0}^{p-1} (p\delta(s^2+2ms+t) - 1)$$

We now check the following 3 cases: For $s = t = 0$,

$$c(0,0) = p + \sum_{m=0}^{p-1} (p-1) = p^2. \tag{3.31}$$

For $s = 0$, $t \neq 0$,

$$c(0,t) = p + \sum_{m=0}^{p-1} -1 = 0. \tag{3.32}$$

For $s \neq 0$, $t \neq 0$,

$$c(s,t) = \sum_{m=0}^{p-1} (p\delta(s^2+2ms+t) - 1). \tag{3.33}$$

Solving for the discriminant, $4m^2 - 4t = 0$, gives $m^2 = t \pmod p$. Hence there is one solution and (3.33) becomes

$$c(s,t) = p - 1 + \sum_{m^2 \neq t} -1 = (p-1) + (p-1)(-1) = 0. \tag{3.34}$$

Therefore, $l(m,n)$ has ideal auto-correlation.

An interesting property of the Legendre array is that a 1:1 affine rotation always yields a symmetric matrix. The 1:1 affine rotation of a Legendre array is given by

$$\begin{aligned} l(m,n) &= (m+n-(m-n)^2)^{(p-1)/2} \mod p + \delta(m-n) \\ &= (n+m-(n-m)^2)^{(p-1)/2} \mod p + \delta(n-m) \\ &= l(n,m) \end{aligned} \tag{3.35}$$

Hence, for all p, we can generate $p-1$ symmetric Legendre arrays given by each of the seed arrays

$$l(m,n) = (n - \alpha m^2)^{(p-1)/2} \mod p + \delta(m) \tag{3.36}$$

for $1 \leq \alpha \leq p-1$.

3.5.3 Legendre Arrays in Higher Dimensions

We begin this section by showing that we can construct similar Legendre arrays in three dimensions, and then generalise this process to higher dimensions. The 3D Legendre array is given by

$$l(m,n,k) = (n - m^2 - k^2)^{(p-1)/2} \mod p + p^{1/2}\delta(m)\delta(k) \tag{3.37}$$

$$= l(n - m^2 - k^2) + p^{1/2}\delta(m)\delta(k) \tag{3.38}$$

The auto-correlation for the 3D array is

$$\begin{aligned} c(s,t,u) =& \sum_{k=0}^{p-1}\sum_{m=0}^{p-1}\sum_{n=0}^{p-1}\big[(l(n-m^2-k^2) + p^{1/2}\delta(m)\delta(k)) \\ & (l(n+t-(m+s)^2-(k+u)^2) + p^{1/2}\delta(m+s)\delta(k+u))\big] \\ =& \sum_{k=0}^{p-1}\sum_{m=0}^{p-1}\Bigg[\sum_{n=0}^{p-1} l(n-m^2-k^2)l(n+t-(m+s)^2-(k+u)^2) \\ & \sum_{n=0}^{p-1} p\delta(m)\delta(k)\delta(s)\delta(u)\Bigg] \\ =& p^2\delta(s)\delta(u) + \sum_{k=0}^{p-1}\sum_{m=0}^{p-1}\big[p\delta(s^2+u^2-2ms-2ku+t) - 1\big] \end{aligned} \tag{3.39}$$

We now check the following scenarios. For $s = t = u = 0$,

$$\begin{aligned} c(0,0,0) &= p^2 + \sum_{k=0}^{p-1}\sum_{m=0}^{p-1}(p-1) \\ &= p^3. \end{aligned} \tag{3.40}$$

For $s = u = 0$, $t \neq 0$,

$$\begin{aligned} c(0,t,0) &= p^2 + \sum_{k=0}^{p-1}\sum_{m=0}^{p-1}(p\delta(t) - 1) \\ &= p^2 + \sum_{k=0}^{p-1}\sum_{m=0}^{p-1}(-1) \\ &= 0. \end{aligned} \tag{3.41}$$

For $s \neq 0$, $t \neq 0$, $u = 0$, and using equation (3.34),

$$c(s,t,0) = \sum_{k=0}^{p-1}\sum_{m=0}^{p-1}(p\delta(s^2+2ms+t) - 1) = 0. \tag{3.42}$$

The case for $s = 0$, $t \neq 0$, $u \neq 0$ is similar. For $s \neq 0$, $t \neq 0$, $u \neq 0$,

$$c(s,t,u) = \sum_{k=0}^{p-1} \sum_{m=0}^{p-1} \left[p\delta(s^2 + u^2 - 2ms - 2ku + t) - 1 \right] \tag{3.43}$$

The discriminant of $s^2 + u^2 - 2ms - 2ku + t$ in s is

$$m^2 - t + 2ku - u^2 = 0.$$

Then the discriminant of this equation in u is $k^2 + m^2 - t = 0$ or $t = m^2 + k^2$. Thus, we have

$$\begin{aligned} c(s,t,u) &= \sum_{k=0}^{p-1} \left[p - 1 + \sum_{t \neq m^2+k^2} (-1) \right] \\ &= \sum_{k=0}^{p-1} \left[p - 1 + (p-1)(-1) \right] \\ &= 0. \end{aligned} \tag{3.44}$$

Therefore, $l(m,n,k)$ had ideal auto-correlation in 3D.

In 3D, we have quadratic shifts to the Legendre sequence in both the rows and as we move through 3D planes. Also, instead of adding 1 to the first row, we now add $p^{1/2}$. As we demonstrated, this value is needed to balance the auto-correlation. Note that we still only add this value to one row, which is in the first slice in 3D. In subsequent 3D planes, the top row does not need this value added. This is illustrated in Fig. 3.13.

$k = 0$

$\sqrt{5}$	$\sqrt{5}+1$	$\sqrt{5}-1$	$\sqrt{5}-1$	$\sqrt{5}+1$
1	0	1	−1	−1
1	−1	−1	1	0
1	−1	−1	1	0
1	0	1	−1	−1

$k = 1$

1	0	1	-1	-1
-1	1	0	1	-1
0	1	-1	-1	1
0	1	-1	-1	1
-1	1	0	1	-1

$k = 2$

1	-1	-1	1	0
0	1	-1	-1	1
-1	-1	1	0	1
-1	-1	1	0	1
0	1	-1	-1	1

$k = 3$

1	-1	-1	1	0
0	1	-1	-1	1
-1	-1	1	0	1
-1	-1	1	0	1
0	1	-1	-1	1

$k = 4$

1	0	1	-1	-1
-1	1	0	1	-1
0	1	-1	-1	1
0	1	-1	-1	1
-1	1	0	1	-1

Figure 3.13: 3D Legendre array for $p = 5$.

A similar argument can be used to generalise this construction to any number of dimensions. Let $c(s_1, \ldots, s_n)$ be the auto-correlation of an n-dimensional Legendre array. When $s_1 = \cdots = s_n = 0$, then $c(0, \ldots, 0) = p^n$ by definition.

When $s_1, \ldots, s_n$ are all non-zero, we get a similar recursion of quadratic discriminants to solve, as in equation 3.43. All other cases will already be solved in a lower dimension, as with 3.42.

Accordingly, Legendre arrays can be generalised to n-dimensions through cumulative quadratic shifts along each dimension.

$$l(k, m_1, \ldots, m_{n-1}) = (k - \alpha_1 m_1^2 - \cdots - \alpha_{n-1} m_{n-1}^2)^{(p-1)/2} \mod p + p^{(n-2)/2} \delta(m_1, \ldots, m_{n-1}) \quad (3.45)$$

Since $\alpha_1, \ldots, \alpha_{n-1}$ can vary independently, this produces $(p-1)^{n-1}$ nD arrays. In nD, we add $p^{(n-2)/2}$ to the first row of the array to balance the auto-correlation. These initial arrays can again be extended to large families through the use of affine transformations and transpositions to periodically shuffle the values, but maintain the perfect correlation properties. This is achieved by performing 2D affine transformations along planes of the nD arrays. For example, in 3 dimensions, we are able to apply transformations of the following form.

$$\begin{bmatrix} k' \\ m_1' \\ m_2' \end{bmatrix} = \begin{bmatrix} r_1 & \pm 1 & 0 \\ 1 & r_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_2 & 0 & \pm 1 \\ 0 & 1 & 0 \\ 1 & 0 & r_2 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & r_3 & \pm 1 \\ 0 & 1 & r_3 \end{bmatrix} \begin{bmatrix} k \\ m_1 \\ m_2 \end{bmatrix} \quad (3.46)$$

This allows us to construct nD families of arrays with prime side length p at least of order $\mathcal{O}(p^n)$.

3.6 Summary

This chapter presented discrete projection-based methods to construct families of perfect $p \times p$ arrays with a fixed alphabet of $\{0, \pm 1, +2\}$. The FRT was employed to back-project ensembles of shifted Kronecker deltas to reconstruct a 2D array that inherits the ideal auto-correlation properties of the Kronecker delta. Each discrete impulse has translate t for projection angle m of the form $t = m^n$ or m^{1-n} for chosen values of n. The exponent n is selected to restrict the alphabet of array elements, but also to fix the frequency with which each element occurs in each of the M $p \times p$ arrays.

Seed families of these $p \times p$ arrays of size $p-1$ can be constructed for each power n. Each family has perfect auto-correlation and exhibits the optimally low cross-correlation level between all array pairs. Seed sets using translations defined by $n = 2$ and $n = -1$ can be constructed for any prime p.

Affine transformations are then applied to these seed sets, using transform coefficients determined by $p-1$ discrete angles of the FRT. This extends seed sets to a family of $M = p^2 - 1$ arrays. The extended family cross-correlation values span either two or three of the lowest possible levels. The merit factors for these cross-correlations levels are given by $v^2/(p^2 - v^2)$, for $v = 2$, 3 and 4.

Multiple families of size $p^2 - 1$ may be made for a given p, using other values of n, with its $1 - n$ pair. To limit the number of zeros in the arrays, we limit the allowed shifts to those which maintain the restricted alphabet of $\{0, \pm 1, +2\}$. Pooling of these extended families permits even larger families of size $M = k(p^2 - 1)$, where k is the number of extended families that are pooled. As M increases, the range of cross-correlation levels between family member pairs also slowly increases. We monitor the distribution of these cross levels and measure the mean merit factor for each family size M.

For the extended families of size $M = k(p^2-1)$, the span of cross-correlation values can be reduced, for example, from 5 levels to 4 levels. We can do this by once again resorting to checking the crosses between arrays and removing one array of each pair that results in a level 5 cross. Then $p^2 - 1 < M < k(p^2 - 1)$.

This filtering process can be accelerated by forming simple arithmetic differences between each row of the FRT for every pair of arrays and then counting the number of differences greater than a selected threshold. A co-occurrence matrix of the difference values can be used to filter out one of each array pair that were too closely matched. This method is quite robust and much faster than computing the actual cross-correlation values in the spatial domain.

The families of arrays presented in this work sit above the Welch bound for the maximum cross-correlation between sets of arrays with perfect autocorrelation. These grey array families of size $p \times p$ with p prime, span a very large range of M and we can control the changes in the corresponding correlation levels as M increases.

At this juncture, each shift exponent must be tested to determine if it satisfies the back projection constraint of no more than three intersecting rays. It would be useful to have an immediate way of determining the set of shifts that are valid for a given n at each prime p.

This work focused on families of perfect arrays with the limited elements $\{-1, 0, +1, +2\}$. We can equally well use the same delta-function FRT construction to produce perfect arrays with elements -1 to 3, -1 to 4 and -1 to $g_{\max}$. It is sufficient to check the histogram of one example $p \times p$ array produced by FRT deltas placed at t^n to ensure that all of the family members produced by αt^n, and under all subsequent affine rotations and skews, will have the same fixed range of elements, even though the histogram distribution may change for different powers. We can thus plan to further extend the size of our families by pooling perfect arrays that come from different histogram distributions with values greater than 2.

The number of powers that build perfect arrays is also observed to depend strongly on the prime p. Some primes are very prolific in the number of powers that they support for a given alphabet. For example, $p = 47$, 59 and 83 (all $4N - 1$ and $6N - 1$ primes) each have 18 (9 positive and 9 negative) powers where each power produces families of -1 to $+2$ perfect arrays. Prime 83 also provides a total 40 powers to make -1 to $+3$ arrays and 20 powers to make -1 to $+4$ arrays. Prime 101 ($4N + 1$ and $6N - 1$) has just 6 powers for -1

to +2 arrays, but has 34 powers that produce −1 to +4 arrays and 16 powers that make families of −1 to +5 perfect arrays.

Including larger valued elements requires more zeros in each array (each 5 element in an array incurs $5^2 - 1 = 24$ zeros). However, we have not observed any arrays where the total number of zeros occupy more than 50% of the array size. The cross-correlation MF within and between these array families, as experimentally observed, become just slightly larger as the range of array grey levels increases and is still $\ll 1$. Families of perfect arrays with elements ranging from −1 to $g_{\max}$ is the subject of future work.

The construction of perfect arrays can be generalised to n-dimensions through cumulative quadratic shifts of Legendre arrays along each dimension. As there are scaling constants with each shift can vary independently, it is possible to produce $(p-1)^{n-1}$ nD arrays. These initial arrays can again be extended to large families through the use of affine transformations and transpositions. This is achieved by performing 2D affine transformations along planes of the nD arrays. The exact number of possible transformations that can be applied in n-dimensions is not currently known, though we have provided a lower bound. Finding a unique way to extend the number of arrays as much as possible will be the focus of forthcoming research.

CHAPTER 4

Maximal and Boundary Ghosts

Discrete tomography reconstructs an image of an object on a grid from its discrete projections along relatively few directions. When the resulting system of linear equations is under-determined, the reconstructed image is not unique. Ghosts are arrays of signed pixels that have zero sum projections along these directions, they define the image pixel locations that have non-unique solutions. In general, the discrete projection directions are chosen to define a ghost that has minimal impact on the reconstructed image. Here we construct binary boundary ghosts, which only affect a thin string of pixels distant from the object centre. This means that a large portion of the object around its centre can be uniquely reconstructed. We construct these boundary ghosts from maximal primitive ghosts, configurations of 2^N connected binary (± 1) points over N directions. Maximal ghosts obfuscate image reconstruction and find application in secure storage of digital data. Much of the work presented in this chapter is published in [14].

Discrete tomography studies the reconstruction of points on a grid from their discrete line sums, as introduced in Chapter 2. Computerized tomography generally requires many projections to recover accurate reconstructions. However, by using the knowledge that the reconstruction consists of a discrete set, we can determine the values of points that would not otherwise be recoverable. Approximate but accurate discrete reconstructions can be made from a relatively small amount of projection information. For this reason, and due to the discrete nature of digital systems, discrete tomography algorithms have found use in image processing, electron microscopy and data security [19, 40, 54, 74]. Discrete tomography can exactly reconstruct data points from the discrete line

sums given sufficient projection information, if the data is inherently discrete and there is no noise present. Hence one can treat projections as a mathematical basis for representing higher dimensional data. For this reason, discrete tomography finds application beyond tomographic imaging applications.

In what follows we consider the case of 2D discrete projections. Define a $P \times Q$ grid $\mathcal{A} = \{(i, j) \in \mathbb{Z}^2 : 0 \leq i < P, 0 \leq j < Q\}$. A lattice direction $v = (p, q)$ where $p, q \in \mathbb{Z}$ and $\gcd(p, q) = 1$, is used to define a discrete line $jp = iq + b$ across the lattice, where $b \in \mathbb{Z}$. A discrete projection is defined as the set of all line sums along a lattice direction (p, q). By a result of Katz [47], for a finite set of lattice directions $S = \{(p_n, q_n) \mid n = 1, \ldots, N\}$, any $P \times Q$ image $I : \mathcal{A} \to \mathbb{R}$ can be reconstructed in the absence of noise if and only if

$$P \leq \sum_{n=1}^{N} |p_n| \quad \text{or} \quad Q \leq \sum_{n=1}^{N} |q_n|. \tag{4.1}$$

In this case, the image is uniquely determined by the discrete projections and can be reconstructed efficiently [27, 51, 52]. For a more detailed discussion, refer back to Section 2.4.

We call a set of directions valid if $\sum_{n=1}^{N} |p_n| < P$ and $\sum_{n=1}^{N} |q_n| < Q$. In this case the system of linear equations defined by the discrete projections is under-determined. Whilst parts of the image may be exactly reconstructed from a valid projection set, a non-empty subset of the image points will have indeterminate values. These indeterminate values are characterised by ghosts, also known as switching circuits, phantoms, or bad configurations.

A ghost $G : \mathcal{A} \to \mathbb{R}$ is a function which has zero sum projections for all parallel rays along every direction in S. By the ghost domain of G we mean the set of points of $\mathcal{A}$ where G assumes a nonzero value. They can be generated and characterized algebraically through products of polynomials using the framework established in [44]. For example, the following matrix represents a ghost G for the directions $S = \{(1, 0), (0, 1), (2, 1), (1, 2)\}$.

$$\begin{bmatrix} 0 & 0 & 0 & -1 & +1 \\ 0 & +1 & -1 & +1 & -1 \\ 0 & -1 & +2 & -1 & 0 \\ -1 & +1 & -1 & +1 & 0 \\ +1 & -1 & 0 & 0 & 0 \end{bmatrix}$$

By Katz's result, the (rectangular) size of a ghost domain for a set of lattice directions $S = \{(p_n, q_n) \mid n = 1, \ldots, N\}$ must be at least $1 + \sum_{n=1}^{N} |p_n|$ by $1 + \sum_{n=1}^{N} |q_n|$. A ghost that reaches these lower bounds is referred to as a primitive ghost (Fig. 4.1a). The Katz criterion (4.1) can be interpreted geometrically via primitive ghosts. If a primitive ghost can fit within a given grid, then the image values on that grid cannot be reconstructed uniquely. Apart from a multiplicative factor and a shift the primitive ghost is unique. Conversely, every ghost is a linear combination of shifts of a primitive ghost (Fig. 4.1b).

The image values of points of the grid $\mathcal{A}$ which are not in the domain of any primitive ghost are uniquely determined by the discrete projections. In the matrix of a ghost, the position of the uniquely determined image values have a ghost value of zero. This follows from the theory presented in [44].

$$\begin{bmatrix} 0 & -1 & +1 & 0 \\ +1 & -1 & +1 & -1 \\ 0 & -1 & +1 & 0 \end{bmatrix} \qquad \begin{bmatrix} 0 & -1 & +1 & 0 \\ 1 & -2 & +2 & -1 \\ 1 & -2 & +2 & -1 \\ 0 & -1 & +1 & 0 \end{bmatrix}$$

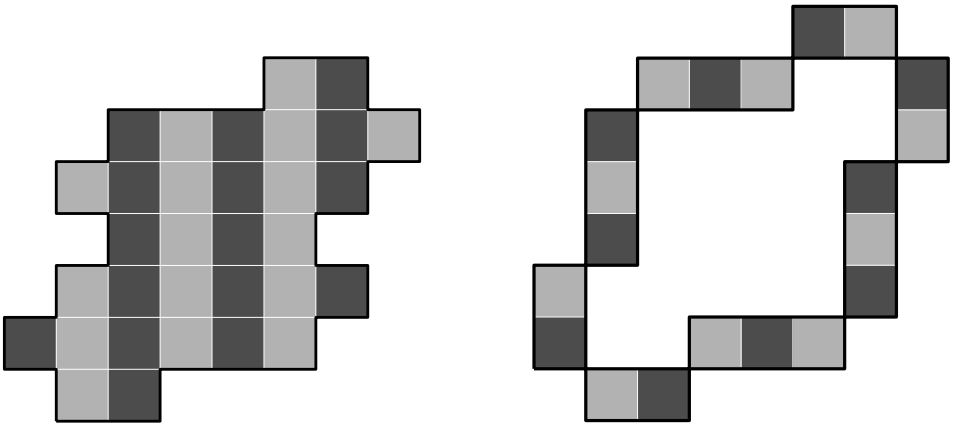
a) b)

Figure 4.1: Ghosts for the directions $S = \{(1,0), (1,1), (1,-1)\}$. a) Primitive ghost. b) Non-primitive ghost, constructed as a linear combination of shifts of the primitive ghost in (a).

For many discrete tomography applications, the aim is to minimise the number of ghost points [76]. For wrapped 2D periodic projections on arrays, minimal ghosts along N co-planar directions can be constructed using N positive and N negative points for any prime-sized array [67]. However, throughout this chapter, we consider the more common situation of unwrapped (aperiodic) projections. In this case, minimal ghosts with $2N$ points only occur for $N = 1, 2, 3, 4$ and 6 [2, 31]. For other N, more than $2N$ ghost points are required and ghosts have been constructed with close to $2N$ points [1, 9, 67]. A primary aim of this chapter is to construct a primitive ghost with a large empty interior so that the central part of the image can be reconstructed.

For some applications, such as watermarking, encryption and secure distributed data storage, it can be favourable that the number of ghost points is large [4, 5, 41]. For example, this can be useful if a group has shared information, but no proper subset of the group should have access to the information. In such a case directions are taken such that there is enough information for exact reconstruction, but that every proper subset of directions is valid. The goal is that if any single server is compromised or any participant is missing, minimal information can be recovered [54].

By construction, a ghost for N directions can have at most 2^N points. We call a ghost maximal if it is primitive and has exactly 2^N points. An example of a maximal ghost for the projection set with zero projections for the set $S = \{(1,0), (1,1), (-1,1), (-3,-1), (-1,-3)\}$ is shown in Fig. 4.2a. If a ghost is maximal, then all ghost values of points in the ghost domain are $+1$ or -1. The notion of a maximal ghost was presented in [65], but a general method for constructing them was not known at the time. In this chapter we present a recursive sequence of directions to create maximal ghosts. Since reconstruction is not unique at ghost points, the amount of information which can be reconstructed when a server or participant is missing is minimal. We derive several properties of maximal ghosts. Geometrically, these ghosts resemble fractals and form tilings that are closely related to rep-tiles [36].



a) b)

Figure 4.2: Ghosts with zero projections for the set $S = \{(1,0), (1,1), (-1,1), (-3,-1), (-1,-3)\}$. Dark and light squares indicate -1 and $+1$ respectively. a) Maximal ghost. b) Boundary ghost, constructed by adding direction $(0,1)$ to S.

We then exploit the structure of maximal ghosts to construct boundary ghosts, the domain of which consists only of a cycle of 8-connected points (see Fig. 4.2). This cycle is closely related to the boundary of the maximal ghosts. Boundary ghosts have a large central region where image values can be uniquely reconstructed. The sequence used to construct these shapes has a possible bifurcation in the choice of lattice directions at each step, therefore we can construct families of such ghosts with a variety of shapes and array sizes. We construct boundary ghosts which fit into a square with side lengths no greater than $3 \cdot 2^{N/2}$ (equations (4.23) and (4.24)). In the remaining sections of this work all ghosts are primitive, and the term ghost will specifically refer to primitive ghosts.

4.1 Constructing Ghosts

A ghost is defined by having zero sum projections across all given directions $S = \{(p_n, q_n) \mid n = 1, \ldots, N\}$. In general, there are many ways in which one can construct a ghost array, including using periodic projections [67] or U-polygons [29]. However, in this work we consider only ghosts of minimal size, or primitive ghosts, as these are the ghosts that arise as reconstruction errors. There are two primary methods for constructing primitive ghosts, which both yield the same end result.

4.1.1 Ghosts via Convolution

An elementary ghost for a discrete angle (p, q), denoted $g_{p,q}$, is a pair of signed points separated by p columns and q rows. This is defined as

$$g_{p,q}(i,j) = \begin{cases} +1 & \text{if } (i,j) = (0,0) \\ -1 & \text{if } (i,j) = (p,q) \\ 0 & \text{elsewhere.} \end{cases} \tag{4.2}$$

A ghost G, over N directions can then be constructed through $N-1$ discrete convolutions of N elementary ghosts.

$$G = g_{p_1,q_1} \otimes \ \ldots \ \otimes g_{p_N,q_N} \tag{4.3}$$

Geometrically, this corresponds to starting with a pair of oppositely signed points that define a ghost in the first chosen direction, which must sum to zero along that direction by construction. Then a positive and negative version of this ghost are separated by a distance defined by the second direction. Ghost values that lie at the same location are summed together. Each ghost sums to zero along the first direction as before, and the composite sums to zero along the second direction as each point in the positive ghost cancels with its negative translated point in the second ghost. The result for the composite 2 direction ghost is then dilated in the third direction, and so on, for N directions. Figure 4.3 shows this process to make the 4 direction ghost with directions $\{(1,0),(0,1),(1,1),(1,-1)\}$.

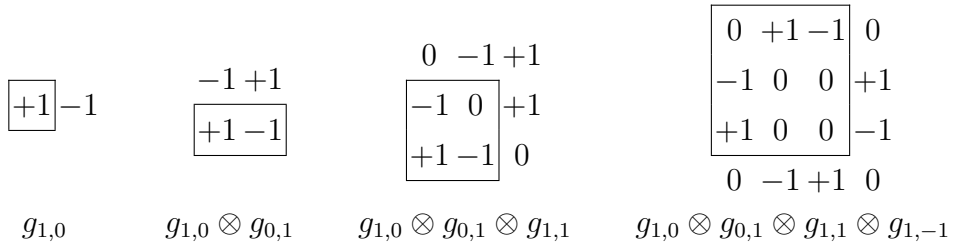


Figure 4.3: Construction of a primitive ghost by convolution. Left to right: the 1D ghost with direction (1,0), translated by direction (0,1), then (1,1) followed by (1,−1) to form the $N = 4$ primitive ghost with 8 non-zero elements. Boxes show the location of the previous ghost before sign reversal and shift by (p, q).

The bounding polygon of the ghost, given by the convex hull of ghost points, is then comprised of $2N$ vectors of S and $-S$. The convex hull can be found through convolution of the lines for the vectors in S. Therefore the convex hull of all primitive ghosts built through convolutions of elementary ghosts are 180° symmetric. The convex hull of ghosts will be explored further in this chapter.

4.1.2 Ghost Polynomials

Ghosts can also be generated algebraically through products of polynomials using the formalism in [44]. For a valid set of d directions, let

$$F_S^{(d)}(x,y) = \prod_{i=1}^{d} f_{(p_i,q_i)}(x,y) \tag{4.4}$$

where

$$f_{(p_i,q_i)}(x,y) = \begin{cases} x^{p_i}y^{q_i} - 1 & \text{if } p_i \neq 0,\ q_i > 0 \\ x^{p_i} - y^{-q_i} & \text{if } p_i \neq 0,\ q_i < 0 \\ x - 1 & \text{if } p_i = 1,\ q_i = 0 \\ y - 1 & \text{if } p_i = 0,\ q_i = 1 \end{cases} \tag{4.5}$$

Using this polynomial, each ghost element is given by $G(k,l) = g_{k,l}$ where $g_{k,l}$ is the coefficient of $x^k y^l$ in $F_S^{(d)}(x,y)$ [9]. Demonstrating the same example, where $S = \{(0,1),(1,0),(1,2),(2,1)\}$ for the algebraic approach gives

$$\begin{aligned} F_S^{(4)}(x,y) &= (x-1)(y-1)(xy^2-1)(x^2y-1) \\ &= 1 - x - y + xy - x^2y + x^3y - xy^2 + 2x^2y^2 - x^3y^2 \\ &\quad + xy^3 - x^2y^3 + x^3y^3 - x^4y^3 - x^3y^4 + x^4y^4. \end{aligned}$$

Taking the coefficients of $F_S^{(4)}(x,y)$, we produce the ghost in Fig. 4.4. For reference, the polynomial is shown alongside with the relevant zero coefficients.

$$\begin{bmatrix} 0 & 0 & 0 & -1 & +1 \\ 0 & +1 & -1 & +1 & -1 \\ 0 & -1 & +2 & -1 & 0 \\ -1 & +1 & -1 & +1 & 0 \\ +1 & -1 & 0 & 0 & 0 \end{bmatrix} \quad \begin{array}{l} +0x^0y^4 +0x^1y^4 +0x^2y^4 -1x^3y^4 +1x^4y^4 \\ +0x^0y^3 +1x^1y^3 -1x^2y^3 +1x^3y^3 -1x^4y^3 \\ +0x^0y^2 -1x^1y^2 +2x^2y^2 -1x^3y^2 +0x^4y^2 \\ -1x^0y^1 +1x^1y^1 -1x^2y^1 +1x^3y^1 +0x^4y^1 \\ +1x^0y^0 -1x^1y^0 +0x^2y^0 +0x^3y^0 +0x^4y^0 \end{array}$$

Figure 4.4: Ghost for directions $\{(1,0),(0,1),(2,1),(1,2)\}$, and its polynomial representation.

4.2 Maximal Primitive Ghosts

In this section, we present a tree of recursive sequences of lattice directions to define a maximal ghost which is 4-connected and has a small boundary. We define the next lattice direction $v_{n+1} = (p_{n+1}, q_{n+1})$ recursively as

$$v_{n+1} = v_n + 2\epsilon_{n+1}v_{n-1}, \quad \epsilon_{n+1} \in \{-1, 1\} \tag{4.6}$$

for $n = 2, 3, \ldots$ with $v_0 = (1, 0)$, $v_1 = (1, 1)$ and the set of ghost points by

$$T_{n+1} = T_n \cup (T_n + v_n) \tag{4.7}$$

for $n = 0, 1, \ldots$ with $T_0 = (0, 0)$. Here $T + x$ is defined as $\{t + x : t \in T\}$. We give T_0 ghost value 1 and the point $t + v_n$ in (4.7) the opposite ghost value of the corresponding point $t \in T$. Further we define for $n = 1, 2, \ldots$

$$U_n = T_n + \{(x, y) \in \mathbb{R}^2 : 0 \leq x \leq 1, 0 \leq y \leq 1\}. \tag{4.8}$$

In the figures throughout this work we use the pixel set U_n to visualise the ghost domain T_n. There is an obvious correspondence between them. This recursion defines directions such that the shifted set $U_n + v_n$ is a shape that fits together with U_n like jigsaw pieces, remaining connected without gaps or any overlap as shown in Fig. 4.5. When $\epsilon_n = -1$ for all n, these directions produce an outward spiral. For $\epsilon_n = +1$ for all n, v_{n+1} will continue in approximately the same direction as v_n.

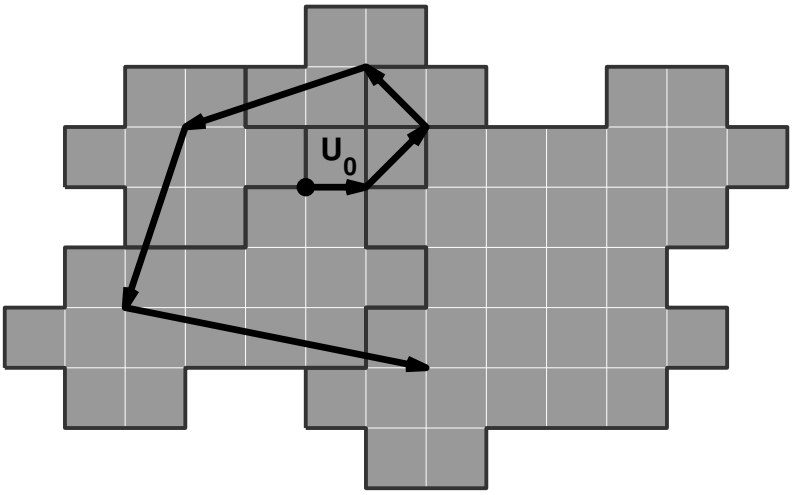


Figure 4.5: Construction of ghost U_6 with $\epsilon_n = -1$ for all n, which has lattice directions $\{(1, 0), (1, 1), (-1, 1), (-3, -1), (-1, -3), (5, -1)\}$. Starting from the unit square U_0, each tile is obtained from the union of the previous tile and its translation by the next lattice direction. The boundaries of $U_0, U_1, U_2, \ldots, U_6$ are outlined.

Since there is no overlap in the geometric construction, the number of ghost points doubles at each step. This results in 2^N ghost points for N directions where each point of T_n has value 1 or -1. In the following sections, we will prove properties of maximal ghosts and their domains.

As we shall show in the next section, the ghosts are obtained by giving value 1 to points $(x, y) \in T_n$ with x even and value -1 to those with x odd. This pattern of signs can naturally be achieved through convolutions of oppositely signed points separated by the directions defined by (4.6). The primitive ghost with 2 directions, $v_0 = (1, 0)$, $v_1 = (1, 1)$ has value 1 for even x and -1 for odd x. This pattern is maintained through all later convolutions independent of the choice of ϵ_n. This follows by induction on n.

4.2.1 Lattice Tilings

We show here that every set of ghost points T_n forms a lattice tiling. We then prove that the set T_n consists of 2^N distinct points. We start with some observations. Write $v_n = (p_n, q_n)$ for $n = 0, 1, \ldots$ where v_n satisfies (4.6). By induction on n we check that, for $n = 1, 2, \ldots,$

$$p_n \text{ and } q_n \text{ are odd, and if } \epsilon_n \text{ is constant, } q_n = p_{n-1}. \tag{4.9}$$

It follows, by the definition of ghost values, that the ghost value of $(x, y) \in T_n$ is 1 if x is even and -1 if x is odd. Thus it suffices to know the ghost domains T_n of the maximal ghosts we construct.

Lemma 1. *We have* $|p_{n-1}q_n - p_nq_{n-1}| = 2^{n-1}$ *and* $\gcd(p_n, q_n) = 1$ *for* $n = 1, 2, \ldots.$

Proof. We have $p_0q_1 - p_1q_0 = 1 = 2^0$ and $\gcd(p_1, q_1) = 1$. For $\epsilon_n = -1$,

$$\begin{aligned} p_{n-1}q_n - p_nq_{n-1} &= p_{n-1}(q_{n-1} - 2q_{n-2}) - (p_{n-1} - 2p_{n-2})q_{n-1} \\ &= 2(p_{n-2}q_{n-1} - p_{n-1}q_{n-2}). \end{aligned}$$

The case for $\epsilon_n = +1$ is similar. The former statement follows by induction.

Since $\gcd(p_n, q_n)$ divides $p_{n-1}q_n - p_nq_{n-1}$, it must be a power of 2. However, by (4.9) p_n is odd. Therefore $\gcd(p_n, q_n) = 1$ for all n. □

For integers r, s, not both zero, we define $\mathbb{Z}(r, s) = \{(mr, ms) : m \in \mathbb{Z}\}$. A lattice Λ in $\mathbb{Z}^2$ is a set $\mathbb{Z}(a, b) + \mathbb{Z}(c, d)$ where $a, b, c, d \in \mathbb{Z}$ are such that (a, b) and (c, d) are linearly independent over $\mathbb{Z}$. A set $A \subset \mathbb{Z}^2$ is a tile for the lattice Λ if every $(x, y) \in \mathbb{Z}^2$ can be written uniquely as $(p, q) + r(a, b) + s(c, d)$ with $(p, q) \in A$ and $r, s \in \mathbb{Z}$. For this, we adopt the shorthand notation $\mathbb{Z}^2 = \Lambda + A$.

Theorem 2. *The set* T_n *forms a tile for the lattice* $\Lambda_n := \mathbb{Z}(v_{n-1} + 2v_{n-2}) + \mathbb{Z}(v_{n-1} - 2v_{n-2})$ *for* $n = 2, 3, \ldots.$

Proof. By induction on n. We have $T_2 = \{(0, 0), (1, 0), (1, 1), (2, 1)\}$ and this forms a tile for the lattice $\Lambda_2 = \mathbb{Z}(-1, 1) + \mathbb{Z}(-3, -1)$. This follows from

$$\begin{aligned} \mathbb{Z}^2 &= \mathbb{Z}(1, 1) + \mathbb{Z}(1, 0) \\ &= \mathbb{Z}(1, 1) + \mathbb{Z}(2, 0) + \{(0, 0), (1, 0)\} \\ &= \mathbb{Z}(3, 1) + \mathbb{Z}(1, 1) + T_1 \\ &= \mathbb{Z}(-3, -1) + \mathbb{Z}(2, 2) + \{(0, 0), (1, 1)\} + T_1 \\ &= \mathbb{Z}(-3, -1) + \mathbb{Z}(-1, 1) + T_2. \end{aligned}$$

Suppose the statement is true for n. Then T_n, the set of all the sums of subsets of $\{(p_0, q_0), \ldots, (p_{n-1}, q_{n-1})\}$, forms a tile for $\Lambda_n = \mathbb{Z}(v_{n-1} + 2v_{n-2}) + \mathbb{Z}(v_{n-1} - 2v_{n-2})$. Taking $\epsilon_n = -1$,

$$\mathbb{Z}^2 = \Lambda_n + T_n$$

$$
\begin{aligned}
&= \mathbb{Z}(v_{n-1} + 2v_{n-2}) + \mathbb{Z}(v_{n-1} - 2v_{n-2}) + T_n \\
&= \mathbb{Z}(2v_{n-1} - v_n) + 2\mathbb{Z}v_n + \{(0,0), v_n\} + T_n \\
&= \mathbb{Z}(v_n - 2v_{n-1}) + \mathbb{Z}(v_n + 2v_{n-1}) + T_{n+1} \\
&= \Lambda_{n+1} + T_{n+1}.
\end{aligned}
$$

The proof for $\epsilon_n = +1$ is similar. Thus $T_{n+1} = T_n + (T_n + v_n)$ forms a tile for the lattice $\Lambda_{n+1} := \mathbb{Z}(v_n + 2v_{n-1}) + \mathbb{Z}(v_n - 2v_{n-1})$. □

Theorem 3. *For $n = 0, 1, 2, \ldots$, the set T_n consists of 2^n distinct points.*

Proof. By induction. The theorem holds for $n = 0, 1$. By Theorem 2, the lattice determinant of Λ_n is the absolute value of

$$
\begin{vmatrix} p_{n-1} + 2p_{n-2} & q_{n-1} + 2q_{n-2} \\ p_{n-1} - 2p_{n-2} & q_{n-1} - 2q_{n-2} \end{vmatrix} = \pm \begin{vmatrix} p_n & q_n \\ 2p_{n-1} & 2q_{n-1} \end{vmatrix} = \pm \begin{vmatrix} p_n & q_n \\ p_{n+1} & q_{n+1} \end{vmatrix}
$$

which equals 2^n by Lemma 1. So the number of points doubles at every step. Therefore all 2^n combinations of T_n are distinct. □

The next theorem establishes our claim in the previous section that T_n with the given ghost values presents a ghost, which means that all the line sums for each direction in S are equal to zero.

Theorem 4. *For the set of directions $S = \{v_h = (p_h, q_h)\}$ $(h = 1, \ldots, n)$ the ghost domain is given by T_n.*

Proof. By induction. The statement is true for $n = 1$. Suppose it is true for T_n. Then the line sums in the directions v_h for $h = 1, \ldots, n$ of $T_{n+1} = T_n \cup (T_n + v_n)$ are 0. For the new direction (p_{n+1}, q_{n+1}) observe that for each point $(x, y) \in T_n$ a point $(x + p_n, y + q_n) \in T_n + v_n$ is added. Since p_n is odd, their point values have opposite signs and therefore sum to 0. Thus the line sums in the direction v_{n+1} are also 0. □

Remark. In the transition from U_n to U_{n+1}, the size of the tile is augmented by $|p_n|$ in the x-direction and by $|q_n|$ in the y-direction. By induction we see that the size of the minimal rectangular block covering U_n is therefore $1 + \sum_{j=1}^{n-1} |p_j|$ by $1 + \sum_{j=1}^{n-1} |q_j|$. (The addition of 1 is due to the size of U_0.) In 1977 Katz proved that this is the minimal size for an object that is not uniquely reconstructable from the discrete projections [47]. It follows that the constructed ghosts are indeed primitive ghosts.

4.2.2 Maximal Ghost Tile Connectivity

Here we show that every neighbour of a point in T_n belongs to T_{n+5}. We define as neighbours of (x, y) the four points $(x + 1, y)$, $(x, y + 1)$, $(x - 1, y)$, $(x, y - 1)$, to identify points which are 4-connected with (x, y). It will follow that our maximal ghosts are 4-connected with no internal holes. We begin by examining the neighbouring tiles of T_n, as shown in Fig. 4.6.

Theorem 5. *For $n = 2, 3, \ldots$ the set T_n is surrounded (4-connected) by the six sets*

$$T_n \pm 2v_{n-1},\ T_n \pm v_{n-1} \pm 2v_{n-2}.$$

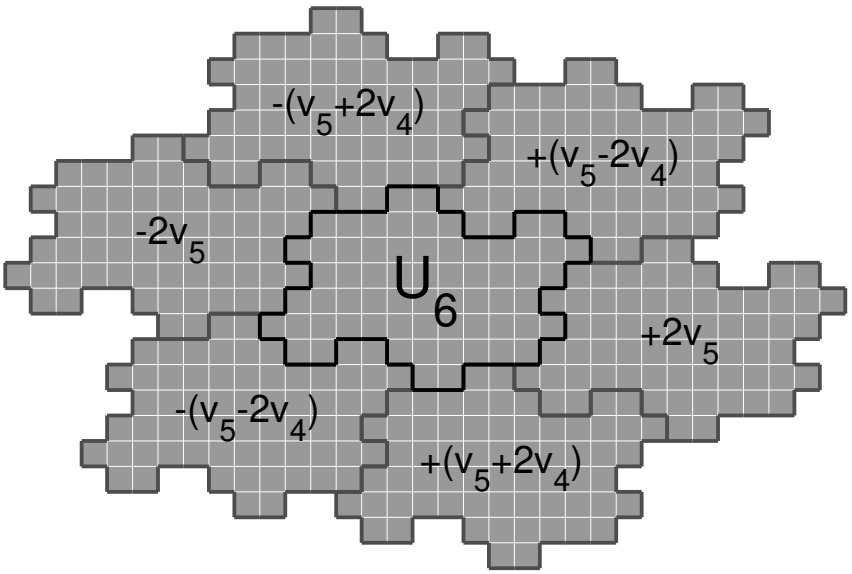


Figure 4.6: The set U_6 and its neighbouring tiles if $\epsilon_n = -1$ for all n. The centred tile U_6 is indeed surrounded by $U_6 \pm 2v_5, U_6 \pm (v_5 - 2v_4) = U_6 \pm v_6, U_6 \pm (v_5 + 2v_4) = U_6 \pm v_7$.

Proof. By induction on n. We have $T_2 = \{(0,0),\ (1,0),\ (1,1),\ (2,1)\}$ and this is surrounded by $T_2 + 2v_1 = \{(2,2),\ (3,2),\ (3,3),\ (4,3)\}$, $T_2 - 2v_1 = \{(-2,-2),\ (-1,-2),\ (-1,-1),\ (0,-1)\}$, $T_2 + v_1 - 2v_0 = \{(-1,1),\ (0,1),\ (0,2),\ (1,2)\}$, $T_2 - v_1 + 2v_0 = \{(1,-1),\ (2,-1),\ (2,0),\ (3,0)\}$, $T_2 + v_1 + 2v_0 = \{(3,1),\ (4,1),\ (4,2),\ (5,2)\}$, $T_2 - v_1 - 2v_0 = \{(-3,-1),\ (-2,-1),\ (-2,0),\ (-1,0)\}$. This can be verified geometrically, and thus the statement is true for $n = 2$.

Now suppose T_n is surrounded by $T_n \pm 2v_{n-1},\ T_n \pm v_{n-1} \pm 2v_{n-2}$. Then, by the lattice structure and (4.6), $T_n + v_n$ is surrounded by $T_n + v_n \pm 2v_{n-1}$ and $T_n + v_n \pm v_{n-1} \pm (v_n - v_{n-1})$, producing the six neighbours $T_n + v_n + 2v_{n-1}$, $T_n + v_n - 2v_{n-1}$, $T_n + 2v_n, T_n + 2v_{n-1}, T_n + 2v_n - 2v_{n-1}$ and T_n. By definition T_{n+1} consists of the tiles T_n and $T_n + v_n$. It is surrounded by

$$\begin{aligned}
T_n \pm 2v_{n-1} &= T_n + v_n - v_n \pm 2v_{n-1} \subset T_{n+1} - (v_n \pm 2v_{n-1}),\\
T_n + v_{n-1} \pm 2v_{n-2} &= T_n + v_{n-1} \pm (v_n - v_{n-1}) \begin{cases} \subset T_{n+1} - (v_n - 2v_{n-1}) \\ = T_n + v_n \subset T_{n+1} \end{cases},\\
T_n - v_{n-1} \pm 2v_{n-2} &= T_n - v_{n-1} \pm (v_n - v_{n-1}) \begin{cases} \subset T_{n+1} + (v_n - 2v_{n-1}) \\ \subset T_{n+1} - 2v_n \end{cases},\\
T_n + 2v_n &\subset T_{n+1} + 2v_n,
\end{aligned}$$

$$
\begin{aligned}
T_n + v_n \pm 2v_{n-1} &\subset T_{n+1} + (v_n \pm 2v_{n-1}),\\
T_n + 2v_{n-1} &= T_n + v_n - v_n + 2v_{n-1} \subset T_{n+1} - (v_n - 2v_{n-1}),\\
T_n + 2v_n - 2v_{n-1} &= T_n + v_n + v_n - 2v_{n-1} \subset T_{n+1} + (v_n - 2v_{n-1}).
\end{aligned}
$$

where the top row of each curly brace refers to the + case of the ± and the bottom row is that of the − case. There are six distinct point sets on the right-hand side that are neighbours of T_{n+1}, because the left-hand sides have this property. Thus T_{n+1} is surrounded by $T_{n+1} \pm 2v_n,\ T_{n+1} \pm v_n \pm 2v_{n-1}$. □

Corollary 6. *For $n = 2, 3, \ldots$ the tile U_n is 4-connected.*

Corollary 7. *For $n = 2, 3, \ldots$ the tile T_n and all its neighbours are contained in T_{n+5} when $\epsilon_n = \epsilon_{n+1} = ... = \epsilon_{n+4} = -1$.*

Proof. In this case $v_n = v_{n-1} - 2v_{n-2}, v_{n+1} = v_n - 2v_{n-1} = -v_{n-1} - 2v_{n-2}$. Therefore T_n is surrounded by $T_n \pm 2v_{n-1}, T_n \pm v_n, T_n \pm v_{n+1}$. We have, using (4.6) and the preceding proof,

$$
\begin{aligned}
T_n + v_n &\subset T_{n+1},\\
T_n + v_{n+1} &\subset T_{n+1} + v_{n+1} \subset T_{n+2},\\
T_n - 2v_{n-1} &= T_n + v_{n+1} - v_n \subset T_{n+1} + (v_{n+1} - 2v_n) \subset T_{n+2} + v_{n+2} \subset T_{n+3},\\
T_n - v_n &\subset T_{n+1} - 2v_n = T_{n+1} + (v_{n+2} - v_{n+1}) \subset T_{n+2} + v_{n+3} \subset T_{n+4},\\
T_n + 2v_{n-1} &= T_n + v_n - v_{n+1} \subset T_{n+1} - v_{n+1} \subset T_{n+2} - 2v_{n+1} \subset T_{n+2} +\\
&\quad + v_{n+3} - v_{n+2} \subset T_{n+3} + v_{n+3} - 2v_{n+2} \subset T_{n+4} + v_{n+4} \subset T_{n+5},\\
T_n - v_{n+1} &\subset T_{n+1} - v_{n+1} \subset T_{n+2} - 2v_{n+1} \subset T_{n+2} - v_{n+2} + v_{n+3}\\
&\quad \subset T_{n+3} + v_{n+3} - 2v_{n+2} \subset T_{n+4} + v_{n+4} \subset T_{n+5}.
\end{aligned}
$$

Thus all of the neighbours of T_n are contained in T_{n+5}. □

An example of this corollary is illustrated in Fig. 4.7, whereby all neighbours of T_4 are contained within T_9. This also shows that the 5 in Corollary 7 is minimal.

4.2.3 The Boundary of Maximal Primitive Ghosts

In this section we investigate the boundary of U_n. We call $Q \in \mathbb{Z}^2$ a 4-neighbour of $P \in \mathbb{Z}^2$ if $|P-Q| = 1$. We define a 4-path as the union of $n+1$ distinct points $P_0, P_1, \ldots, P_n \in \mathbb{Z}^2$ with $|P_i - P_{i-1}| = 1$ for $i = 1, \ldots, n$ together with the line segments between P_{i-1} and P_i for $i = 1, \ldots, n$. If $P_1, \ldots, P_n$ are distinct, but $P_0 = P_n$, we call it a 4-contour. We say that a path has length (τ_h, τ_v) if the number of horizontal line segments of length 1 of a path is τ_h and the number of vertical line segments of length 1 of the path is τ_v. In T_n we shall define six separation points A_n, B_n, C_n, D_n, E_n, F_n where three tiles meet (Fig. 4.8).

We study how these points move as we move from T_n to T_{n+1}. Two of these six points are boundary points of U_n where it meets $U_{n+1} \setminus U_n$. If $\epsilon_n = -1$,

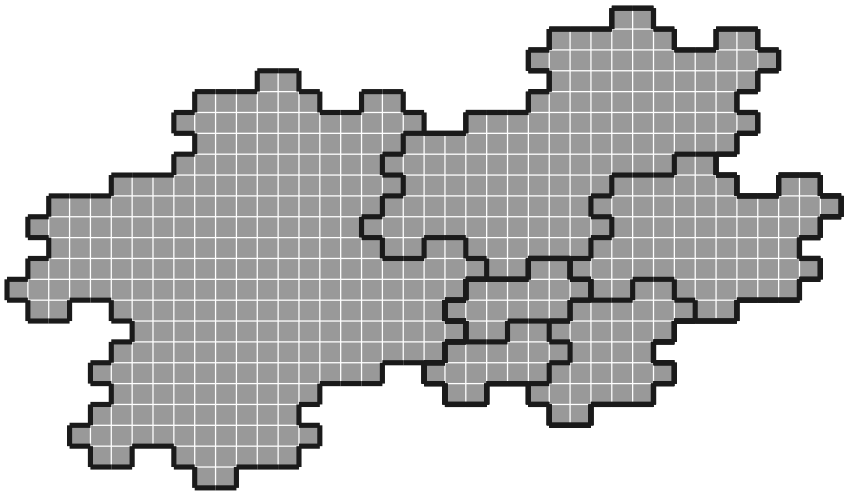


Figure 4.7: This figure shows an example of Corollary 7 for U_9, in which U_4 (in the center) and all of its neighbours are contained. Black borders separate the ghost tiles $U_4, U_5, \ldots, U_9$.

then these points are E_n and F_n (see Fig. 4.9), if $\epsilon_n = 1$, then these points are A_n and B_n. Define for $n = 2, 3, \ldots,$

$$A_{n+1} = F_n + v_n,\ B_{n+1} = A_n + v_n,\ C_{n+1} = B_n,$$

$$D_{n+1} = C_n,\ E_{n+1} = D_n,\ F_{n+1} = E_n + v_n$$

when $\epsilon_n = -1$. In the case that $\epsilon_n = +1$, we define

$$A_{n+1} = A_n + v_n,\ B_{n+1} = F_n + v_n,\ C_{n+1} = E_n,$$

$$D_{n+1} = D_n,\ E_{n+1} = C_n,\ F_{n+1} = B_n + v_n.$$

For both cases, the initial values are

$$A_2 = (3, 2),\ B_2 = (3, 1),\ C_2 = (1, 0),$$

$$D_2 = (0, 0),\ E_2 = (0, 1),\ F_2 = (2, 2).$$

Lemma 8. *For $n = 2, 3, \ldots$ we have $A_n - E_n = v_{n-1} + 2v_{n-2}$, $B_n - F_n = -(v_{n-1} - 2v_{n-2})$, $C_n - A_n = -2v_{n-1}$, $D_n - B_n = -(v_{n-1} + 2v_{n-2})$, $E_n - C_n = v_{n-1} - 2v_{n-2}$, $F_n - D_n = 2v_{n-1}$.*

Proof. By induction. The formulas hold for $n = 2$ and we shall choose the last formula for the inductive hypothesis when $\epsilon = -1$. The increment of the indices then implies that the formulas hold for all n, as follows

$$\begin{aligned}
&A_{n+1} - E_{n+1} = (F_n + v_n) - D_n = v_n + 2v_{n-1},\\
&B_{n+1} - F_{n+1} = (A_n + v_n) - (E_n + v_n) = v_{n-1} + 2v_{n-2} = -(v_n - 2v_{n-1}),\\
&C_{n+1} - A_{n+1} = B_n - (F_n + v_n) = -v_n - v_{n-1} + 2v_{n-2} = -2v_n.
\end{aligned}$$

The other proofs are similar. □

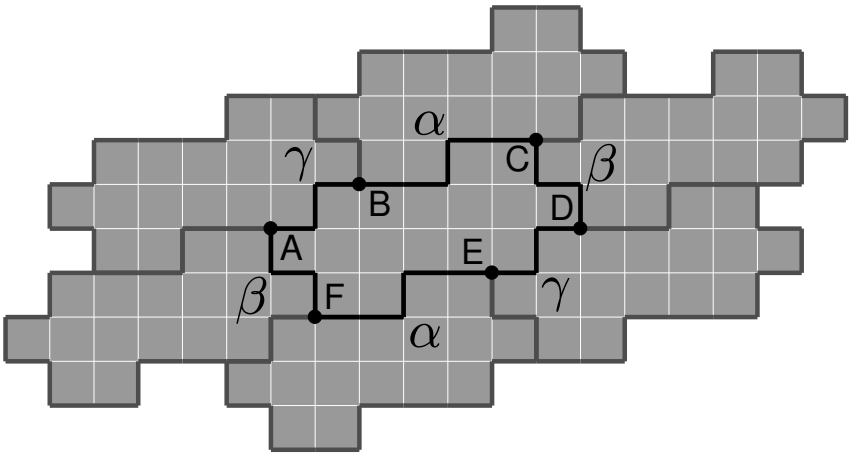


Figure 4.8: The case $\epsilon_n = -1$ for all n. Tile U_4 surrounded by six neighbours, connected by path lengths given in Theorem 9. In the points A, B, C, D, E, F three tiles meet.

In the next theorem we measure the lengths along the boundary of U_n between consecutive separation points. Here index h refers to horizontal distance and index v to vertical distance. Define for $n = 2, 3, \ldots,$

$$\alpha_{h,n+1} = \beta_{h,n} + 2\gamma_{h,n},\ \beta_{h,n+1} = \alpha_{h,n},\ \gamma_{h,n+1} = \beta_{h,n},$$

$$\alpha_{v,n+1} = \beta_{v,n} + 2\gamma_{v,n},\ \beta_{v,n+1} = \alpha_{v,n},\ \gamma_{v,n+1} = \beta_{v,n},$$

when $\epsilon_n = -1$. In case $\epsilon_n = +1$, define

$$\alpha_{h,n+1} = \beta_{h,n} + 2\alpha_{h,n},\ \beta_{h,n+1} = \gamma_{h,n},\ \gamma_{h,n+1} = \beta_{h,n},$$

$$\alpha_{v,n+1} = \beta_{v,n} + 2\alpha_{v,n},\ \beta_{v,n+1} = \gamma_{v,n},\ \gamma_{v,n+1} = \beta_{v,n}.$$

In both cases define

$$\alpha_n = \alpha_{h,n} + \alpha_{v,n},\ \beta_n = \beta_{h,n} + \beta_{v,n},\ \gamma_n = \gamma_{h,n} + \gamma_{v,n}$$

with initial values

$$\alpha_{h,2} = 2,\ \beta_{h,2} = 1,\ \gamma_{h,2} = 0,$$
$$\alpha_{v,2} = 1,\ \beta_{v,2} = 0,\ \gamma_{v,2} = 1.$$

Theorem 9. *The total length of the 4-contour surrounding U_n is equal to $2(\alpha_n + \beta_n + \gamma_n)$. For $n = 2, 3, \ldots$ we have*
$U_n \cap (U_n + (v_{n-1} + 2v_{n-2}))$ is a 4-path of length γ_n between A_n and B_n,
$U_n \cap (U_n - (v_{n-1} - 2v_{n-2}))$ is a 4-path of length α_n between B_n and C_n,
$U_n \cap (U_n - 2v_{n-1})$ is a 4-path of length β_n between C_n and D_n
$U_n \cap (U_n - (v_{n-1} + 2v_{n-2}))$ is a 4-path of length γ_n between D_n and E_n,
$U_n \cap (U_n + (v_{n-1} - 2v_{n-2}))$ is a 4-path of length α_n between E_n and F_n,
$U_n \cap (U_n + 2v_{n-1})$ is a 4-path of length β_n between F_n and A_n.

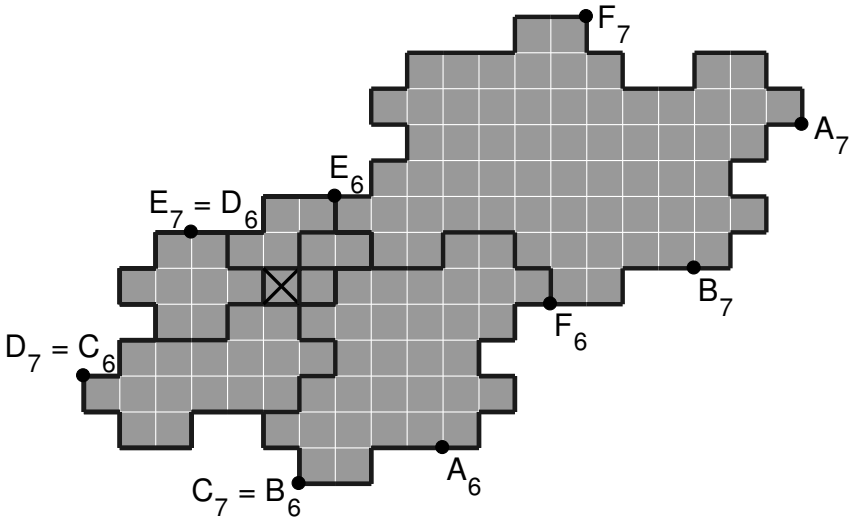


Figure 4.9: Tiles U_1 to U_7 for $\epsilon_2 = ... = \epsilon_6 = -1$. $A_6 = (5,-4), B_6 = C_7 = (1,-5), C_6 = D_7 = (-5,-2), D_6 = E_7 = (-2,2), E_6 = (2,3), F_6 = (8,0), A_7 = (15,5), B_7 = (12,1), F_7 = (9,8)$. E_6 and F_6 are the boundary points of U_7 where U_6 and $U_7 \setminus U_6$ meet.

Proof. We prove Theorem 9 by induction. We have $U_2 = \{(x,y) : 0 \leq x \leq 2, 0 \leq y \leq 1\} \cup \{(x,y) : 1 \leq x \leq 3, 1 \leq y \leq 2\}$, $v_0 = (1,0)$, $v_1 = (1,1)$ and the statements hold for $n = 2$. Suppose the statements are valid for n. By Theorem 5 the neighbours of U_{n+1} are $U_{n+1} \pm (v_n - 2v_{n-1}), U_{n+1} \pm (v_n + 2v_{n-1})$ and $U_{n+1} \pm 2v_n$. We compute the intersection of U_{n+1} and each of its neighbours.

We have

$$U_{n+1} \cap (U_{n+1} + (v_n + 2v_{n-1})) = (U_n \cap (U_n + v_n + 2v_{n-1}))$$

$$\cup (U_n \cap (U_n + 2v_n + 2v_{n-1})) \cup ((U_n + v_n) \cap (U_n + v_n + 2v_{n-1}))$$

$$\cup ((U_n + v_n) \cap (U_n + 2v_n + 2v_{n-1})).$$

Since $v_n + 2v_{n-1} = 3v_{n-1} \pm 2v_{n-2}, 2v_n + 2v_{n-1} = 4v_{n-1} \pm 4v_{n-2} \notin \{\pm(v_{n-1} - 2v_{n-2}), \pm(v_{n-1} + 2v_{n-2}), \pm 2v_{n-1}\}$, the first two intersections on the right-hand side are empty. By the induction hypothesis, and $(U_n + v_n) \cap (U_n + v_n + 2v_{n-1}) = v_n + (U_n \cap (U_n + 2v_{n-1}))$, the third intersection is the 4-path between $v_n + F_n$ and $v_n + A_n$ of length $(\beta_{h,n}, \beta_{v,n})$, that is the 4-path between A_{n+1} and B_{n+1} of length $(\gamma_{h,n+1}, \gamma_{v,n+1})$. The fourth intersection is empty, because it equals the first intersection shifted by v_n. We conclude that the second statement of Theorem 9 is true for $n + 1$.

Secondly

$$U_{n+1} \cap (U_{n+1} - (v_n - 2v_{n-1})) = (U_n \cap (U_n - v_n + 2v_{n-1}))$$

$$\cup (U_n \cap (U_n + 2v_{n-1})) \cup ((U_n + v_n) \cap (U_n - v_n + 2v_{n-1}))$$

$$\cup ((U_n + v_n) \cap (U_n + 2v_{n-1})).$$

If $\epsilon_{n+1} = -1$, the first intersection on the right-hand is $U_n \cap (U_n + (v_{n-1} + 2v_{n-2}))$, which is the 4-path between $A_n = B_{n+1} - v_n$ and $B_n = C_{n+1}$ of length $(\gamma_{h,n}, \gamma_{v,n})$ due to the induction hypothesis. Similarly with $\epsilon_{n+1} = +1$, this intersection becomes $U_n \cap (U_n + (v_{n-1} - 2v_{n-1}))$, the 4-path between $E_n = C_{n+1}$ and $F_n = B_{n+1} - v_n$ of length $(\alpha_{h,n}, \alpha_{v,n})$. The second intersection is the 4-path between F_n and A_n of length $(\beta_{h,n}, \beta_{v,n})$. If $\epsilon_{n+1} = -1$ this is the path from $A_{n+1} - v_n$ to $B_{n+1} - v_n$, if $\epsilon_{n+1} = 1$ this is the path from $B_{n+1} - v_n$ to $A_{n+1} - v_n$. The third intersection is empty by Theorem 5. The fourth intersection is the same as the first, shifted by v_n, hence if $\epsilon_{n+1} = -1$ a 4-path between B_{n+1} and $C_{n+1} + v_n$, if $\epsilon_{n+1} = 1$ a 4-path between $C_{n+1} + v_n = A_{n+1} - v_n$ and B_{n+1}. Therefore the intersection on the left-hand side is a 4-path from C_{n+1} via $B_{n+1} - v_n$ and $A_{n+1} - v_n$ to B_{n+1} of length $(\beta_{h,n} + 2\gamma_{h,n}, \beta_{v,n} + 2\gamma_{v,n}) = (\alpha_{h,n+1}, \alpha_{v,n+1})$ if $\epsilon_{n+1} = -1$ and of length $(\beta_{h,n} + 2\alpha_{h,n}, \beta_{v,n} + 2\alpha_{v,n}) = (\alpha_{h,n+1}, \alpha_{v,n+1})$ if $\epsilon_{n+1} = 1$.

Thirdly

$$U_{n+1} \cap (U_{n+1} - 2v_n) = (U_n \cap (U_n - 2v_n)) \cup (U_n \cap (U_n - v_n))$$

$$\cup((U_n + v_n) \cap (U_n - 2v_n)) \cup ((U_n + v_n) \cap (U_n - v_n)).$$

Prior arguments show that the first, third, and fourth intersections on the right-hand side are empty. The second intersection becomes $U_n \cap (U_n - (v_{n-1} - 2v_{n-1}))$ when $\epsilon_{n+1} = -1$ is used, which is the 4-path between $B_n = C_{n+1}$ and $C_n = D_{n+1}$ of length $(\alpha_{h,n}, \alpha_{v,n})$. When we take $\epsilon_{n+1} = +1$, the intersection is $U_n \cap (U_n - (v_{n-1} + 2v_{n-1}))$, a 4-path between $D_n = D_{n+1}$ and $E_n = C_{n+1}$ of length $(\gamma_{h,n}, \gamma_{v,n})$. We conclude that the left-hand side is the 4-path between C_{n+1} and D_{n+1} of length $(\beta_{h,n+1}, \beta_{v,n+1})$.

By symmetry the proofs of the other three statements are similar. Summing up the above results the length of the 4-contour surrounding U_{n+1} equals $(2(\alpha_{h,n+1} + \beta_{h,n+1} + \gamma_{h,n+1}), 2(\alpha_{v,n+1} + \beta_{v,n+1} + \gamma_{v,n+1}))$. □

We call $P \in \mathbb{Z}^2$ an 8-neighbour of $Q \in \mathbb{Z}^2$ if $|P - Q| \in \{1, \sqrt{2}\}$. We define an 8-cycle as the set of m distinct points $P_1, P_2, \ldots, P_m = P_0 \in \mathbb{Z}^2$ such that P_i is an 8-neighbour of P_{i-1} for $i = 1, \ldots, m$. Observe that a contour includes the connecting line segments, whereas a cycle consists only of a series of points.

4.3 Boundary Ghosts

We now investigate boundary ghosts which are closely related to the previously constructed maximal ghosts. The domains of boundary ghosts consist of an annulus with a hollow interior (see Fig. 4.10). Boundary ghosts are constructed using the same recursive sequence of directions (4.6), but preceded by the direction $v_{-1} = (0, 1)$.

Let G_n be the maximal ghost studied in the preceding sections with domain T_n. Define the boundary ghost G_n^* by directions $v_{-1}, v_0, \ldots, v_{n-1}$ where G_{-1}^* has

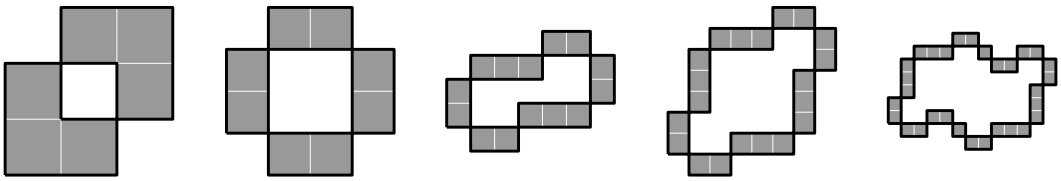


Figure 4.10: The boundary ghost domains U_2^*, U_3^*, U_4^*, U_5^*, U_6^* in case $\epsilon_n = -1$ for $n = 2$ to 5.

domain $\{(0,0)\}$ and $G_{-1}^*(0,0) = 1$. Recall that if $(x,y) \in T_n$, then $G(x,y) = 1$ if x is even and $G(x,y) = -1$ if x is odd. If $(x,y) \in \mathbb{Z}^2 \setminus T_n$, then $G(x,y) = 0$. We have $G_n^* = G_n(x,y) - G_n(x,y-1)$. Let T_n^* be the ghost domain of G_n^*. By the definition of a ghost domain it follows that

$$T_n^* = \big(T_n \cup \big(T_n + (0,1)\big)\big) \setminus \big(T_n \cap \big(T_n + (0,1)\big)\big).$$

Theorem 10. *For $n > 1$ the point $(x,y) \in \mathbb{Z}^2$ is an element of T_n^* if and only if the line segment between (x,y) and $(x+1,y)$ is part of the 4-contour around U_n.*

Proof. If there is a boundary line segment between (x,y) and $(x+1,y)$ of U_n, then one among (x,y) and $(x,y-1)$ belongs to T_n and the other does not. Therefore $(x,y) \in T_n^*$. If $(x,y) \in T_n^*$, then $G(x,y-1) \neq G(x,y)$. Hence one of them is 0 and the other is ± 1. It follows that the line segment between (x,y) and $(x+1,y)$ is a boundary line segment of U_n. □

This theorem is illustrated in Fig. 4.11 for $n = 8$. The number of horizontal line segments of length 1 in Fig. 4.11a equals the number of light grey pixels in Fig. 4.11b.

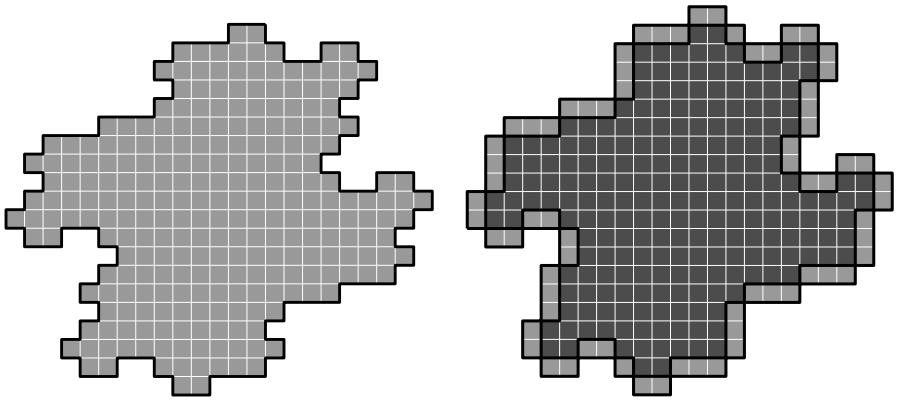


Figure 4.11: a) Maximal ghost domain U_8. b) Boundary ghost domain U_8^*. The number of horizontal line segments of length 1 on the left equals the number of grey pixels on the right. Light squares indicate pixels of the boundary ghost domain, and dark squares show the interior region $T_n \cap \big(T_n + (0,1)\big)$ with ghost values 0.

By Theorem 9 we know that the total number of horizontal boundary unit segments of U_n equals $2(\alpha_{h,n} + \beta_{h,n} + \gamma_{h,n})$. Observe that it follows from Theorem 9 that for $n > 0$ every point in T_n has a 4-neighbour. Half of the points in T_n^*, the points (x, y) with $(x, y+1) \notin T_n^*$, do not belong to T_n and half of them, the points (x, y) with $(x, y-1) \notin T_n^*$, do belong to T_n. Hence we have constructed a boundary ghost T_n^* consisting of $2(\alpha_{h,n}+\beta_{h,n}+\gamma_{h,n})$ points with $2^n - (\alpha_{h,n} + \beta_{h,n} + \gamma_{h,n})$ points in its interior. We shall show that every boundary ghost forms an 8-cycle. To do so we need the following results.

Lemma 11. *Let $n, x, y \in \mathbb{Z}$ with $n > 0$.* a) *If the line segment from (x, y) to $(x, y+1)$ is a boundary line segment of U_n, then $x + y$ is even.* b) *Every vertical boundary line segment of U_n has length* 1.

Proof. a) By induction on n. It is true for $n = 1$. Suppose it is valid for n. Since $U_{n+1} = U_n \cup (U_n + v_n)$, a boundary line segment from (x, y) to $(x, y+1)$ of U_{n+1} either originates from U_n itself, in which case $x+y$ is even by induction, or it originates from a boundary line segment from $(x - p_n, y - q_n)$ to $(x - p_n, y - q_n + 1)$ in U_n in which case $x - p_n + y - q_n$ is even by induction. Since p_n and q_n are odd for all $n > 0$ by (4.9), $x + y$ is even in both cases.
b) If the line segment from (x, y) to $(x, y+2)$ would be a boundary line segment of U_n, then by a) both $x+y$ and $x+y+1$ would be even, a contradiction. □

Theorem 12. *For all $n > 0$, the points of the boundary ghost domain T_n^* constitute an 8-cycle.*

Proof. According to Theorem 10 for every $P^* \in \mathbb{Z}^2$ we have $P^* \in T_n^*$ if and only if the line segment between P^* and $P^* + (1, 0)$ is part of the 4-contour around U_n. Thus going around the 4-contour and collating the left lattice points of the horizontal unit line segments we write T_n^* as a cycle $(P_1^*, P_2^*, \ldots, P_m^* = P_0^*)$ for some m.

It suffices to show that for $i = 0, 1, \ldots, m-1$ the distance between P_i^* and P_{i+1}^* is 1 or $\sqrt{2}$. Let $0 \le i < m$. Without loss of generality we may assume that the contour leaves P_i^* to the right to reach P_{i+1}^* as the next element of T_n^*. Because of Lemma 11 there are only five routes to reach P_{i+1}^* as is illustrated in Fig. 4.12 where the elements of T_n^* are indicated by thick dots. In each case the distance between P_i^* and P_{i+1}^* is 1 or $\sqrt{2}$. □

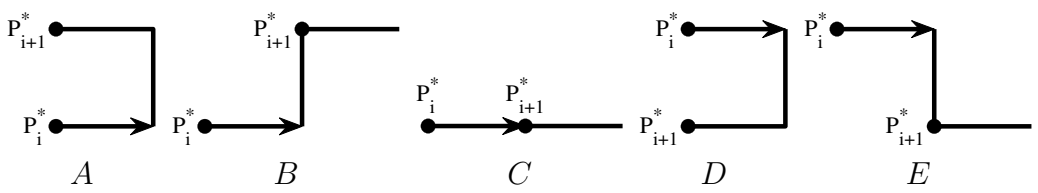


Figure 4.12: The five possible extensions of the contour around U_n starting at the line segment from P_i^* to $P_i^* + (1, 0)$. The distance from P_i^* to P_{i+1}^* is either 1 or $\sqrt{2}$.

Figure 4.13 shows a range of possible boundary ghost domains at three levels. At the top of the tree is U_2^*, and each step gives rise to two possible boundary ghost domains. Along the left branches, the directions are generated from the recursion with $\epsilon_n = -1$, and along the right branches are the directions with $\epsilon_n = +1$. Although there are degeneracies in the shapes produced, this is uncommon for a larger number of directions.

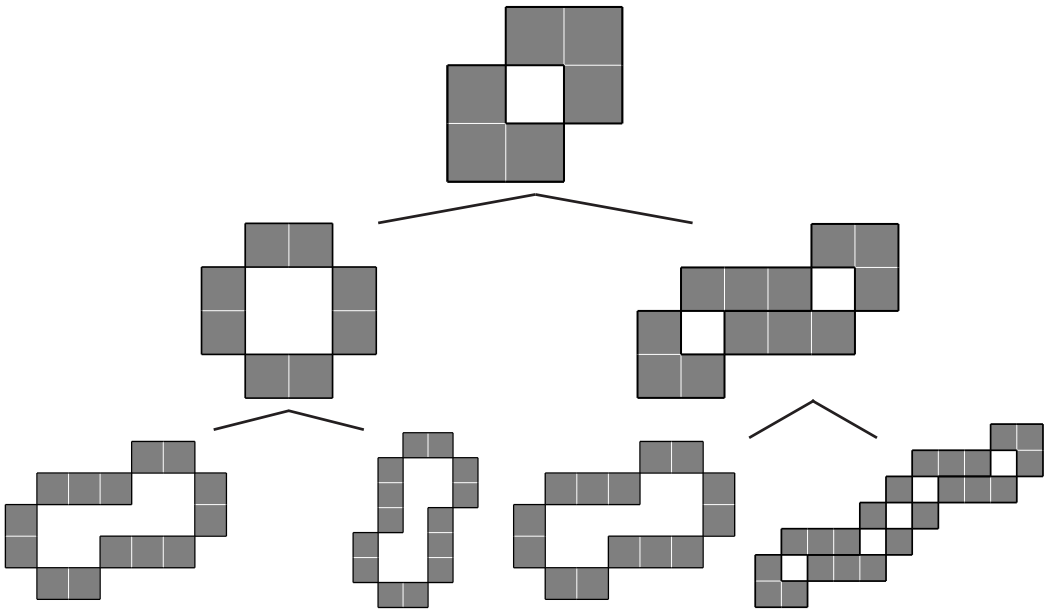


Figure 4.13: Tree of boundary ghosts shown for three levels. Ghosts branching to the left in the tree ($\epsilon_n = -1$) will produce a more radially symmetric boundary, while the right side branches ($\epsilon_n = +1$) produce longer diagonal sets of points.

Maximal pixel ghost domains that have parts connected by a single point will result in multiple empty interior regions. Rather than the recursive construction, boundary ghosts can alternatively viewed as a transformation of a maximal ghost by adding the direction (0,1). From this perspective, the direction (0,1) removes all points $(x, y) \in T_n$ for which $(x, y-1) \in T_n$. If $(x, y) \in T_n$, then $(x, y) \in T_n^*$ if and only if $(x, y-1) \notin T_n$. If $(x, y) \notin T_n$, then $(x, y) \in T_n^*$ if and only if $(x, y-1) \in T_n$. If $(x, y) \in T_n^*$ and $(x, y-1), (x, y+1) \notin T_n^*$, then the interior of U_n^* may split into two components.

An example of this phenomenon can be found in Fig. 4.14, and Fig. 4.13 with the ghosts along the right branches of the tree. Here, the maximal ghosts have more than one part joined by a single point of connection, which results in boundary ghosts having more than one interior region. This phenomenon only occurs when recursions with $\epsilon_n = +1$ are used.

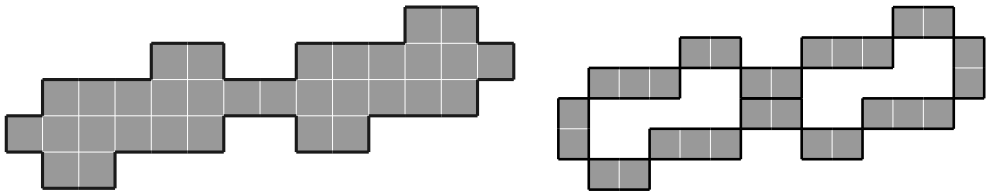


Figure 4.14: Maximal ghost with $\epsilon_2 = +1$, $\epsilon_3 = -1$ and $\epsilon_4 = +1$, and boundary ghost.

4.4 A Projective Analogue to Pick's Theorem

Before deriving some properties on the size of maximal primitive ghosts, we take an aside to present a new result linking the area of convex symmetric polygons to the number of projection bins and number of polygon sides. This result was published in [15]. While we shall use this result in the context of convex hulls of ghosts, it is a general geometric result.

Pick's theorem expresses the area of a polygon on a grid in terms of the number of boundary and interior integer lattice points. Here, we present an analogous theorem for the area of a symmetric, convex polygon in terms of the number of polygon edges and total projection bins. These polygons arise naturally through discrete projection ghosts. In this section, we show that the area A of a ghost's convex hull is related to the number of non-trivial discrete projection bins B over the ghost image for any set of N 2D discrete projections by $A = B/2 - N/2$. The ratio B/A has a strong upper bound of exactly 2. This relation is analogous to Pick's theorem for polygons with lattice point vertices.

The problem of counting the number of discrete projection bins is usually concerned with a rectangular region, which has been studied in great detail by Verbert [77] for n-dimensional projections. In this section we find the total number of non-trivial discrete (parallel ray) projection bins B over the null-set or ghost image for any set of 2D discrete projections comprised of N discrete angles, (p_i, q_i) for $i = 1, 2, \ldots, N$. We prove $B = 2A + N$, where A is the area of the convex hull of the ghost.

Though we state the result for ghosts, it may be applied to any polygonal region of interest defined by the discrete projection directions. This allows for quick calculation of the number of bins required to store information contained within an area of interest in tomographic reconstructions, also of relevance to digital security applications where data is encoded in discrete projections.

Pick's theorem evaluates the exact area of a simple polygon whose vertices are fixed to any set of 2D integer lattice points. The polygon area A is evaluated by counting the number of interior i and boundary points b through $A = i + b/2 - 1$ [56]. Generalisations of Pick's theorem to higher dimensions do not exist, as demonstrated through the Reeve tetrahedron [60]. The number of projection bins over a discrete ghost in 3D produces similar difficulties.

4.4.1 Convex Hull of a Ghost

Let S be any set of N 2D discrete projection directions, $S = \{(p_i, q_i)\}$, $1 \leq i \leq N$, ordered by their angles, $\theta_i = \tan^{-1}(q_i/p_i)$. To avoid degenerate angles, we require $\gcd(p_i, q_i) = 1$ and $p_i \geq 0$. The set of vectors along with its negatives, $S \cup -S$, form the sequential tangents of a closed, convex, symmetric polygon. As the convex hull circumscribes all points of the projection ghost, it provides an upper bound for the area of the reconstructed image where pixel values cannot be uniquely determined. Direction (p_i, q_i) accumulates q_i discrete projection bins for each horizontal unit step and p_i projection bins for each vertical unit step on a square grid [40].

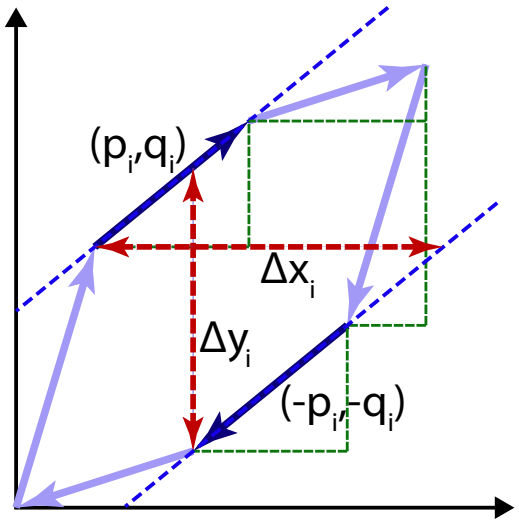


Figure 4.15: Geometry of a ghost's convex hull. Red lines show the vertical and horizontal distances, found by summing tangent vector components and subtracting similar triangles, shown in green.

To find the horizontal and vertical distance between parallel sides of the symmetric polygon, we first find the linear equations for each line corresponding to the i^{th} direction. Both lines have a gradient of q_i/p_i, and we can find the y-intersection using a point from each line. To find the first point, we sum the sorted vectors up to i to traverse the polygon clockwise until we reach the desired direction. Hence, the line for the vector (p_i, q_i) is

$$y_+ = \frac{q_i}{p_i}\left(x - \sum_{j=1}^{i} p_j\right) + \sum_{j=1}^{i} q_j. \tag{4.10}$$

To find the second point, we subtract the negative vectors back to $i+1$ to traverse polygon in the counter-clockwise direction from the origin, and the line for $(-p_i, -q_i)$ is given by

$$y_- = \frac{q_i}{p_i}\left(x - \sum_{j=i+1}^{N} p_j\right) + \sum_{j=i+1}^{N} q_j. \tag{4.11}$$

From here, it is simple to find the vertical and horizontal distances between the two lines. This can also be found by traversing the proceeding N tangent vectors and subtracting similar triangles (Fig. 4.15). The vertical distance Δy_i between tangent vector (p_i, q_i) and its negative $(-p_i, -q_i)$ is therefore

$$\Delta y_i = \sum_{j=1}^{i} q_j - \sum_{j=i+1}^{N} q_j - \frac{q_i}{p_i}\left(\sum_{j=1}^{i} p_j - \sum_{j=i+1}^{N} p_j\right). \tag{4.12}$$

When $(p_i, q_i) = (0, 1)$, we use the horizontal distance Δx_i which is calculated similarly.

$$\Delta x_i = -\sum_{j=1}^{i} p_j + \sum_{j=i+1}^{N} p_j + \frac{p_i}{q_i}\left(\sum_{j=1}^{i} q_j - \sum_{j=i+1}^{N} q_j\right) \tag{4.13}$$

4.4.2 Number of Discrete Projection Bins

The number of bins between the parallel tangents b_i, for (p_i, q_i) scales in proportion to Δx_i (and Δy_i), as illustrated in Fig. 4.16.

$$\begin{aligned}
b_i &= \Delta x_i q_i + 1 \\
&= \Delta y_i p_i + 1 \\
&= p_i\left[\sum_{j=1}^{i} q_j - \sum_{j=i+1}^{N} q_j - \frac{q_i}{p_i}\left(\sum_{j=1}^{i} p_j - \sum_{j=i+1}^{N} p_j\right)\right] + 1 \\
&= p_i\left(\sum_{j=1}^{i} q_j - \sum_{j=i+1}^{N} q_j\right) - q_i\left(\sum_{j=1}^{i} p_j - \sum_{j=i+1}^{N} p_j\right) + 1
\end{aligned} \tag{4.14}$$

The total number of discrete projection bins B is given by the sum of the number of bins for all N angles.

$$\begin{aligned}
B &= \sum_{i=1}^{N} b_i \\
&= \sum_{i=1}^{N}\left[p_i\left(\sum_{j=1}^{i} q_j - \sum_{j=i+1}^{N} q_j\right) - q_i\left(\sum_{j=1}^{i} p_j - \sum_{j=i+1}^{N} p_j\right) + 1\right] \\
&= \sum_{i=1}^{N}\left[p_i\left(2\sum_{j=1}^{i} q_j - \sum_{j=1}^{N} q_j\right) - q_i\left(2\sum_{j=1}^{i} p_j - \sum_{j=1}^{N} p_j\right)\right] + N \\
&= 2\left[\sum_{i=1}^{N} p_i \sum_{j=1}^{i} q_j - \sum_{i=1}^{N} q_i \sum_{j=1}^{i} p_j\right] + N \\
&= 2\left[\left(p_1 q_1 + \sum_{i=2}^{N} p_i \sum_{j=1}^{i} q_j\right) - \left(q_1 p_1 + \sum_{i=2}^{N} q_i \sum_{j=1}^{i} p_j\right)\right] + N
\end{aligned}$$

$$
\begin{aligned}
&= 2\sum_{i=2}^{N}\left[\left(p_i\sum_{j=1}^{i-1}q_j + p_iq_i\right) - \left(q_i\sum_{j=1}^{i-1}p_j + q_ip_i\right)\right] + N \\
&= 2\sum_{i=1}^{N-1}\left[p_{i+1}\sum_{j=1}^{i}q_j - q_{i+1}\sum_{j=1}^{i}p_j\right] + N
\end{aligned} \tag{4.15}
$$

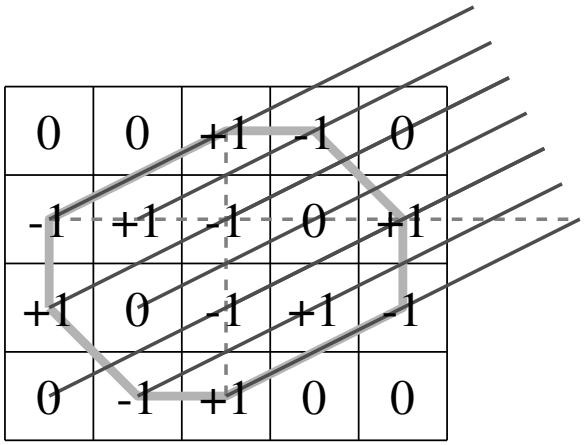


Figure 4.16: Projection $(2,1)$ of a ghost with $S = \{(0,1), (2,1), (1,0), (1,-1)\}$. There are 7 bins covered by ghost elements, with $\Delta x_i = 6$ and $\Delta y_i = 3$. By definition, all bins have a zero sum.

4.4.3 Convex Hull Area

The area A inside the convex polygon tangents is the sum of internal triangles, given by half of the sum of the determinants of sequential pairs of the coordinates around the M polygon vertices (u_i, v_i) for $i = 1, 2, \ldots, M$. The area is positive if these coordinates are traversed in anticlockwise order.

$$
\begin{aligned}
A =& \frac{1}{2}\left(\begin{vmatrix} u_1 & u_2 \\ v_1 & v_2 \end{vmatrix} + \begin{vmatrix} u_2 & u_3 \\ v_2 & v_3 \end{vmatrix} + \ldots + \begin{vmatrix} u_M & u_1 \\ v_M & v_1 \end{vmatrix}\right) \\
=& \frac{1}{2}(u_1v_2 - u_2v_1 + u_2v_3 - u_3v_2 + \cdots + u_{M-1}v_M - u_Mv_{M-1} + u_Mv_1 - u_1v_M)
\end{aligned} \tag{4.16}
$$

We can choose to set our first vertex at the origin, so $(u_1, v_1) = (0, 0)$. We also have $M = 2N$ vertices.

$$
\begin{aligned}
A &= \frac{1}{2}\left(u_2v_3 - u_3v_2 + \ldots + u_{2N-1}v_{2N} - u_{2N}v_{2N-1}\right) \\
&= \frac{1}{2}\sum_{i=2}^{2N-1} u_iv_{i+1} - v_iu_{i+1}
\end{aligned} \tag{4.17}
$$

The points (u_i, v_i) for a ghost's polygon can be found by traversing the tangent angles.

$$(u_i, v_i) = \left(\sum_{j=1}^{i-1} q_j, \sum_{j=1}^{i-1} p_j \right) \tag{4.18}$$

Substituting (4.18) into (4.17), we can find the area in terms of the discrete directions that define the convex hull.

$$\begin{aligned}
A =& \frac{1}{2} \sum_{i=2}^{N} (u_i v_{i+1} - v_i u_{i+1}) + \frac{1}{2} \sum_{i=N+1}^{2N-1} (u_i v_{i+1} - v_i u_{i+1}) \\
=& \frac{1}{2} \sum_{i=2}^{N} \left[\sum_{j=1}^{i-1} q_j \sum_{j=1}^{i} p_j - \sum_{j=1}^{i-1} p_j \sum_{j=1}^{i} q_j \right] \\
&+ \frac{1}{2} \sum_{i=1}^{N-1} \left[\sum_{j=1}^{i} (-q_j) \sum_{j=1}^{i+1} (-p_j) - \sum_{j=1}^{i} (-p_j) \sum_{j=1}^{i+1} (-q_j) \right] \\
=& \sum_{i=1}^{N-1} \left[\sum_{j=1}^{i} q_j \sum_{j=1}^{i+1} p_j - \sum_{j=1}^{i} p_j \sum_{j=1}^{i+1} q_j \right] \\
=& \sum_{i=1}^{N-1} \left[p_{i+1} \sum_{j=1}^{i} q_j - q_{i+1} \sum_{j=1}^{i} p_j \right]
\end{aligned} \tag{4.19}$$

Through substitution of equation (4.19) into (4.15), we find

$$B = 2A + N. \tag{4.20}$$

Therefore, the number of discrete projection bins for any set $S = \{(p_i, q_i)\}$ of N directions can be found directly from the polygon area of the discrete ghost. For a large number of angles, the ratio B/A converges to 2.

As with Pick's theorem, equation (4.20) does not generalise simply to higher dimensions. This can be demonstrated with a counter example in 3D, by calculating the volume and the number of bins required to store projections of the object. Let $S = \{(p_i, q_i, r_i)\}$ be any set of 3D discrete directions. The sets $S_1 = \{(1,0,0), (0,1,0), (1,1,1)\}$, $S_2 = \{(1,0,0), (1,0,1), (0,1,1)\}$, and $S_3 = \{(1,1,0), (0,1,1), (1,0,1)\}$ have volumes $V_1 = 1$, $V_2 = 1$, $V_3 = 2$, and total number of bins $B_1 = 12$, $B_2 = 15$, $B_3 = 15$ respectively. The convex hulls are illustrated in Fig. 4.17. Hence, B and V are not one-to-one and therefore cannot have a similar relation in 3D. It is clear that trivial generalisations of equation (4.20) cannot similarly count bins in higher dimensions. However, simple relations between projection bins, hyper-volumes, and number of directions, may exist for certain classes of projection sets.

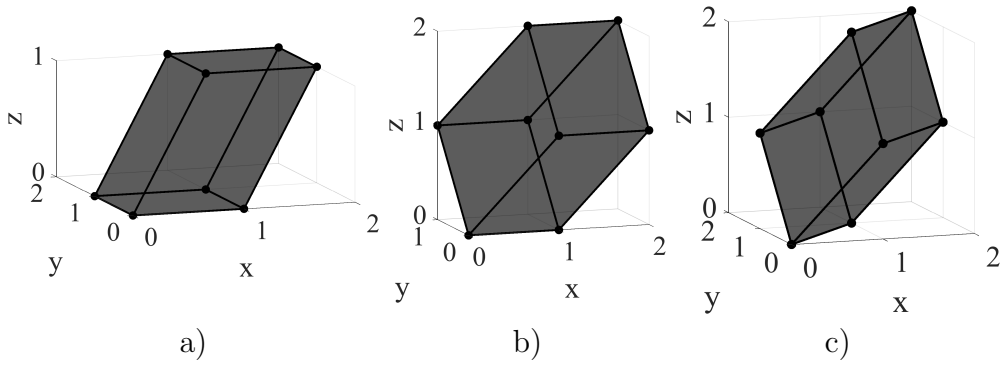


Figure 4.17: Convex hulls of ghosts for the set of directions: a) $S_1 = \{(1,0,0), (0,1,0), (1,1,1)\}$. b) $S_2 = \{(1,0,0), (1,0,1), (0,1,1)\}$. c) $S_3 = \{(1,1,0), (0,1,1), (1,0,1)\}$.

4.5 Explicit Size Bounds and Fill Factor

In most practical applications, ghosts are constructed on a rectangular array. Therefore, we wish to find the smallest rectangle that bounds the ghost domain, and calculate the fraction of this rectangle that contains ghost points. Here, the recursion (4.6) is restricted to $\epsilon_n = -1$ for all n. This restriction is made to analyse ghost domains that are closest to being radially symmetric. We prove that the size of the boundary ghost domain is logarithmically smaller than the size of the interior area.

We define the fill factor of U_N as the quotient of the area of U_N and the area of the smallest rectangle containing U_N. We have $q_0 = 0, q_1 = 1, q_{n+1} = q_n - 2q_{n-1}$ for $n = 1, 2, \ldots$. This is a binary recurrence sequence with characteristic polynomial $x^2 - x + 2$. Its zeros are $(1 \pm \sqrt{-7})/2$. Substituting the values for q_0 and q_1 we find

$$\begin{aligned} p_{n-1} = q_n &= \frac{1}{\sqrt{-7}}\left(\frac{1}{2} + \frac{1}{2}\sqrt{-7}\right)^n - \frac{1}{\sqrt{-7}}\left(\frac{1}{2} - \frac{1}{2}\sqrt{-7}\right)^n \\ &= \frac{2}{\sqrt{7}}\mathrm{Im}\left(\frac{1}{2} + \frac{1}{2}\sqrt{-7}\right)^n \end{aligned} \tag{4.21}$$

for $n = 1, 2, \ldots$. A rough estimate, but on average no more than a factor 2 wrong, yields

$$|q_n| \leq \frac{2}{\sqrt{7}} 2^{n/2}, \; |p_n| \leq \frac{2}{\sqrt{7}} 2^{(n+1)/2}. \tag{4.22}$$

It follows that

$$1 + \sum_{j=0}^{n-1} |q_j| \leq \frac{2}{\sqrt{7}} \frac{2^{n/2}}{\sqrt{2} - 1} \leq 1.825 \cdot 2^{n/2} \tag{4.23}$$

and

$$1+\sum_{j=0}^{n-1}|p_j| = 1+\sum_{j=0}^{n}|q_j| \le \frac{2}{\sqrt{7}}\frac{2^{(n+1)/2}}{\sqrt{2}-1} \le 2.581\cdot 2^{n/2}. \tag{4.24}$$

Therefore the fill factor of U_N equals

$$\frac{2^n}{\left(1+\sum_{j=0}^{n-1}|p_j|\right)\left(1+\sum_{j=0}^{n-1}|q_j|\right)} \ge \frac{2^n}{1.825\cdot 2.581\cdot 2^n} \ge 0.212. \tag{4.25}$$

A fill factor cannot exceed 1. The diameter of T_n is at most

$$\sqrt{\left(1+\sum_{j=0}^{n-1}|p_j|\right)^2+\left(1+\sum_{j=0}^{n-1}|q_j|\right)^2} \le \sqrt{1.825^2+2.581^2}\cdot 2^{n/2} < \sqrt{10}\cdot 2^{n/2}. \tag{4.26}$$

On the other hand, for a rectangular array with 2^n pixels you need a diameter which is at least $\sqrt{2}\cdot 2^{n/2}$. Thus the estimates (4.25) and (4.26) are wrong by factors at most 5 and $\sqrt{5}$, respectively. Computations show the fill factor to be above 45% (Fig. 4.18).

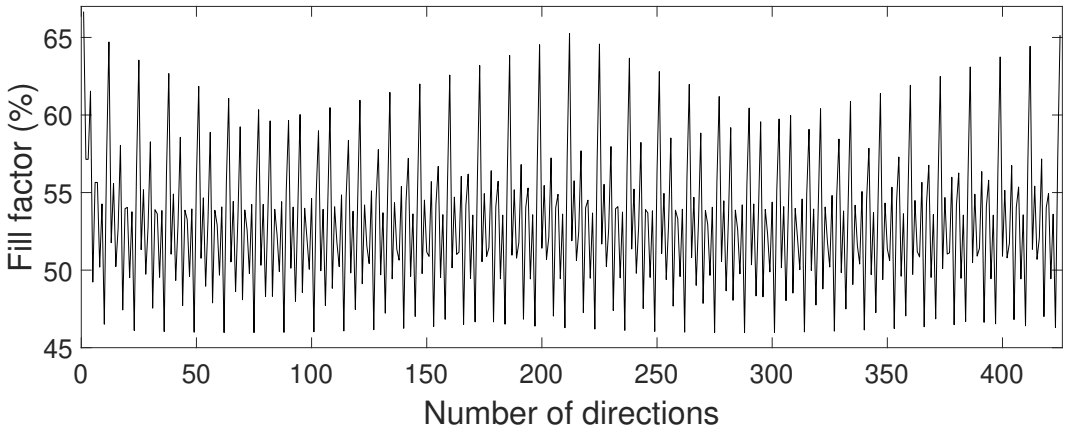


Figure 4.18: Computations show the fill factor to vary periodically between 0.46 and 0.65.

Remark. If the recursion (4.6) is used such that from some point on only $\epsilon_n = -1$ occurs, then the above estimates (4.25) and (4.26) hold with adjusted constants. The choice of the initial epsilons can be used to adjust the ratio of $\sum_{j=0}^{n}|p_j|$ and $\sum_{j=0}^{n}|q_j|$ to fit the length and height of the image to be reconstructed. If the ratio is between 1 and 1.5, then it is appropriate to use $\epsilon_j = -1$ for all j up to an n for which the ratio fits. If, for example, the ratio lies between 4 and 6, then a better choice is $\epsilon_2 = \epsilon_4 = \epsilon_5 = 1$ and the others -1. For higher ratios, longer strings of $\epsilon_j = 1$ are needed.

Using the results from Section 4.21, we can also derive bounds for the convex hull and number of bins for maximal primitive ghosts. Substituting

equation (4.21) into the formula for the area of a ghost convex hull yields

$$\begin{aligned} A &= \sum_{i=0}^{n-1}\sum_{j=0}^{i-1} |p_i q_j - p_j q_i| \qquad (4.27)\\ &= \frac{1}{7}\sum_{i=0}^{n-1}\sum_{j=0}^{i-1} 2^{(5+i+j)/2} \Big|(\sin((1+i)\arctan(\sqrt{7}))\sin(j\arctan(\sqrt{7})) \\ &\qquad\qquad - \sin(i\arctan(\sqrt{7}))\sin((1+j)\arctan(\sqrt{7})))\Big| \\ &\le \frac{1}{7}\sum_{i=0}^{n-1}\sum_{j=0}^{i-1} 2^{(5+i+j)/2} \\ &= \frac{8}{7}(\sqrt{2}+1)(2^{n/2}-1)(2^{(n+1)/2}-1). \end{aligned}$$

Therefore, from equation 4.20, the number of bins is no more than

$$B \le \frac{16}{7}(\sqrt{2}+1)(2^{n/2}-1)(2^{(n+1)/2}-1) + n. \qquad (4.28)$$

Theorem 13. *Suppose $\epsilon_n = -1$ for all n. Let $\rho \approx 1.5214$ be the positive zero of $x^3 - x - 2$ and σ and $\overline{\sigma}$ the nonreal zeros. Then the number of points of T_n^* equals*

$$c\rho^n + 2|\sigma|^n(\mathrm{Re}(c_1)\cos(n\arg(\sigma)) - \mathrm{Im}(c_1)\sin(n\arg(\sigma)))$$

with $\rho = 1.5214$, $\sigma = -0.7607 + 0.8579i$, $c = 2.4757$, and $c_1 = -0.2378 + 0.0747i$.

Proof. The number of points with a nonzero value equals

$$\begin{aligned} \delta_n &= 2(\alpha_{h,n} + \beta_{h,n} + \gamma_{h,n}) \\ &= 2(\alpha_{h,n} + \alpha_{h,n-1} + \alpha_{h,n-2}) \\ &= 2(\beta_{h,n-1} + 2\gamma_{h,n-1} + \beta_{h,n-2} + 2\gamma_{h,n-2} + \beta_{h,n-3} + 2\gamma_{h,n-3}) \\ &= 2(\alpha_{h,n-2} + \beta_{h,n-2} + \gamma_{h,n-2} + 2\alpha_{h,n-3} + 2\beta_{h,n-3} + 2\gamma_{h,n-3}) \\ &= \delta_{n-2} + 2\delta_{n-3} \end{aligned}$$

for $n \ge 3$. Since $x^3 - x - 2 = (x-\rho)(x-\sigma)(x-\overline{\sigma})$, there exist constants $c, c_1 \in \mathbb{C}$ such that

$$\begin{aligned} \delta_n &= c\rho^n + c_1\sigma^n + \overline{c_1}\,\overline{\sigma}^n \\ &= c\rho^n + |\sigma|^n(c_1 e^{in\arg(\sigma)} + \overline{c_1} e^{in\arg(\overline{\sigma})}) \\ &= c\rho^n + 2|\sigma|^n(\mathrm{Re}(c_1)\cos(n\arg(\sigma)) - \mathrm{Im}(c_1)\sin(n\arg(\sigma))). \end{aligned}$$

The constants c and c_1 can be computed from the initial values $\delta_1 = 4, \delta_2 = 6, \delta_3 = 8$. □

The following corollary is immediate.

Corollary 14. *T_n^* has $c\rho^n + O(|\sigma|^n)$ elements.*

Note that $\rho^n \approx 1.5214^n$ is relatively small compared to the size 2^n of T_n and that $|\sigma| \approx 1.1466$ is in turn much smaller than ρ. Since $1.5214 < 2$, the number of boundary ghost points in T_n^* is small compared to points in the interior area. For example, the ghost domain T_8^* in Fig. 4.11b has 70 boundary points and 221 interior points.

4.6 The Structure of Boundary Ghosts

In this section we analyse the internal structure of maximal, and consequently, boundary ghosts. We assume throughout this section that $\epsilon_j = -1$ for $j = 0, 1, \ldots$. The structure shall be formulated using two different methods, both giving a lower bound on the central interior area. Though Theorem 13 gives an exact number of points for the interior, calculating the shape of these points requires significant computation. Therefore the aim of this section is to provide simple bounds which are quick to compute to decide which ghost is most useful for a given application.

4.6.1 T_{n+5} as 32 Tiles of T_n

We show here that T_{n+5} is constructed from 32 translated tiles of T_n (Fig. 4.19), and these tiles maintain the same connectivity. This preserved connectivity can be used to develop a lower bound on the central area that can be uniquely determined in the case of boundary ghosts.

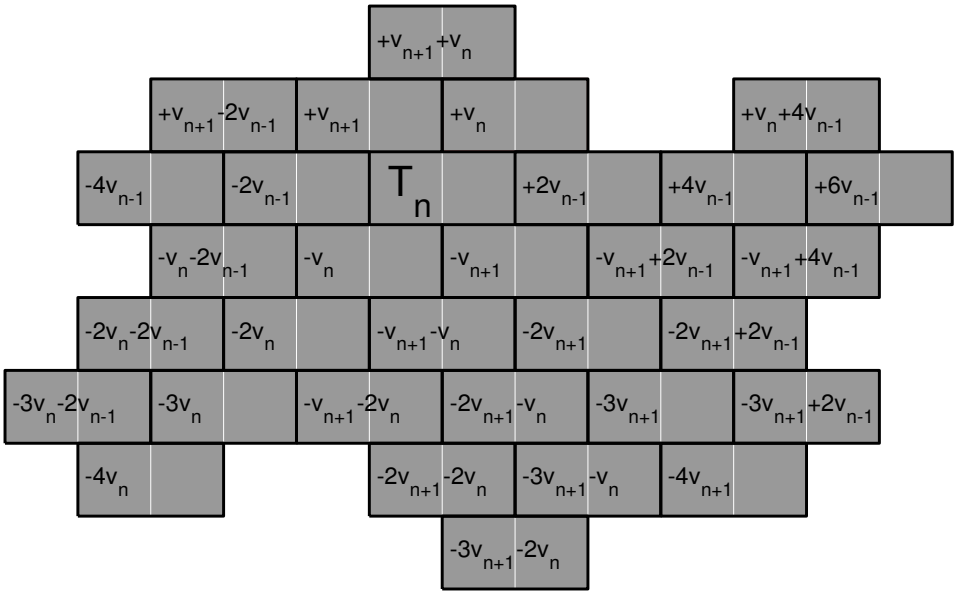


Figure 4.19: T_6 as 32 translated tiles of T_1. All T_{n+5} have the same connectivity for tiles T_n.

Let Δ denote the closed triangle with corners $(0,0), v_n, v_{n+1}$ and ∇ the closed triangle with corners $(0,0), v_{n+1}, v_{n+1} - v_n$.

Theorem 15. *Let $n \geq 2$. U_{n+5} contains the following 40 closed triangles:* $\Delta + v_{n+1} - v_n, \nabla + v_{n+1} - v_n, \Delta + v_{n+1} - 2v_n, \nabla + v_n, \Delta, \nabla, \Delta - v_n, \nabla - v_n, \Delta - 2v_n, \nabla - 2v_n, \Delta - 3v_n, \nabla - 3v_n, \Delta - 4v_n, \nabla - v_{n+1} + v_n, \Delta - v_{n+1}, \nabla - v_{n+1}, \Delta - v_{n+1} - v_n, \nabla - v_{n+1} - v_n, \Delta - v_{n+1} - 2v_n, \nabla - v_{n+1} - 2v_n, \Delta - 2v_{n+1} + v_n, \nabla - 2v_{n+1} + v_n, \Delta - 2v_{n+1}, \nabla - 2v_{n+1}, \Delta - 2v_{n+1} - v_n, \nabla - 2v_{n+1} - v_n, \Delta - 2v_{n+1} - 2v_n, \nabla - 3v_{n+1} + 3v_n, \Delta - 3v_{n+1} + 2v_n, \nabla - 3v_{n+1} + 2v_n, \Delta - 3v_{n+1} + v_n, \nabla - 3v_{n+1} + v_n, \Delta - 3v_{n+1}, \nabla - 3v_{n+1}, \Delta - 3v_{n+1} - v_n, \nabla - 3v_{n+1} - v_n, \Delta - 3v_{n+1} - 2v_n, \nabla - 4v_{n+1} + v_n, \Delta - 4v_{n+1}, \nabla - 4v_{n+1}$.

Proof. We can write T_{n+5} in terms of T_n, v_{n+1}, v_n as

$$\begin{aligned} T_{n+5} = {} & (T_n - 4v_{n+1}) \cup (T_n - 4v_{n+1} + v_n) \cup (T_n - 3v_{n+1} - 2v_n) \cup \\ & (T_n - 3v_{n+1} - v_n) \cup (T_n - 3v_{n+1}) \cup (T_n - 3v_{n+1} + v_n) \cup \\ & (T_n - 3v_{n+1} + 2v_n) \cup (T_n - 3v_{n+1} + 3v_n) \cup (T_n - 2v_{n+1} - 2v_n) \cup \\ & (T_n - 2v_{n+1} - v_n) \cup (T_n - 2v_{n+1}) \cup (T_n - 2v_{n+1} + v_n) \cup \\ & (T_n - 2v_{n+1} + 2v_n) \cup (T_n - 2v_{n+1} + 3v_n) \cup (T_n - v_{n+1} - 2v_n) \cup \\ & (T_n - v_{n+1} - v_n) \cup (T_n - v_{n+1}) \cup (T_n - v_{n+1} + v_n) \cup \\ & (T_n - 4v_n) \cup (T_n - 3v_n) \cup (T_n - 2v_n) \cup (T_n - v_n) \cup \\ & T_n \cup (T_n + v_n) \cup (T_n + v_{n+1} - 4v_n) \cup (T_n + v_{n+1} - 3v_n) \cup \\ & (T_n + v_{n+1} - 2v_n) \cup (T_n + v_{n+1} - v_n) \cup (T_n + v_{n+1}) \cup \\ & (T_n + v_{n+1} + v_n) \cup (T_n + 2v_{n+1} - 2v_n) \cup (T_n + 2v_{n+1} - v_n). \end{aligned} \tag{4.29}$$

This gives T_{n+5} in terms of 32 translated tiles of T_n (shown in Fig. 4.19). Let A be a 32×32 adjacency matrix that indicates if the 32 tiles T_n of T_{n+5} are separated by $\pm 2v_{n-1}, \pm v_n, \pm v_{n+1}$. This can be constructed by inspection of T_{n+5}, which is shown in Fig. 4.20b. The rows and columns indicate each tile of T_n in the order of equation (4.29), dark pixels indicate a value of 1 denoting connectivity.

The trace of A^3 gives the number of paths of length 3 from each vertex to itself. Each triangle has 3 vertices, and the path is counted twice, once in each direction. Hence, the number of triangles in this graph is given by

$$\frac{1}{6}\mathrm{tr}(A^3) = 40$$

and the statement of the theorem follows. □

Theorem 16. *All triangles with corner points $\pm 2v_{n-1}, \pm v_n, \pm v_{n+1}$ have area 2^{n-1}.*

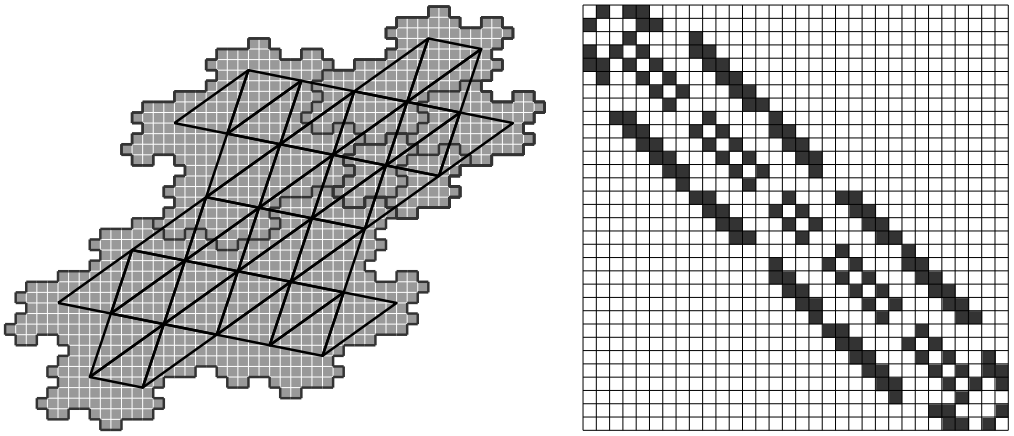
Proof. The triangle with side lengths $2v_{n-1}$ and v_n from $(0,0)$ has a third side of length $v_n - 2v_{n-1} = v_{n+1}$. The triangle with side lengths v_n and v_{n+1} has

a third side of length $v_{n+1} - v_n = 2v_{n-1}$. The triangle with side lengths v_{n+1} and $-2v_{n-1}$ has a third side of length $v_{n+1} + 2v_{n-1} = v_n$. The 3 triangles made up of the negative angles will have the same areas. Therefore all 6 triangles in the hexagon with corner points $\pm 2v_{n-1}, \pm v_n, \pm v_{n+1}$ have the same area. Computing one of these areas gives

$$\frac{1}{2}\left|2v_{n-1} \times v_n\right| = p_{n-1}q_n - p_n q_{n-1} = 2^{n-1}.$$

Therefore triangles constructed by joining lines between neighbouring tiles have area 2^{n-1}. □

Hence, the total area for the triangles in Theorem 15 is $40 \cdot 2^{n-1}$, as shown for T_{10} in Fig. 4.20a. Figure 4.20b gives the adjacency matrix of T_n tiles in T_{n+5} connected by the directions $\pm 2v_{n-1}, \pm v_n, \pm v_{n+1}$. The rows and columns are listed in the same order as T_{n+5} in (4.29). The white squares represent zero and dark squares are one.



a) b)

Figure 4.20: a) T_{10} with the unique area formed by 40 triangles. b) 32×32 adjacency matrix, A, for the 32 tiles T_n connected by $\pm 2v_{n-1}, \pm v_n, \pm v_{n+1}$, represented by each row/column.

4.6.2 Internal Area as a Linear Transformation

In this section we analyse the internal structure of maximal, and consequently, boundary ghosts. A maximal ghost domain has a fractal-like structure and it may be complicated to compute the inner part of a boundary ghost domain in order to know the points where the image value is unique. However, the nature of the linear transforms presented in Theorem 17c) enables us to use simpler ghost domains to identify central regions in which all the integer points have unique image values. First we describe the linear structure and then we illustrate the method with an example.

Consider again the case that $v_n = v_{n-1} - 2v_{n-2}$ for $n = 3, 4, \ldots$ and $v_0 = (1,0), v_1 = (1,1), v_2 = (-1,1)$. Other cases require natural adjustments. We observe that, by definition,

$$T_n = \{\varepsilon_0 v_0 + \cdots + \varepsilon_{n-1} v_{n-1} : \varepsilon_0, \ldots, \varepsilon_{n-1} \in \{0,1\}\}.$$

We use the fact that every point in T_n can be written as a linear combination of $v_0, \ldots, v_{n-1}$ to prove the following theorem.

Theorem 17. *Let $f : \mathbb{R}^2 \to \mathbb{R}^2$ be defined by $f(x,y) = (x - 2y, y)$. Then for all positive integers m, n and real numbers x, y we have:*
a) $f^m(v_n) = v_{n+m}$,
b) $T_{m+n} = f^m(T_n) + T_m$,
c) $f^m(x,y) = xv_m - 2yv_{m-1}$.

Proof. a) It suffices to prove the theorem for $m = 1$. We use induction on n. Observe that $f(v_1) = f(1,1) = (-1,1) = v_2$. For $n = 2, 3, \ldots$ we have

$$\begin{aligned} f(v_n) &= f(v_{n-1} - 2v_{n-2}) \\ &= f(v_{n-1}) - 2f(v_{n-2}) \\ &= v_n - 2v_{n-1} \\ &= f(v_{n+1}). \end{aligned}$$

b) By a) we have, with all choices $\varepsilon_j \in \{0,1\}$ for $j = 0, 1, \cdots, m+n-1$,

$$\begin{aligned} T_{m+n} &= \{\varepsilon_0 v_0 + \cdots + \varepsilon_{m-1} v_{m-1} + \varepsilon_m f^m(v_0) + \cdots + \varepsilon_{m+n-1} f^m(v_{n-1})\} \\ &= T_m + f^m(\{\varepsilon_m v_0 + \cdots + \varepsilon_{m+n-1} v_{n-1}\}) \\ &= T_m + f^m(T_n). \end{aligned}$$

c) By induction on m. We have

$$f(x,y) = (x - 2y, x) = x(1,1) - 2y(1,0) = xv_1 - 2yv_0$$

and

$$\begin{aligned} f^{m+1}(x,y) &= f(f^m(x,y)) \\ &= f(xv_m - 2yv_{m-1}) \\ &= xf(v_m) - 2yf(v_{m-1}) \\ &= xv_{m+1} - 2yv_m. \end{aligned}$$

□

To demonstrate the use of this theorem consider the case $n = 5$. A simple calculation yields (Fig. 4.21) $T_5 = \{(0,0),\ (1,0),\ (1,1),\ (2,1),\ (-1,1),\ (0,1),\ (0,2),\ (1,2),\ (-3,-1),\ (-2,-1),\ (-2,0),\ (-1,0),\ (-4,0),\ (-3,0),\ (-3,1),\ (-2,1),\ (-1,-3),\ (0,-3),\ (0,-2),\ (1,-2),\ (-2,-2),\ (-1,-2),\ (-1,-1),$

$(0,-1)$, $(-4,-4)$, $(-3,-4)$, $(-3,-3)$, $(-2,-3)$, $(-5,-3)$, $(-4,-3)$, $(-4,-2)$, $(-3,-2)\}$. Observe that the full triangles with vertices $v_1 - 4v_2$, $-4v_1 + v_2$, $v_1 + v_2$ and with vertices $-2v_1 - 3v_2$, $3v_1 - 3v_2$, $-2v_1 + 2v_2$ together cover 28 points of T_5 and no integer points outside T_5. For any positive integer m we have $T_{m+5} = f^m(T_5) + T_m$. The 32 points $f^m(T_5)$ are given by

$$\{\varepsilon_0 v_m + \cdots + \varepsilon_4 v_{m+4} : \varepsilon_0, \ldots \varepsilon_4 \in \{0,1\}\}.$$

We obtain T_{m+n} by replacing each such a point p by $p + T_m$. The full triangles with vertices

$$v_{m+1} - 4v_{m+2}, -4v_{m+1} + v_{m+2}, v_{m+1} + v_{m+2}$$

and with vertices

$$-2v_{m+1} - 3v_{m+2}, 3v_{m+1} - 3v_{m+2}, -2v_{m+1} + 2v_{m+2}$$

cover together 28 of such points p. We can show that for $m \geq 5$ all the integer points in these full triangles are points in the interior of boundary ghost domain T^*_{m+n} and have therefore unique image values.

Example 18. (See Fig. 4.21.) Take $m = n = 5$. We have $T_{10} = f^5(T_5) + T_5$. The 32 points $f_5(T_5)$ are the large points in Fig. 4.21 and lie in the lattice $\mathbb{Z}(5,-1)+\mathbb{Z}(7,5)$. We obtain T_{10} by replacing every large point p by $p+T_5$. All the integer points in the full triangles with vertices $v_6 - 4v_7, -4v_6 + v_7, v_6 + v_7$ and with vertices $-2v_6 - 3v_7, 3v_6 - 3v_7, -2v_6 + 2v_7$ belong to T_{10} and are in the interior of T^*_{10}. Thus they have unique image values.

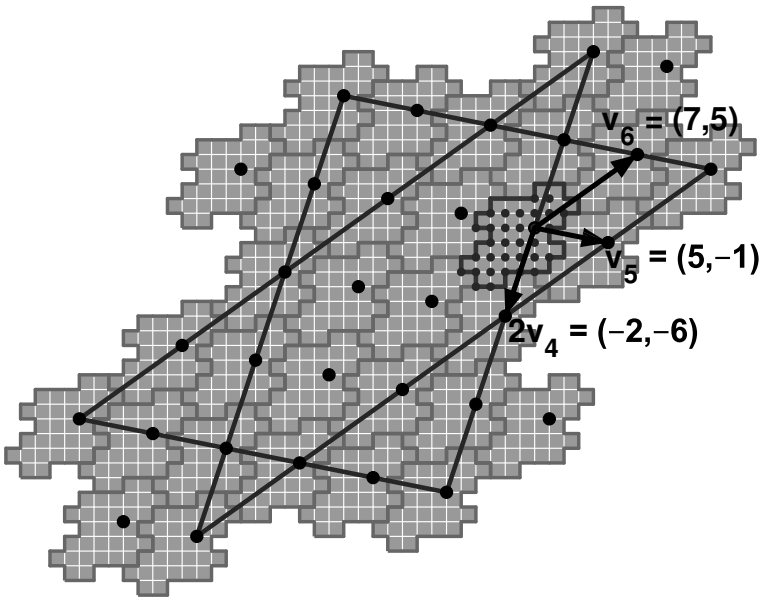
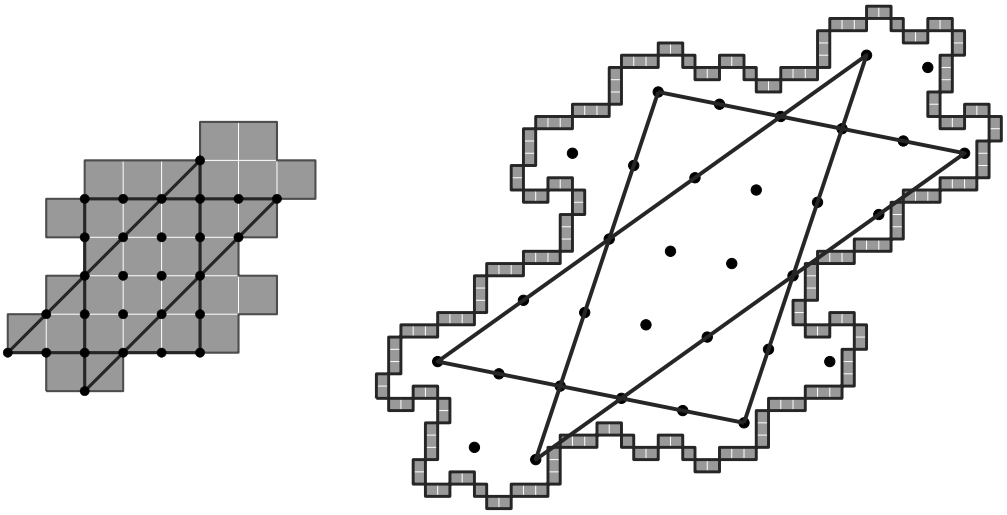


Figure 4.21: U_{10} as 32 translated tiles of U_5.

The consequence of this result for the boundary ghost can be seen in Fig. 4.22b. The union of the two triangles provides a lower bound for the central uniquely determined area of boundary ghosts for $n \geq 10$. In practice, one can quickly verify if an object under discrete reconstruction is contained within this area, and is thereby uniquely determinable.



a) b)

Figure 4.22: a) U_5 tile. b) The uniquely determinable area within the boundary ghost domain U_{10}^*.

We can then use Pick's theorem to determine the exact number of points in this region that can be reconstructed uniquely.

Theorem 19. *T_n^* contains a convex region of at least $25 \cdot 2^{n-6} + 11$ uniquely determinable points for $n \geq 10$.*

Proof. The triangles with vertices $v_5 - 4v_6, -4v_5 + v_6, v_5 + v_6$ and vertices $-2v_5 - 3v_6, 3v_5 - 3v_6, -2v_5 + 2v_6$ both have area

$$\begin{aligned} A &= \frac{1}{2}|5v_{n-5} \times 5v_{n-4}| \\ &= \frac{25}{2}|p_{n-5}q_{n-4} \times p_{n-4}q_{n-5}| \\ &= 25 \cdot 2^{n-6} \end{aligned} \tag{4.30}$$

by Lemma 1. Either triangle has side lengths $5v_{n-4}$, $5v_{n-5}$ and $10v_{n-6}$. As (p_n, q_n) are co-prime for all n, the triangle must have $b = 20$ integer points that lie exactly on the boundary. By Pick's theorem, the number of interior points must be

$$i = A - \frac{b}{2} + 1 \tag{4.31}$$

$$= 25 \cdot 2^{n-6} - 9.$$

Therefore, the total number of integer points contained within the convex region is $25 \cdot 2^{n-6} + 11$. □

By the same process, we give a similar result for the union of both triangles.

Corollary 20. *The union of triangles with vertices* $v_5 - 4v_6, -4v_5 + v_6, v_5 + v_6$ *and vertices* $-2v_5 - 3v_6, 3v_5 - 3v_6, -2v_5 + 2v_6$ *contain* $17 \cdot 2^{n-5} + 14$ *unique points in* T_n^* *for* $n \geq 10$.

Depending on the application, different aspects of the uniquely determinable inner region may be relevant. In this section, we have provided a variety of measures for this internal region which can be computed efficiently.

4.7 Summary

We have presented a method for constructing discrete maximal ghosts for N directions that consist of 2^N distinct, 4-connected points with ghost values ± 1. Ghosts constructed from discrete lattice directions defined by the recursion $v_{n+1} = v_n + 2\epsilon_{n+1}v_{n-1}$ (where $\epsilon_{n+1} \in \{-1, 1\}$) tile the lattice. If ϵ_n is the same for all n greater than some n_0, then the recursion can be viewed as generating two Lucas sequences, with different initial values for the x and y directions [11]. The set T_n formed by this recursion is surrounded by six neighbours: $T_n \pm 2v_{n-1},\ T_n \pm v_{n-1} \pm 2v_{n-2}$. These neighbours are all contained in T_{n+5}.

Due to the internal structure of maximal ghosts, all non-boundary points can be removed with an additional lattice direction to create boundary ghosts. When we restrict to $\epsilon_j = -1$ for $j = 3, 4, \ldots n-1$, the number of points of the boundary ghost T_n^* is $c\rho^n + O(|\sigma|^n)$ with $\sigma < \rho < 2$. This is exponentially smaller than the size of its interior. The internal structure of ghosts can be determined quickly through the linear mapping of triangles, which gives a central uniquely determined area for boundary ghosts T_n^* with $n \geq 10$.

Figure 4.23 shows reconstructions of an image with insufficient projection data from the sets of projections that define the maximal ghost T_{14} ($\sum |p_n| = 210$ and $\sum |q_n| = 119$) and the resultant boundary ghost domain T_{14}^*. The reconstructions are $P \times Q$ images that have the same horizontal and vertical size as the ghosts, and therefore there is only $(P - \sum |p_n|)(Q - \sum |q_n|) = 1$ unknown variable in the linear system of discrete projections. The white pixels mark the points which cannot be uniquely recovered. The maximal ghost in Fig. 4.23b completely distorts the important information, while in Fig. 4.23c the boundary ghost strongly reduces the number of these errors and restricts their location to less crucial parts of the image. Figure 4.23d uses the same projection set that defines the boundary ghost domain T_{14}^*, but for a 211×131 image such that $(P - \sum |p_n|)(Q - \sum |q_n|) = 10$. The shape of the errors is the same, however the boundary of errors becomes thicker. This illustrates how

the careful selection of projection directions suggested in this work can either obfuscate the important information in a reconstruction or leave it mostly untouched.

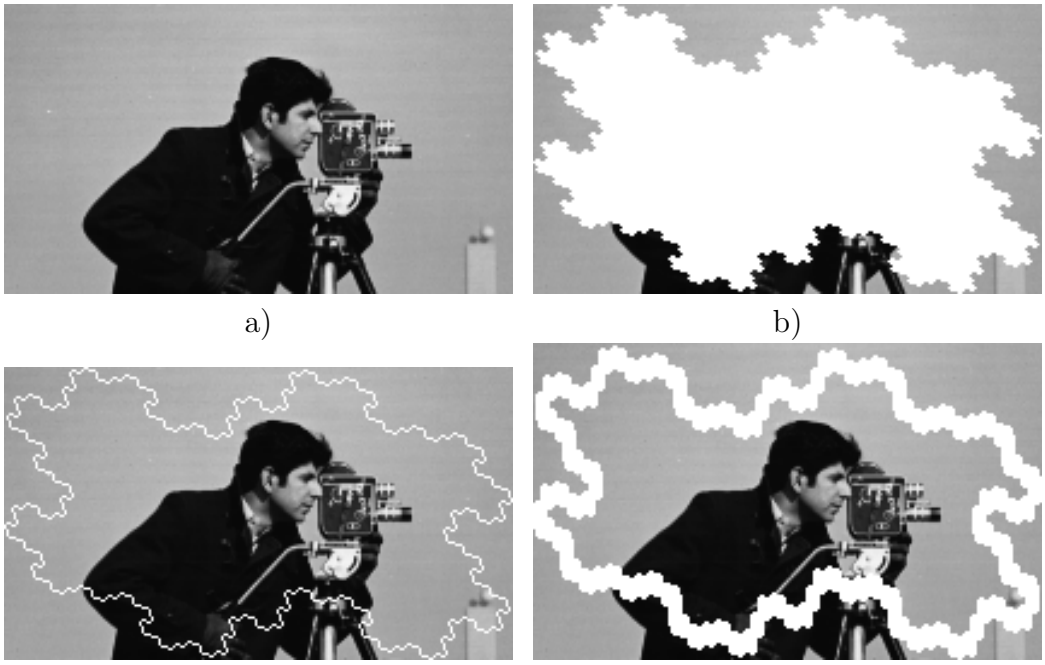


Figure 4.23: Reconstruction of the cameraman test image (a) where white pixels mark non-unique image locations. b) 211×120 image with the maximal ghost T_{14} projection set. c) 211×121 image with the boundary ghost T_{14}^* projection set. d) 211×131 image with the boundary ghost T_{14}^* projection set.

Boundary ghosts can be constructed by choosing recursive projection directions $v_{n+1} = v_n \pm 2v_{n-1}$. This choice of ϵ_n at each step allows for many different possible boundary ghosts to be constructed, with a vast range of shapes and sizes. The following chapter will look to extend these results to higher dimensions.

CHAPTER 5

Ghosts in Higher Dimensions

Tomography most often finds application in three dimensions, as we are usually interested in the reconstruction of 3D objects. Therefore, it is useful to extend the formulation of maximal and boundary ghosts into the third dimension. As we have seen, due to the number theoretic aspect of discrete tomography, the field also finds application far beyond imaging. It can be used in the wider context of communications and encryption. In these areas, we are not limited to the three spacial dimensions of the real world. Hence, it may be of use to develop our ideas to higher dimensions. In this chapter, we build upon the work of Chapter 4 to construct maximal and boundary ghosts in any dimension. New degrees of freedom allow for a wider range of shapes and sizes that can be produced, and also different ways of removing the interior points which give rise to boundary ghosts.

Thus far, we have limited our focus to constructing ghosts in two dimensions. Given the theory developed in this chapter, it is natural to ask: Do similar ghost structures exist in higher dimensions? In this section, we use the 2D shapes as building blocks to construct maximal primitive ghosts in any dimension that maintain the same properties as in 2D. But first, we examine why this approach is necessary, as one may assume that a simple recursion similar to (4.6) can be used for higher dimensions.

Consider recursion (4.6) with $\epsilon_{n+1} = -1$ for $n = 2, 3, \ldots$, which generates a maximal ghost such that the n-th ghost domain T_n fits into a square with side length $3 \cdot 2^{n/2}$ for every n. It would seem to be likely that there exists a recurrence relation $v_n = c_1 v_{n-1} + c_2 v_{n-2} + c_3 v_{n-3}$ with suitable initial values such that a maximal ghost is generated which fits into a cube of side length

$c \cdot 2^{n/3}$ for every n, where c_1, c_2, c_3, c are integer constants. Here we show that such a recurrence does not exist.

The recurrence relation $v_n = c_1 v_{n-1} + c_2 v_{n-2} + c_3 v_{n-3}$ has as the characteristic polynomial $z^3 - c_1 z^2 - c_2 z - c_3 = \prod_{j=1}^{3}(z - \alpha_j)$. Since $c_1, c_2, c_3 \in \mathbb{Z}$, one of the roots, say α_1, should be real. Let $v_n = (p_n, q_n, r_n)$. We require that v_n, v_{n-1}, v_{n-2} are linearly independent over $\mathbb{Z}$ for all $n \geq 2$. Then p_n, q_n and r_n satisfy the recurrence relation $x_n = c_1 x_{n-1} + c_2 x_{n-2} + c_3 x_{n-3}$. It follows that for $j = 1, 2, 3$, it is not possible that $c_j = 0$ for all three sequences. In order for T_n to fit into the specified box, we conclude that $\max_{j=1,2,3} |\alpha_j| \leq 2^{1/3}$. On the other hand, the number of elements of T_n is 2^n and therefore requires that $|\alpha_1 \alpha_2 \alpha_3| = |c_3| \geq 2$. Thus $|\alpha_j| = 2^{1/3}$ for $j = 1, 2, 3$, and $\alpha_1 = \pm 2^{1/3}$. It follows that $z^3 - c_1 z^2 - c_2 z - c_3 = z^3 \pm 2$ and the recurrence reads $v_n = \pm 2 v_{n-3}$ for $n \geq 3$. We conclude that for $n \geq 3$ all coefficients p_n, q_n, r_n are even so that we don't obtain a sequence of relatively prime vectors.

Let us provide an example. A natural choice for a recursion with degree 3 is $v_n = v_{n-2} - 2v_{n-3}$ with initial vectors $v_0 = (1, 0, 0)$, $v_1 = (0, 1, 0)$ and $v_2 = (0, 0, 1)$. An illustration of a ghost generated using this recursion is shown in Fig. 5.1. This ghost is cylindrical in shape, and grows along a diagonal. Unlike the 2D ghosts, this shape has a fill factor that tends towards zero for a large number of directions. This shape is not so favourable, as it only covers a small diagonal part of a rectangular box.

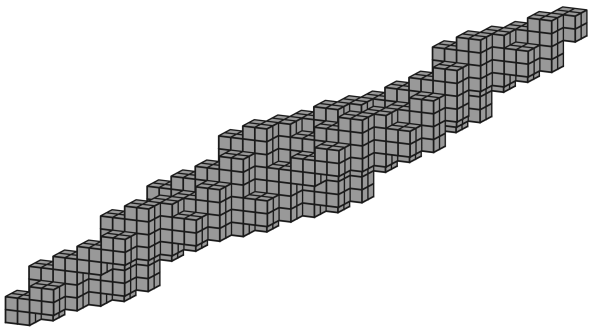


Figure 5.1: Maximal ghost in 3D with zero line sums over ten directions, given by $v_n = v_{n-2} - 2v_{n-3}$ with initial vectors $v_0 = (1, 0, 0)$, $v_1 = (0, 1, 0)$ and $v_2 = (0, 0, 1)$.

Hence, it is not possible to extend boundary ghosts to three dimensions simply using a recursion of degree 3. We may still generalise boundary ghosts to higher dimensions, but a different approach is required.

5.1 From 2D to 3D

We now look to construct 3D maximal primitive ghosts by using the 2D shapes as building blocks. These building blocks can be stacked along a third dimension to produce new ghost tiles. Just as with 2D ghosts, this can be done in many different ways to produce a variety of shapes. In practice, one may

construct a 2D ghost that fits the desired size in x and y, and then stack these tiles in z to the required height.

However here, we look to achieve a ghost that fits into a cube with side lengths $c \cdot 2^{n/3}$ for all n. Recall that the characteristic polynomials for the 2D recurrence are given by

$$\begin{aligned} p_n = q_{n+1} =& \frac{1}{\sqrt{-7}} \left(\frac{1}{2} + \frac{1}{2}\sqrt{-7} \right)^n - \frac{1}{\sqrt{-7}} \left(\frac{1}{2} - \frac{1}{2}\sqrt{-7} \right)^n \\ =& \frac{2}{\sqrt{7}} \mathrm{Im} \left(\frac{1}{2} + \frac{1}{2}\sqrt{-7} \right)^n \end{aligned} \tag{5.1}$$

and the size of the ghost is the absolute sum of the directions.

$$\left(1 + \sum_{j=1}^{n} |p_j|, 1 + \sum_{j=1}^{n} |q_j| \right) \tag{5.2}$$

The growth rates in the x and y direction are both of order $\mathcal{O}(2^{n/2})$, hence the size in both directions approximately doubles every two directions. For each direction that stacks 2D ghosts in the z-direction, the height of the ghost doubles, therefore the step size r_i must also double. To ensure all sides have the same rate of growth in 3D, a step along the z-direction should be added every three directions, thereby ensuring that the size in x, y and z all approximately double every three steps. The p_i and q_i components should be minimal in these steps to maximise the fill factor. By Theorem 4, we know that the set T_n becomes a ghost when we give value 1 to points $(x, y) \in T_n$ with even x and value -1 to those with odd x, which is maintained due to the fact that p_n is always odd. Hence, in 3D we define lattice directions recursively by

$$\begin{aligned} v_{3n-1} &= ((-1)^{n-1}, 0, (-2)^{n-1}), \\ v_{3n} &= v_{3n-2} - 2v_{3n-3}, \\ v_{3n+1} &= v_{3n} - 2v_{3n-2} \end{aligned} \tag{5.3}$$

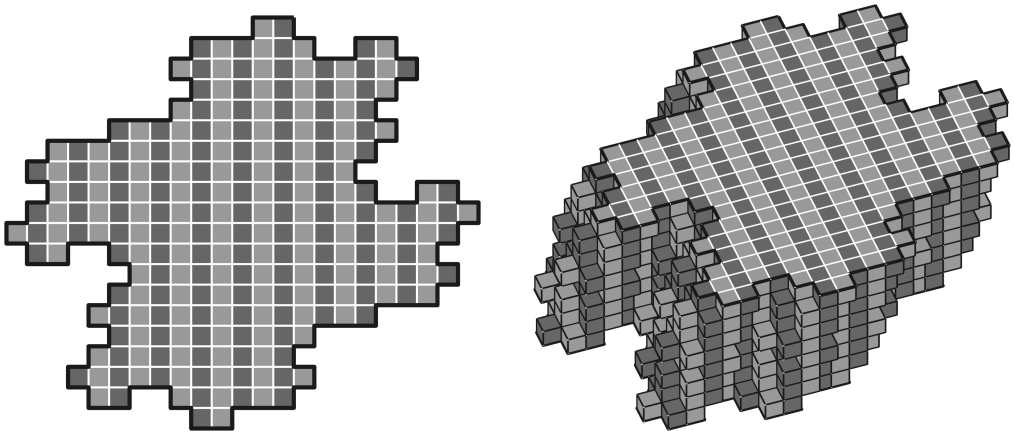
for $n = 1, 2, \ldots$, with initial vectors $v_0 = (1, 0, 0), v_1 = (1, 1, 0)$. We similarly define the set of ghost points by $T_{n+1} = T_n \cup (T_n + v_n)$ for $n = 0, 1, \ldots$, with $T_0 = \{(0, 0, 0)\}$. We now consider voxels in 3D space, so we define

$$U_n = T_n + \{(x, y, z) \in \mathbb{R}^3 : 0 \leq x \leq 1, 0 \leq y \leq 1, 0 \leq z \leq 1\} \tag{5.4}$$

for $n = 1, 2, \ldots$. The recursions v_{3n} and v_{3n+1} simply give the 2D directions. The justification for the choice of p_{3n-1} and q_{3n-1} is that they cannot be zero, as then we would have degenerate angles. So we pick the smallest shift, 1, and add it to p_{3n-1} so that all even ghost points in x have value 1 and odd x points have value -1. The results from 2D all follow for each x-y slice in 3D, as we shall demonstrate in the forthcoming sections.

Ghosts constructed from U_8 in 2D, and correspondingly U_{11} in 3D, are shown in Fig. 5.2. Dark squares show where the ghost has value -1, and light

squares are +1 ghost values. These values have been indicated to demonstrate how the structure has been extended from 2D to 3D. Notice that in both cases, ghost values alternate sign along the x direction. For the remainder of this chapter, ± 1 values will not be indicated, as all ghosts have the same structure and our focus will be on the shape of the tiles.



a) b)

Figure 5.2: a) Maximal ghost U_8 in 2D. b) Maximal ghost U_{11} in 3D, constructed as a stack of the 2D ghost in (a). Light squares denote +1 and dark squares are -1.

5.2 Connectedness of the Tiles

Let $w_1, w_2, w_3, w_4 \in \mathbb{Z}^3$. We call the points w_1 and w_2 6-connected if $|w_1 - w_2| = 1$. We say that the disjoint tiles $T_n + w_3$ and $T_n + w_4$ are neighbours if there are points $w_1 \in T_n + w_3, w_2 \in T_n + w_4$ such that w_1 and w_2 are 6-connected. If so, then $U_n + w_3$ and $U_n + w_4$ have a face in common. We aim to generalise Theorem 5 and Corollary 7. First we describe which tiles are 6-connected with T_n and thereafter we show that all neighbouring tiles of T_n are in T_{n+8}.

Theorem 21. *For $n = 1, 2, \ldots$ we have:*

T_{3n} *is surrounded by* $T_{3n} \pm V$ *where* $V = v_{3n-2} \pm v_{3n-3}, 2v_{3n-2}, (0, n, 2^n)$,
T_{3n+1} *is surrounded by* $T_{3n+1} \pm V$ *where* $V = v_{3n} \pm v_{3n-2}, 2v_{3n}, (0, n, 2^n)$,
T_{3n+2} *is surrounded by* $T_{3n+2} \pm V$ *where* $V = v_{3n+1} \pm v_{3n}, 2v_{3n+1}, (0, n, 2^n)$.

Proof. By Theorem 5, we can immediately see that in the x-y plane,

$$T_{3n} \text{ is surrounded by } T_{3n} \pm v_{3n-2} \pm v_{3n-3}, \pm 2v_{3n-2},$$
$$T_{3n+1} \text{ is surrounded by } T_{3n+1} \pm v_{3n} \pm v_{3n-2}, \pm 2v_{3n},$$
$$T_{3n+2} \text{ is surrounded by } T_{3n+2} \pm v_{3n+1} \pm v_{3n}, \pm 2v_{3n+1}.$$

The separation between the bottom and top 2D ghost plane is given by

$$\sum_{h=1}^{n} |v_{3h}| = \sum_{h=1}^{n} \left(1, 0, 2^{h-1}\right) = (n, 0, 2^n - 1)$$

and therefore in all three cases, T_{3n+m} is a neighbour of $T_{3n+m} \pm (n, 0, 2^n)$ for $m = 0, 1, 2$, and the statement of the theorem follows. □

As a result of Theorem 21, we are again free to arbitrarily choose neighbouring directions to construct a ghost to fit a specified shape. The theorem is illustrated in Fig. 5.3 for U_8, represented as the dark tile. Only half of the neighbours are visualised here to show how each neighbour connects to the central tile. The other neighbours are symmetric.

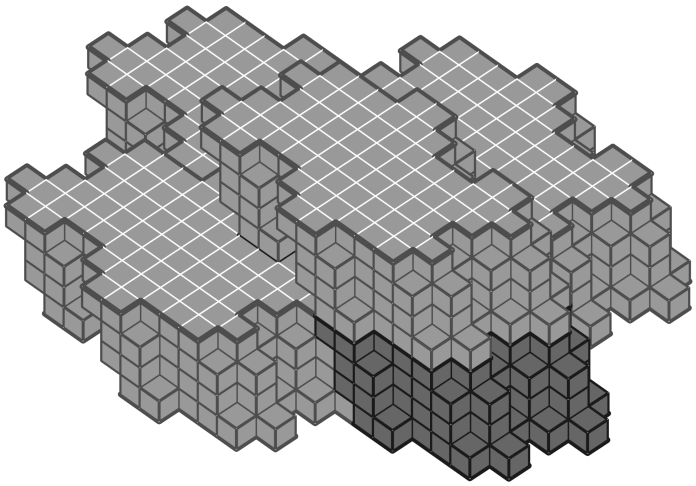


Figure 5.3: U_8 (dark tile) and its neighbours U_8+V where $V = \{(7, 5, 0), (3, -7, 0), (10, -2, 0), (2, 0, 4)\}$. Negative neighbours $(-V)$ are not illustrated.

Theorem 22. *For every n, all neighbours of T_n are contained in T_{n+8}.*

Proof. We consider T_{3n+m} for $m = 0, 1, 2$. Let

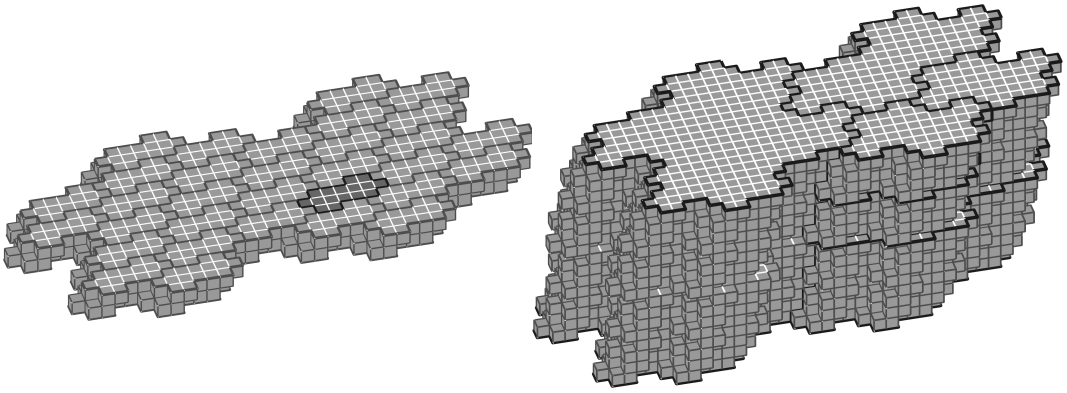
$$\begin{aligned}
\Gamma_{3n} &= \{T_{3n} + \varepsilon_1 v_{3n} + \varepsilon_2 v_{3n+1} + \varepsilon_3 v_{3n+3} + \varepsilon_4 v_{3n+4} + \varepsilon_5 v_{3n+6}\}, \\
\Gamma_{3n+1} &= \{T_{3n+1} + \varepsilon_1 v_{3n+1} + \varepsilon_2 v_{3n+3} + \varepsilon_3 v_{3n+4} + \varepsilon_4 v_{3n+6} + \varepsilon_5 v_{3n+7}\}, \\
\Gamma_{3n+2} &= \{T_{3n+2} + \varepsilon_1 v_{3n+3} + \varepsilon_2 v_{3n+4} + \varepsilon_3 v_{3n+6} + \varepsilon_4 v_{3n+7} + \varepsilon_5 v_{3n+9}\}.
\end{aligned}$$

such that $\varepsilon_1, \ldots, \varepsilon_5 \in \{0, 1\}$ for each set. By Theorem 5, T_{3n} and its neighbours in the x-y plane $T_{3n} \pm v_{3n-2} \pm v_{3n-3}$, $T_{3n} \pm 2v_{3n-2}$ are contained in Γ_{3n}. $T_{3n} \pm (0, n, 2^n)$ are contained within $\Gamma_{3n} + v_{3n+2}$ and $\Gamma_{3n} + v_{3n+5}$, therefore T_{3n} all the neighbours are contained in T_{3n+7}.

Similarly, T_{3n+1} and its horizontal neighbours are contained in Γ_{3n+1}, and T_{3n+2} and its horizontal neighbours are contained in Γ_{3n+2}. Then $T_{3n+1} \pm (0, n, 2^n)$ are contained within $\Gamma_{3n+1} + v_{3n+2}$ and $\Gamma_{3n+1} + v_{3n+5}$, and $T_{3n+2} \pm$

$(0, n, 2^n)$ are contained within $\Gamma_{3n+2} + v_{3n+2}$ and $\Gamma_{3n+2} + v_{3n+5}$. Therefore, all the neighbours of T_{3n+1} are contained in T_{3n+8}, and all the neighbours of T_{3n+2} are contained in T_{3n+10}. Thus for all n, T_n and all its neighbours are contained in T_{n+8}. □

Theorem 22 is illustrated in Fig. 5.4, where U_5 is contained within U_{13}. The structure of the tile Γ_5 (Fig. 5.4a) is reminiscent of the 2D structure shown in Fig. 4.19.



a) b)

Figure 5.4: a) Tile Γ_5, with U_5 highlighted. b) Ghost U_{13}, which contains U_5 and all its neighbours.

5.3 3D Boundary Ghosts

We investigate so-called boundary ghosts generated by the previously constructed maximal ghosts. Boundary ghost domains T_n^* consist of boundary points of T_n. They are constructed using the same recursive sequences of directions, but with an extra direction $(0, 1, 0), (0, 0, 1)$, or a linear combination of these directions. We shall deal with the case that we add the direction $v_{-1} = (0, 1, 0)$.

Recall that the ghost value of $(x, y, z) \in T_n$ is 1 if x is even and -1 if x is odd. All other points in $\mathbb{Z}^3$ have value 0. Let T_n^* be the ghost by adjoining $(0, 1, 0)$ to $v_0, \ldots, v_{n-1}$. Then the value at (x, y, z) becomes 0 if (x, y, z) and $(x, y-1, z)$ both belong to T_n or both do not belong to T_n, and becomes ± 1 if exactly one of both belongs to T_n. Let U_n^* be the set of voxels corresponding to T_n^* in the same way as U_n corresponds to T_n. We prove the following theorem.

Theorem 23. *There is a one-to-one correspondence between the boundary squares of area 1 of U_n in the direction of the y-axis and the elements of T_n^*.*

Proof. Observe that the ghost value of (x, y, z) in T_n^* is the value of (x, y, z) minus the value of $(x, y-1, z)$ in T_n. If there is a boundary square between

the voxels of (x, y, z) and $(x, y-1, z)$ in U_n, then one among (x, y, z) and $(x, y-1, z)$ belongs to T_n and the other does not. Therefore $(x, y, z) \in T_n^*$.

If $(x, y, z) \in T_n^*$, then the ghost values of $(x, y-1, z)$ and (x, y, z) in T_n differ by 1. That is, one value is ± 1 and the other value is 0. It follows that the square between the voxels in U_n corresponding to these points is a boundary square of U_n. □

Half of the points of T_n^* belong to T_n and half of them do not. A point of T_n can generate at most two points of T_n^*, that is, if its neighbours in the y-direction both do not belong to T_n. Given the theory we have developed in Section 4.5, we can count the number of boundary points of T_n^*.

Theorem 24. *Let $\rho \approx 1.5214$ be the positive zero of $x^3 - x - 2$ and σ and $\overline{\sigma}$ the nonreal zeros. Then the number of points of T_n^* in three dimensions equals*

$$2^{\lfloor n/3 \rfloor} \left[c\rho^{\eta} + 2|\sigma|^{\eta} (\mathrm{Re}(c_1) \cos(\eta \arg(\sigma)) - \mathrm{Im}(c_1) \sin(\eta \arg(\sigma))) \right]$$

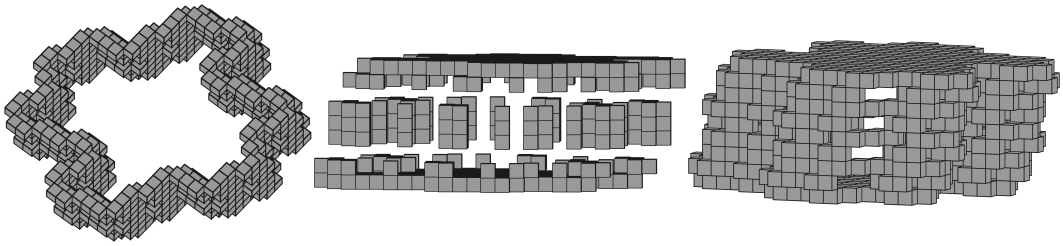
with $\eta = \lfloor 2(n+1)/3 \rfloor$, $\rho = 1.5214$, $\sigma = -0.7607 + 0.8579i$, $c = 2.4757$, and $c_1 = -0.2378 + 0.0747i$.

Proof. In three dimensions, the ghost domain T_n is made up of $2^{\lfloor n/3 \rfloor}$ tiles of the ghost domain $T_{\lfloor 2(n+1)/3 \rfloor}$ in two dimensions, as the stack doubles every three directions. Recall from Theorem 13, $\delta_n = \delta_{n-2} + 2\delta_{n-3}$ with initial values $\delta_1 = 4, \delta_2 = 6, \delta_3 = 8$. Therefore, the number of ghost points in three dimensions is given by $2^{\lfloor n/3 \rfloor} \delta_{\lfloor 2(n+1)/3 \rfloor}$ and the statement of the theorem follows. □

Corollary 25. *In three dimensions T_n^* has $2^{\lfloor n/3 \rfloor}(c\rho^{\lfloor 2(n+1)/3 \rfloor} + O(|\sigma|^{\lfloor 2(n+1)/3 \rfloor}))$ elements.*

In choosing $v_{-1} = (0, 0, 1)$ or $v_{-1} = (0, 1, 1)$, the boundary ghost takes the shape of a sphere with considerably more points, as shown in Fig. 5.5. In this case of $v_{-1} = (0, 0, 1)$, there is a solid 2D ghost on the top and bottom of the shape, with some ghost points on the sides. Using $v_{-1} = (0, 1, 1)$, results in a similar shape, but with more ghost points on the sides. There are still some holes in the side walls, which can be seen in Fig. 5.5c. However, all of these shapes have no ghost points within their surfaces. The maximal ghost T_{11} has $2^{11} = 2048$ ghost points, while the boundary ghosts with directions $v_{-1} = (0, 1, 0)$, $v_{-1} = (0, 0, 1)$ and $v_{-1} = (0, 1, 1)$ have 560, 752 and 1032 ghost points respectively.

It is worth noting that these choices of v_{-1} will have a similar effect on any 3D ghost with this structure. Therefore, one could construct stacks from a variety of 2D ghost shapes, and create boundary ghosts in the same way.



a) b) c)

Figure 5.5: Boundary ghost U_{11}^* with: a) $v_{-1} = (0, 1, 0)$, b) $v_{-1} = (0, 0, 1)$, c) $v_{-1} = (0, 1, 1)$.

5.4 Alternate Sequences

In this section, we show other example of sequences that produce 3-dimensional ghosts. As demonstrated, ghosts in 3D are formed by stacking 2D ghosts in a direction that is normal to their surface. Up to this point, all examples have shown stacking along the z-axis. However, this can be performed along any 2D plane in 3D, provided the directions have co-prime components. Consider the following recursion:

$$\begin{aligned} v_{3n+1} &= v_{3n-1} + 2v_{3n-2}, \\ v_{3n+2} &= v_{3n} - 2v_{3n-1}, \\ v_{3n+3} &= v_{3n+2} - 2v_{3n}. \end{aligned} \tag{5.5}$$

In Table 5.1 we give two sets of 11 directions generated with the recursions in (5.5). Two different sets of initial conditions are used for these directions. In Table 5.1a we have $v_0 = (1, 0, 0)$, $v_1 = (0, 1, 0)$, $v_2 = (0, 0, 1)$, which generates a set of directions reminiscent of the sequence used throughout this work, but stacked along the z direction. Table 5.1b has a set of directions generated from the initial directions $v_0 = (1, 0, 0)$, $v_1 = (1, 1, 0)$, $v_2 = (1, 0, 1)$. Looking at the directions, it is not obvious that this set also produces a 3D ghost from a 2D ghost stack. However, here the stacking occurs along a diagonal direction, which can be seen in Fig. 5.6.

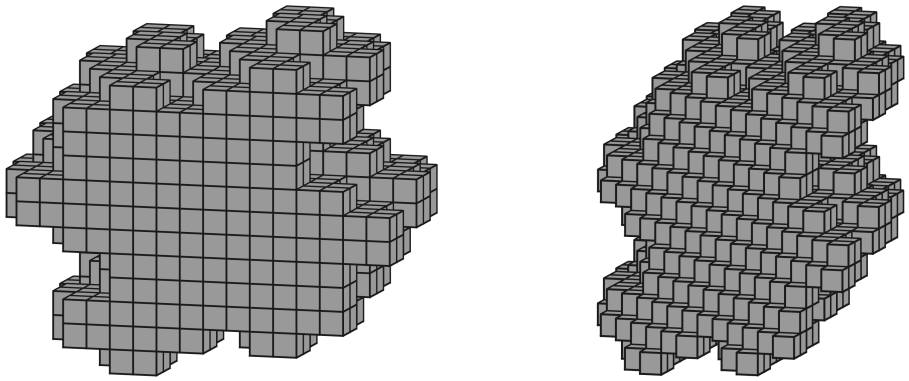
Though the link between these two examples is visually apparent, computationally it has been observed that all 3D ghosts which have the same order of growth along all dimensions can be viewed as stacks of 2D ghosts. This allows us to generalise the construction method to any number of dimensions.

5.5 Ghosts in k-Dimensions

Using the construction outlined for threedimensions, we can generate ghosts in higher dimensions. For a maximal primitive ghost in dimension k, we define

Table 5.1: Sets of directions from the recursions (5.5) with initial values: a) $v_0 = (1,0,0)$, $v_1 = (0,1,0)$, $v_2 = (0,0,1)$, b) $v_0 = (1,0,0)$, $v_1 = (1,1,0)$, $v_2 = (1,0,1)$.

a)	b)
(1, 0, 0)	(1, 0, 0)
(0, 1, 0)	(1, 1, 0)
(0, 0, 1)	(1, 0, 1)
(2, 1, 0)	(3, 1, 0)
(0, −2, 1)	(−1, −2, 1)
(0, −2, −1)	(−3, −2, −1)
(4, 0, 1)	(5, 0, 1)
(0, 2, −3)	(−1, 2, −3)
(0, 6, −1)	(5, 6, −1)
(8, 2, −1)	(9, 2, −1)
(0, 2, 5)	(7, 2, 5)



a) b)

Figure 5.6: Two alternate constructions of U_{11} with the initial values: a) $v_0 = (1,0,0)$, $v_1 = (0,1,0)$, $v_2 = (0,0,1)$, b) $v_0 = (1,0,0)$, $v_1 = (1,1,0)$, $v_2 = (1,0,1)$.

the lattice directions by

$$\begin{aligned} v_{kn-k+2} &= ((-1)^{n-1}, 0, (-2)^{n-1}, 0, \ldots, 0), \\ v_{kn-k+3} &= ((-1)^{n-1}, 0, 0, (-2)^{n-1}, \ldots, 0), \\ &\vdots \\ v_{kn-1} &= ((-1)^{n-1}, 0, \ldots, 0, (-2)^{n-1}), \\ v_{kn} &= v_{k(n-1)+1} - 2v_{k(n-1)}, \\ v_{kn+1} &= v_{kn} - 2v_{k(n-1)+1}. \end{aligned} \tag{5.6}$$

for $n = 1, 2, \ldots$ with initial vectors $v_0 = (1,0,0,\ldots), v_1 = (1,1,0,\ldots)$. The ghost domain T_n then has a similar structure to that outlined in the 3D case. We take two steps in the x-y plane, as in the 2D case, followed by a step of size 2^{n-1} along all other dimensions to ensure each side of the k-dimensional hyperrectangle has approximately even side lengths. Again, a shift of 1 in the

x direction is also present in this step to both ensure that the direction has co-prime components, and to maintain the structure of the ghost having value 1 for all points $(x, y) \in T_n$ with even x and value -1 to those with odd x.

The same connectivity of the tiles also follows through higher dimensions. A maximal ghost with k-dimensions will have $2(k+1)$ neighbours. There are six neighbours in the x-y plane by Theorem 5, and then two neighbours along each additional dimensions, given by

$$T_n \pm \{(n, 0, 2^n, \dots, 0), \dots, (n, 0, 0, \dots, 2^n)\}.$$

Hence, T_n with k dimensions has $2(k-2)+6 = 2(k+1)$ neighbours.

A k-dimensional maximal ghost domain at a given size will also be enclosed within a fixed number of steps. As with 3D, this happens once five steps have been taken in the direction of v_{kn} or v_{kn+1}. The scenario where T_n is enclosed within the fewest steps, the next direction is either v_{kn} or v_{kn+1}. This is then followed by $(k-2)$ directions $v_{kn-k+2}, \dots, v_{kn-1}$, and then repeats the cycle until five steps of the directions v_{kn} or v_{kn+1} have taken place. Here, we require two cycles of the directions in 5.6, plus one of either v_{kn} or v_{kn+1} at the start or end. Therefore, at best, all neighbours of T_n are contained in T_{n+2k+1}. This was seen in the proof of Theorem 22, as two steps were enclosed by T_{n+7}. In the worst case scenario, the next direction is v_{kn-k+2}. To enclose T_n here, we need two full cycles of the directions in 5.6, and third cycle omitting v_{kn+1}. Hence for every n, all neighbours of T_n are contained in T_{n+3k-1}.

Using this set of directions, we can again obtain a boundary ghost by adding the direction $(0, 1, 0, \dots)$. Using this direction is the most efficient, as seen in the 3D case, but there are more options to obtain shell-like boundary ghosts. Let $v_{-1} = (0, 1, x_3, ..., x_k)$ be a k-dimensional vector. Then any vector in the set $\{(0, 1, x_3, ..., x_k) \mid x_3, ..., x_k \in \{0, 1\}\}$ will produce a k-dimensional boundary ghost. Therefore, there are 2^{k-2} possible boundary ghosts.

5.6 Ghost Tiles in Physical Systems

In this section we will explore the interlocking of ghost tiles computationally, and how this may be used to model physical systems. Though we have been examining these shapes in the context of ghosts, we have primarily studied how similar tiles connect with each other. We have shown a recursion that produced non-degenerate directions that gives a translation for a shape to bind with itself. This may offer insight into processes where lattice structures grow, for example flocculation. Similarly to the Fibonacci sequence modelling the pattern of sunflower seeds, this recursion defines maximal contact areas of self similar tiles which can be viewed as an energy minimisation.

There are still many unanswered questions when it comes to the formation pathways of aggregated crystal growth [63]. Here, we have aggregation-based crystallization, starting from so-called nanoblocks. The true nature for

the growth of common mineral such as gypsum is still not well understood. The self-assembly in these minerals and also biological semi-organic crystals seem very similar to the tilings studied throughout this work. The diversity of growth directions may provide an important explanation for particular instances of symmetry breaking in simple nano-particles. Furthermore, the signed ghost elements may be useful for modelling the strong interactions for short-range surface bonding in material systems. There is a link here between the content of Chapter 3 and the Bernasconi model of pseudo-noise binary arrays for nearest neighbour spin-spin interactions that can be closely approximated by an auto-correlation coupling term in the Hamiltonian [48]. Ghosts have also been used to construct perfect arrays, which uses zero sum projections in the finite Radon transform [12].

The area of connection between two tiles was analysed computationally. We take the tile that results from our initial set of directions, and then find a direction that defines the distance between the two identical tiles such that the boundary contact is maximised. All non-degenerate directions are tested to find the ideal translation that maximises contact. Since the ghost domains are symmetric, it is only necessary to test directions (p, q) with $p > 0$.

We return briefly to the two dimensional case as a starting point. In 2D, the directions that maximise contact area correspond exactly to the recursion given in (4.6). Therefore, this recursion is maximal in a sense, when we limit the geometry to two dimensions. The boundary length was computed for U_n up to $n = 18$. Moreover, this maximal direction was unique for almost all n, aside from its negative counterpart due to the symmetry of the tiles. Only v_3 produced two directions that both give the same contact area. The two directions $(3, 1)$ and $(3, -1)$ have the same contact area due to additional symmetry of T_3. After this, there is no ambiguity in the maximal step. From Theorem 9, we know the exact length of this boundary. The connecting boundary length of U_n and $U_n + v_n$ is given by α_n. It can be computed through the recursion

$$\alpha_{n+1} = \alpha_{n-1} + 2\alpha_{n-2}$$

for $n = 3, 4, \ldots$, with $\alpha_1 = 1$, $\alpha_2 = 3$ and $\alpha_3 = 3$.

The same computation is then extended to 3D, where we compute the contact area of the joining faces of U_n and $U_n + v_n$. In Table 5.2, we give the directions which define the maximal contact area for a tile and its translation.

The sequence from (5.3) that we have been working with throughout this chapter is also shown for reference. We begin with the shape defined by the set of directions $\{(1, 0, 0), (1, 1, 0), (1, 0, 1)\}$, and look for the non-degenerate direction that gives a translation with the most contact faces between the identical shapes. The shape that results from the initial directions $\{(1, 0, 0), (0, 1, 0), (0, 0, 1)\}$ can be obtained through a linear transformation, and therefore similar results would hold. Due to the various symmetries that exist in these tiles, this direction is not unique, so we choose the direction that matches in sign with recursion (5.3).

Table 5.2: The set of directions v_n from recursion (5.3), and the non-degenerate directions with maximal contact to the previous tile.

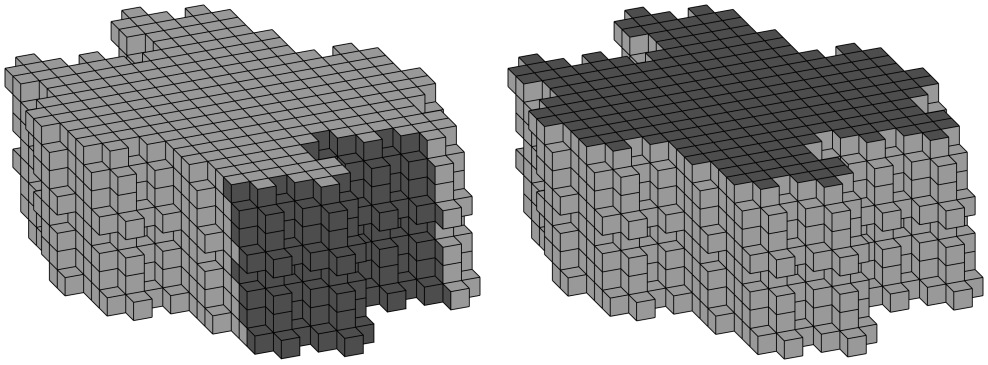
v_n from recursion (5.3)	Maximal contact direction
$(1, 0, 0)$	$(1, 0, 0)$
$(1, 1, 0)$	$(1, 1, 0)$
$(1, 0, 1)$	$(1, 0, 1)$
$(-1, 1, 0)$	$(-1, 1, 0)$
$(-3, -1, 0)$	$(-3, -1, 0)$
$(1, 0, 2)$	$(1, 0, 2)$
$(-1, -3, 0)$	$(-1, -3, 0)$
$(5, -1, 0)$	$(5, -1, 0)$
$(-1, 0, -4)$	$(-1, 0, -4)$
$(7, 5, 0)$	$(7, 5, 0)$
$(-3, 7, 0)$	$(-3, 7, 0)$
$(1, 0, 8)$	$(-17, -3, 0)$
$(-17, -3, 0)$	$(3, 0, 8)$
$(-11, -17, 0)$	$(-11, -17, 0)$
$(-1, 0, -16)$	$(23, -11, 0)$
$(23, -11, 0)$	$(45, 23, 0)$
$(45, 23, 0)$	$(-7, 0, -16)$
$(1, 0, 32)$	$(-1, 45, 0)$
$(-1, 45, 0)$	

Here, we can see that the two sets of directions match very closely. In some places directions are switched, but are only off by one step in the sequence. One main difference is the x component of the step in the z direction. In the context of tomography, we wanted x to be as small as possible to minimise the bounding box of the ghost. However, here the x component is approximately half of the z component, such that the faces of the shapes align more closely. Here, it is favourable for this direction to be as close as possible to 45° in the x-z plane.

From Table 5.2, it can be seen that before the directions $(3, 0, 8)$ and $(-7, 0, -16)$, there are three steps in the x-y plane. However, we know that this results in uneven growth for when the number of directions is large. This may be a temporary occurrence, or there may be a bias towards the directions in the x-y plane due to the surface around the edge of the tile.

In Fig. 5.7, the tile U_{11} is illustrated, and the faces that connect to $U_{11} + (-17, -3, 0)$ and $U_{11} + (3, 0, 8)$ respectively are shaded. The intersection $U_{11} \cap U_{11} + (-17, -3, 0)$ has an area of 297 while $U_{11} \cap U_{11} + (3, 0, 8)$ has area 256. Since the tile is textured around the edge (Fig. 5.7a) a single point in T_n may have multiple neighbouring points in the translated tile, which increases the area of contact. However, for shifts in the z direction, the two tiles meet along a flat face, limiting the number of joining faces. Though we have not explicitly

proven this idea of maximal contact area, this recursion may be useful for modelling physical processes.



a) b)

Figure 5.7: U_{11} where the dark shaded squares indicate the joining faces connected to: a) $U_{11} + (-17, -3, 0)$, b) $U_{11} + (3, 0, 8)$

5.7 Summary

In this chapter, we have extended the construction of maximal and boundary ghosts to higher dimensions using the 2D shapes as building blocks. Therefore, all of the properties of the 2D shapes hold in k-dimensions. However, due to the new degrees of freedom there are more possibilities. For a given maximal ghost in kD, we can produce 2^{k-2} unique boundary ghosts. Using the set of directions from (5.6), the similar connectivity properties hold from the 2D case. A maximal ghost with k-dimensions will have $2(k+1)$ neighbours, and every neighbour of T_n is contained within T_{n+3k-1}.

Computationally, these tiles have shown to have maximally interlocking boundaries under the co-prime direction constraint in two dimensions. Furthermore, in three dimensions the recursion was shown to nearly be maximal. Hence, this packing may give insight into various physical systems that have a structure analogous to the discrete grid. Processes such as aggregation-based crystallization are still not well understood, and this recursion may prove useful in modelling the growth of these systems. This will be a focus of future work.

Although we have shown some properties of the ghost domain with directions given by (5.6), in general there are many different shapes of ghosts that we can construct, and the process is similar for a ghost in any dimension. We start by using recursion (4.6) with different choices of $\epsilon_n \in \{-1, 1\}$. In Fig. 5.8, there are some examples of different possible shapes with seven directions.

Once the desired shape has been acquired in 2D, these tiles can then be stacked along any dimension to produce a ghost in k dimensions. These directions can be chosen to fit a given application, provided that the components

of these directions are co-prime. To maintain the property that ghost points with even x have value 1 and odd x points have value -1, p_n must be odd. However, if p_n is even, this will still produce a ghost where all layers have alternating columns of $+1$ and -1, but these columns will not necessarily line up across layers. In either case, a boundary ghost can be obtained by adding the direction $(0, 1)$. Though if p_n is not odd for all n then not all directions in the set $\{(0, 1, x_3, ..., x_k) \mid x_3, ..., x_k \in \{0, 1\}\}$ will give boundary ghosts.

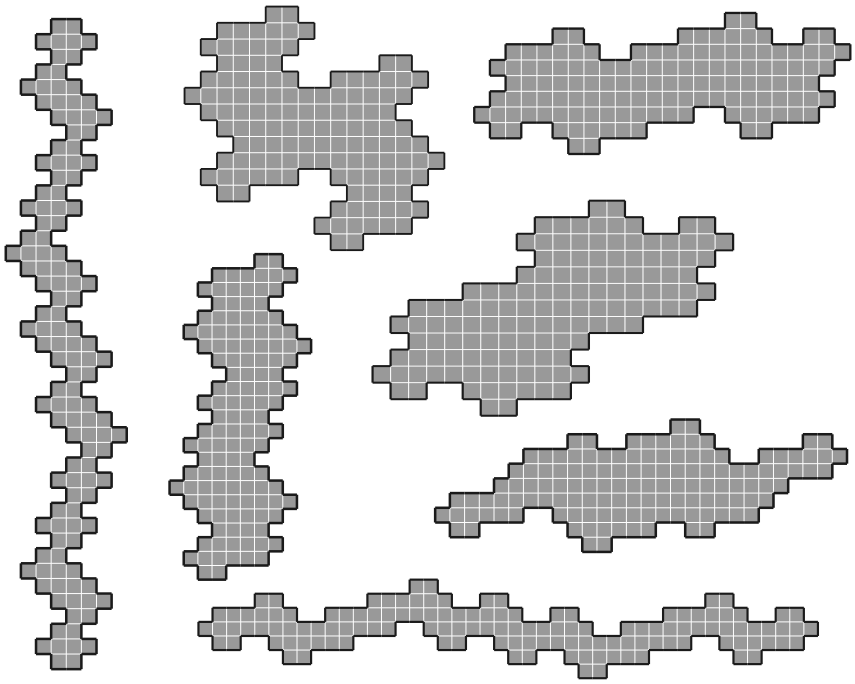


Figure 5.8: 2D ghost shapes for $N = 7$ with different choices of ϵ_n.

CHAPTER 6

Linear Time Reconstruction by Discrete Tomography

The reconstruction of an unknown function f from its line sums is the aim of discrete tomography. In general, many solutions are permitted, due to the presence of ghosts. Even in the case where there exists a unique solution, the problem is generally NP-hard, which makes many reconstruction algorithms inefficient. We show that this is not the case when f takes values in a unique factorization domain, such as $\mathbb{R}$ or $\mathbb{Z}$. In this chapter, we present a linear time reconstruction algorithm (in the number of directions and in the size of the grid), which outputs the original function values for all points outside of the switching domains. Freely chosen values are assigned to the points of the switching domains. This work is presented in [13], along with Silvia Pagani and Rob Tijdeman, who authored parts of this chapter. This extends their previous work from [52] to allow all pixels to be computed in linear time. My role in this research was to investigate the computational aspects of this algorithm, including implementation and complexity.

In this thesis, we have used aspects of discrete tomography to build arrays that can be utilised for tomography as well as broader communications applications. However, at this juncture we have not explicitly considered any methods for discrete reconstruction. It is important that tomographic reconstructions are not only accurate, but can be performed in a timely manner. In this chapter, we present an efficient algorithm that reconstructs an unknown function from its line sums in linear time. We assume the line sums are free of noise, but the algorithm does not expect the discrete projections to provide

sufficient information to admit a unique solution.

Discrete tomography finds its origin in the fifties, mainly for only two directions [61]. By 1978, M. Katz [47] gave a necessary and sufficient condition for the presence of a nontrivial function with vanishing line sums, known as a switching function or ghost. The theory started to blossom in the nineties when it became relevant in the study of crystals. In 1991 Fishburn, Lagarias, Reeds and Shepp [30] gave necessary and sufficient conditions for uniqueness of reconstruction of functions $f : A \to \{1, 2, \ldots, N\}$ for some positive integer N. An important distinction is whether the line sums are exact or may be inconsistent due to noise in the measurements. In the case where noise is present, the reconstruction can only be an approximation [6, 7, 53]. In what follows, we assume that the line sums are exact.

One of the main goals of discrete tomography is to ensure that the reconstructed function is true to the function f from which the line sums originate. However, in general the problem is ill-posed. Therefore one investigates which additional constraints can be imposed in order to achieve uniqueness. For instance, one may use some known information about the shape of the domain of f such as convexity [31], the values f can attain (for the binary case see [10, 43], for the integer case see [23]), or the size of the domain of f [10, 44]. In this work we assume that the line sums come from some function f and are therefore consistent.

In 1999 Gardner, Gritzmann and Prangenberg [32] showed that the problem of reconstructing a function $f : A \to \mathbb{N}$ from its line sums in d directions is solvable in polynomial time if $d = 2$, but it is NP-complete if $d \geq 3$. The NP-completeness concerns both consistency and uniqueness, as well as reconstruction. Moreover, a year later they showed that the three mentioned problems are NP-complete for two and more directions when more than five types of atoms are involved in the crystal [33]. Therefore, it is intrinsically hard to determine if there is a solution and if so, to find it.

In 2001 Hajdu and Tijdeman [44] gave an algebraic representation of the complete set of solutions over the integers. Their result also holds for solutions over the reals or any other unique factorization domain. They gave a polynomial expression for the nontrivial switching function with domain of minimal size, the so-called primitive switching polynomial, and showed that every switching polynomial is a multiple of the primitive switching polynomial. This implies that every switching function is a linear combination of domain shifts of the corresponding primitive switching function. Their result suggests that arbitrary function values can be given to a certain set of points and that thereafter the function values of the other points of A are uniquely determined by the line sums. This was made explicit by Dulio and Pagani [28] and serves as a building block in this chapter.

In 2015 Dulio, Frosini and Pagani [24] showed that in the corners of A the function values are uniquely determined and can be computed in linear time if the number of directions $d = 2$. Later they proved conditional results for $d = 3$

[25, 26]. Recently, Pagani and Tijdeman [52] generalized the result for any number of directions. In particular the object function can be reconstructed in linear time if there are no switching functions. Moreover, they showed that in general the part of A outside the convex hull of the union of all switching domains is uniquely determined and can be reconstructed in linear time. This result is another building block of our work here.

In the two papers [24, 52], it was shown that the reconstruction of $f : A \to \mathbb{R}$ is NP-hard in view of the above mentioned results of Gardner, Gritzmann and Prangenberg. However, this is incorrect, as we shall show. We prove that given the line sums of a function $f : A \to \mathbb{R}$ in the directions of D we can compute a function $g : A \to \mathbb{R}$ with the same line sums. Using the theory of [44] this implies that the complete set of such functions g can be explicitly presented. The crux of the result of Gardner, Gritzmann and Prangenberg is therefore the requirement that the solution g should have non-negative values.

In Chapter 4, we examined switching components called boundary ghosts, with a relatively large interior of points having values uniquely determined by their line sums. Figure 6.1 shows a boundary ghost, where the grey pixels signify the switching domain. All pixels that are not in the switching domain can be uniquely determined. This chapter introduces a method which makes it possible to compute these values in linear time ($\mathcal{O}(dmn)$).

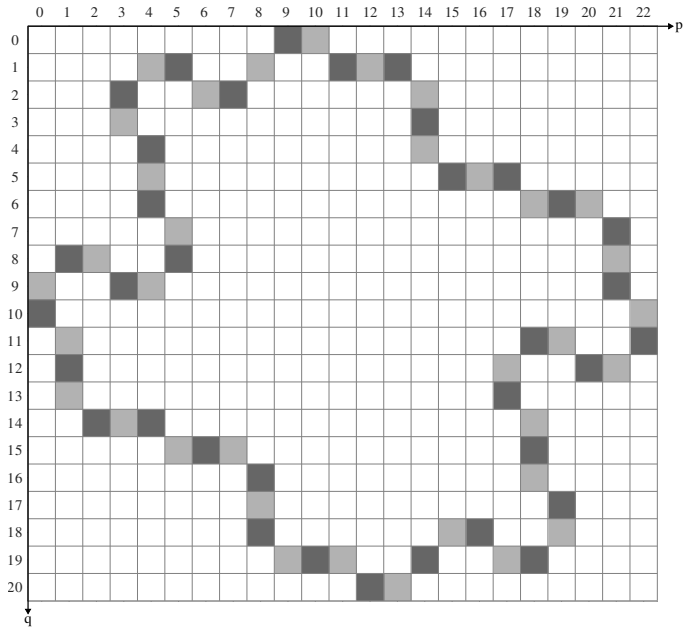


Figure 6.1: The grey pixels form the switching domain of a boundary ghost. The white pixels interior to the ghost domain have f-values which are uniquely determined by the line sums in the directions of $D = \{(0,1), (1,0), (1,1), (-1,1), (-3,-1), (-1,-3), (5,-1), (7,5), (-3,7)\}$.

In Section 6.1 we present notation and definitions, as well as information on switching functions. Section 6.2 shows how values of f in a corner region of

A can be obtained from the line sums. The case without switching components is treated in Section 6.3, that with switching components in Section 6.4. A general algorithm to compute g can be found in Section 6.5. The justification of our linear time claim is given in Section 6.6. A summary of the work is provided in Section 6.8.

6.1 Definitions and Known Results

We consider an $m \times n$ rectangular grid of points

$$A = \{(p, q) \in \mathbb{Z}^2 : 0 \leq p < m,\ 0 \leq q < n\}.$$

In the figures of this chapter, the x-axis is oriented from left to right and the y-axis downwards. The origin is therefore the upper-left corner point of A. To each point $(p, q) \in \mathbb{Z}^2$ we attach the pixel $(x, y) \in \mathbb{R}^2$ with $p \leq x < p + 1, q \leq y < q + 1$. In the figures, coordinates of a pixel are the coordinates of the attached point.

Primitive directions are pairs (a, b) of co-prime integers with $a \geq 0$, and $b = 1$ if $a = 0$. Since we only consider primitive directions, we simply call them directions. The horizontal and the vertical direction are given by $(1, 0)$ and $(0, 1)$, respectively. We consider a finite set of directions $D = \{(a_h, b_h) : h = 1, \ldots, d\}$. We say that D is valid for A if $M := \sum_{h=1}^{d} a_h < m$ and $N := \sum_{h=1}^{d} |b_h| < n$, and nonvalid otherwise.

A lattice line L is a line containing at least two points in $\mathbb{Z}^2$. Let $f : A \to \mathbb{R}$. The line sum of f along the lattice line $L(a, b, c) : ay = bx + c$ with direction (a, b) is defined as

$$\ell(a, b, c, f) = \sum_{aq=bp+c,\ (p,q)\in A} f(p, q).$$

A function $F : A \to \mathbb{R}$ is called a *switching function* or *ghost* of (A, D) if all the line sums of F in all the directions of D are zero. Observe that then f and $f + F$ have the same line sums in the directions of D. The support of a switching function is called a *switching domain*.

We say that a computation can be completed in linear time if the number of operations grows linearly with the size of the input. Here a basic operation is an addition, subtraction, multiplication, division, decision which of two quantities is larger or an assignment.

6.1.1 The Location of Switching Domains

M. Katz [47] proved that $f : A \to \mathbb{R}$ is uniquely determined by the line sums in the directions of D if and only if (A, D) is nonvalid. Fishburn et al. [30] showed that $(p, q) \in A$ has a unique f-value if and only if (p, q) is not located in a switching domain (or weakly bad configuration). Hajdu and

Tijdeman [44] associated to the function $f : A \to \mathbb{R}$ the polynomial $f^*(x,y) = \sum_{i=0}^{m-1}\sum_{j=0}^{n-1} f(i,j)x^i y^j$. In this way every switching function corresponds with a switching polynomial. They defined

$$g^*_{(a,b)}(x,y) = \begin{cases} x^a y^b - 1 & \text{if } a > 0, b > 0, \\ x^a - y^{-b} & \text{if } a > 0, b < 0, \\ x - 1 & \text{if } a = 1, b = 0, \\ y - 1 & \text{if } a = 0, b = 1, \end{cases}$$

and

$$G^*_{i,j}(x,y) = x^i y^j \prod_{h=1}^{d} g^*_{(a_h,b_h)}(x,y)$$

for $0 \le i < m - M, 0 \le j < n - N$. For an example of a switching polynomial, see Section 4.1.2. It was shown that $G^*_{0,0}$ is a switching polynomial of minimal degree. We call the corresponding function a primitive switching function. Furthermore they proved the following result.

Theorem 26 (Hajdu, Tijdeman [44], Theorem 1)**.** *Suppose D is valid for A. Put $M = \sum_{h=1}^{d} a_h, N = \sum_{h=1}^{d} |b_h|$. Then for every switching function $g : A \to \mathbb{R}$ its switching polynomial g^* can be uniquely written as*

$$g^* = \sum_{i=0}^{m-1-M} \sum_{j=0}^{n-1-N} c_{i,j} G^*_{i,j} \tag{6.1}$$

with $c_{i,j} \in \mathbb{R}$ for all i, j. Conversely, every function g of which the switching polynomial is of the form (6.1) *is a switching function.*

This result is also valid if $\mathbb{R}$ is replaced by $\mathbb{Z}$ or any other unique factorization domain. A corollary of the theorem relevant for this work is that the lexicographically lowest degree term of $G^*_{i,j}$ is given by $x^i y^{j+N_n}$ where $N_n = \sum_{b_h<0} -b_h$. Thus we have free choice for the values of $c_{i,j}$ for $0 \le i < m - M, N_n \le j < N_n + n - N$ and by this choice the function g^* is uniquely determined.

An illustration of Theorem 26 is given in Fig. 6.2 for a 26×19 grid A and the set of directions $D = \{(5,-2), (4,-3), (3,-4), (6,1), (3,2), (2,5)\}$. The dark grey pixels indicate the union of the switching domains. Here, the function f is not uniquely determined by its line sums in the directions of D. The f-values of the complement, the white and light grey pixels, are uniquely determined by these line sums. Since $M = 23$, $N = 17$, there are six pixels where the choice is free, for example the pixels $(0,9)$, $(0,10)$, $(1,9)$, $(1,10)$, $(2,9)$, $(2,10)$. Any other 3 by 2 block of dark grey pixels can be chosen instead. If the choice is made all the values of the unique solution satisfying the made choices are determined by the line sums in the directions of D. The white pixels form four corner regions. The blue line indicates the convex hull of the union of the switching components.

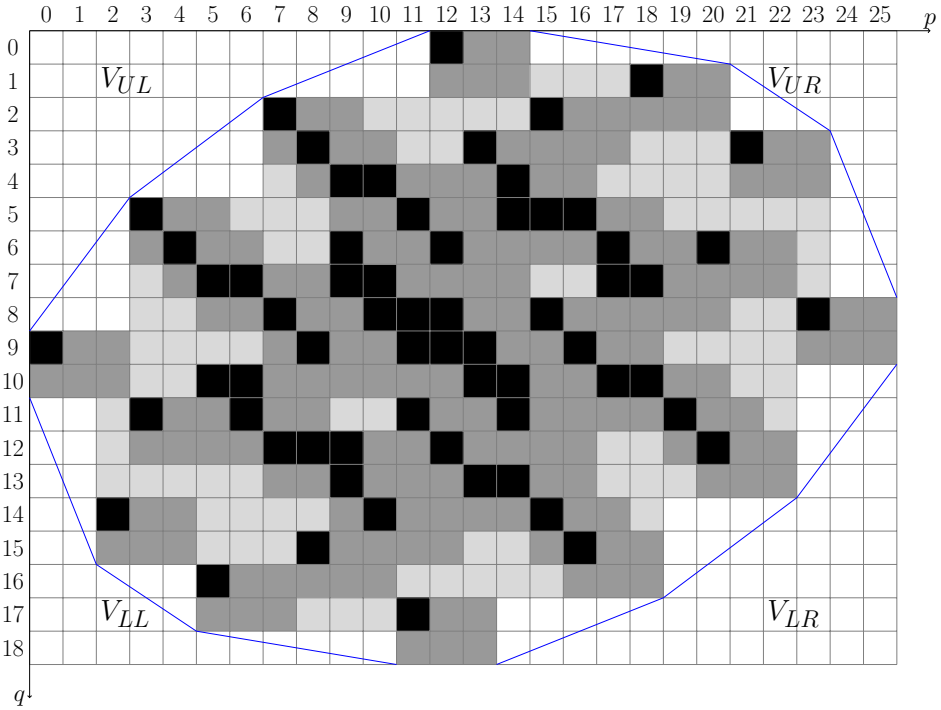


Figure 6.2: Points of a 26×19 grid A and the set of directions $D = \{(5,-2), (4,-3), (3,-4), (6,1), (3,2), (2,5)\}$. Dark grey pixels show the union of switching domains. White and grey pixels are uniquely determinable points. The black pixels represent the switching domain related to $G^*_{0,0}$.

6.2 Uniqueness in the Corner Regions

Again let $A = \{(p,q) \in \mathbb{Z}^2 : 0 \leq p < m, 0 \leq q < n\}$. Let D be a set of directions $(a_1,-b_1),\ldots,(a_k,-b_k)$ with $k \geq 2$ where $a_1,\ldots,a_k,b_1,\ldots b_k$ are positive integers ordered such that

$$\frac{b_1}{a_1} < \frac{b_2}{a_2} < \ldots < \frac{b_k}{a_k}. \tag{6.2}$$

Note that by primitivity all the ratios are distinct. We call the points

$$\left(\sum_{h=1}^{k} a_h,\, 0\right),\ \left(\sum_{h=2}^{k} a_h,\, b_1\right),\ \left(\sum_{h=3}^{k} a_h,\, \sum_{h=1}^{2} b_h\right),\ \ldots,\ \left(0,\, \sum_{h=1}^{k} b_h\right)$$

the border points $(P_0,Q_0),(P_1,Q_1),\ldots,(P_k,Q_k)$, respectively. We denote the convex hull of the three points $(0,0),(P_{H-1},Q_{H-1}),(P_H,Q_H)$ by V_H for $H = 1,2,\ldots,k$ (see Fig. 6.3). Let

$$V_{UL} = \bigcup_{H=1}^{k} V_H$$

be the upper left corner region.

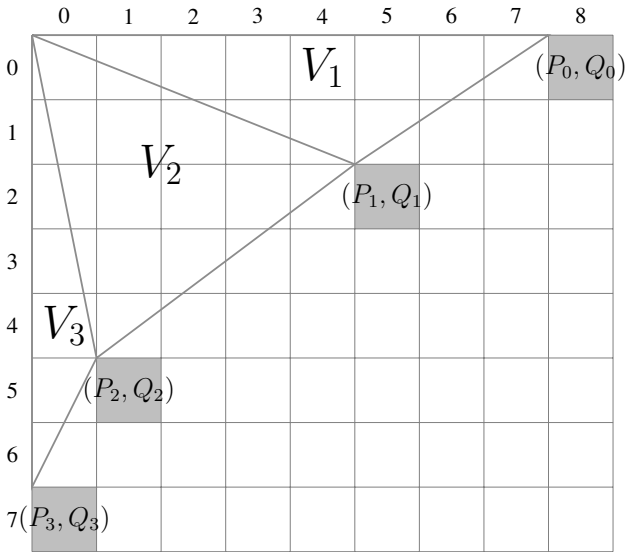


Figure 6.3: The triangles V_1, V_2, V_3 for the set $D = \{(3,-2),(4,-3),(1,-2)\}$. The border points are $(P_0, Q_0) = (8,0), (P_1, Q_1) = (5,2), (P_2, Q_2) = (1,5), (P_3, Q_3) = (0,7)$. For every H the line through (P_{H-1}, Q_{H-1}) and (P_H, Q_H) is a side of triangle V_H, and intersects each other triangle V_h, since the slopes increase with increasing h by the ordering in (6.2).

For a point $(p,q) \in A$ we define its weight $w(p,q)$ by

$$w(p,q) = \min_{h=1,2,\ldots,k} \frac{b_h p + a_h q}{b_h P_h + a_h Q_h}. \tag{6.3}$$

The following lemma implies that if $(p,q) \in V_{UL}$, then the minimum in the definition of $w(p,q)$ is reached for h such that $(p,q) \in V_h$.

Lemma 27 ([52], Lemma 2)**.** *For $(p,q) \in A$ the weight $w(p,q)$ is reached for h such that*

$$\frac{Q_{h-1}}{P_{h-1}} \leq \frac{q}{p} \leq \frac{Q_h}{P_h}.$$

and only for such h. The weight 1 is reached at the border points and not at other points of A.

The next result states that the corner region V_{UL} except for the border points has unique f-values which can be computed in linear time.

Theorem 28 ([52], Theorem 4, Corollary 6)**.** *Let $A = \{(p,q) \in \mathbb{Z}^2 : 0 \leq p < m,\ 0 \leq q < n\}$. Let D be a set of directions $(a_1, -b_1), \ldots, (a_k, -b_k)$ where $a_1, \ldots, a_k, b_1, \ldots b_k$ are positive integers ordered as in (6.2). Let the line sums of $f : A \to \mathbb{R}$ in the directions of D are given. Then all the points (p,q) in V_{UL} except for the border points have uniquely determined f-values.*

The points which have uniquely determined f-values can be computed by treating points (p,q) in V_{UL} according to increasing weights and, if $(p,q) \in V_h$,

by subtracting the sum of the f-values of the other points of A on the line through (p, q) in the direction of (a_h, b_h) from its line sum.

Corollary 29. *Under the above conditions the f-values of the points $(0, 0)$, $(0, 1)$, ..., $\left(0, -1 + \sum_{h=1}^{k} b_h\right)$ can all be computed in linear time.*

A demonstration of these results can be seen in Fig. 6.4 for the directions $(3, -2)$, $(4, -3)$, $(1, -2)$. The upper number in each pixel is the weight of the point. The border points are $(8, 0)$, $(5, 2)$, $(1, 5)$, $(0, 7)$. All the points with weight less than 1 are in the corner region and have uniquely determined f-values (Theorem 28). The (dark grey) border pixels are part of the switching domain and their f-values are therefore not uniquely determined. They have weight 1. All entirely white pixels have weight > 1.

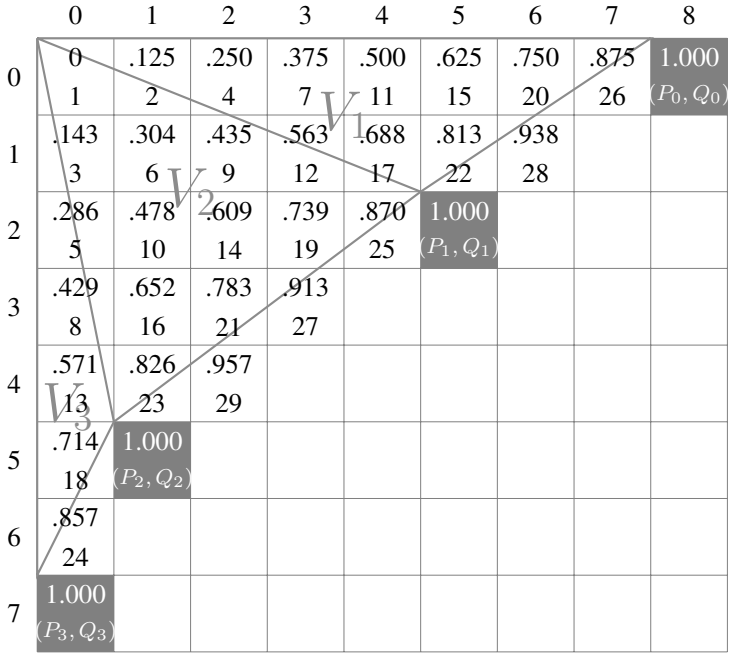


Figure 6.4: The weights (upper number inside each pixel) of the points for directions $(3, -2), (4, -3), (1, -2)$. The lower numbers enumerate the increasing weights.

6.3 The Nonvalid Case

Suppose we are in the nonvalid case, so $M \geq m$ or $N \geq n$. Without loss of generality assume that $N \geq n$. We then apply Theorem 28 to both the upper corner region V_{UL} and to the lower corner region V_{LL}.

Let A be as above. Let

$$D = \{(a_1, -b_1), \ldots, (a_k, -b_k), (a_{k+1}, b_{k+1}), \ldots, (a_d, b_d), (0, 1)^*, (1, 0)^*\}$$

where $a_1, \ldots, a_d, b_1, \ldots b_d$ are positive integers ordered such that

$$\frac{b_1}{a_1} < \frac{b_2}{a_2} < \ldots < \frac{b_k}{a_k}, \qquad \frac{b_{k+1}}{a_{k+1}} > \frac{b_{k+2}}{a_{k+2}} > \ldots > \frac{b_d}{a_d}$$

and the asterisk indicates that $(0,1)$ and $(1,0)$ may or may not occur in D. Thus we assume that $n \leq \sum_{h=1}^{d} b_h$ or $(n = 1 + \sum_{h=1}^{d} b_h$ and $(0,1) \in D)$.

By Corollary 29 applied to V_{UL}, the f-values of the points $(0,0)$, $(0,1)$, $\ldots$, $\left(0, -1 + \sum_{h=1}^{k} b_k\right)$ can be computed. In a similar way we can apply the same corollary to V_{LL} and the directions $(a_{k+1}, b_{k+1}), \ldots, (a_d, b_d)$ to conclude that the f-values of the points $(0, n-1), (0, n-2), \ldots, \left(0, n - \sum_{h=k+1}^{d} b_h\right)$ can be computed. It follows that the f-values of the points $(0,0), (0,1), \ldots, (0, n-1)$ can all be computed except when $n = 1 + \sum_{h=1}^{d} b_h$ and $(0,1) \in D$. In the latter case $(p,q) = \left(0, \sum_{h=1}^{k} b_h\right)$ is the only point in the column $p = 0$ with unknown f-value. However, this value can be found by subtracting from the line sum of the column $p = 0$ the f-values of the other points in that column. In this way we have made our problem of computing the f-values one column smaller. We can repeat the procedure in order to find the f-values of the next column. Continuing the process we arrive at the following conclusion.

Theorem 30. *Let $A = \{(p,q) \in \mathbb{Z}^2 : 0 \leq p < m,\ 0 \leq q < n\}$. Let D be a set of directions such that A is nonvalid for D. Let the line sums of $f : A \to \mathbb{R}$ be given. Then the f-values of all points of A can be computed in linear time.*

In [52] algorithms are given for computing the f-values. These algorithms are more efficient than the procedure described above. However, the algorithm in Section 6.5 is as efficient as these algorithms.

6.4 The Valid Case

Let $A = \{(p,q) \in \mathbb{Z}^2 : 0 \leq p < m,\ 0 \leq q < n\}$ and

$$D = \{(a_1, -b_1), \ldots, (a_k, -b_k), (a_{k+1}, b_{k+1}), \ldots, (a_d, b_d), (0,1)^*, (1,0)^*\}$$

where $a_1, \ldots, a_d, b_1, \ldots b_d$ are positive integers ordered such that

$$\frac{b_1}{a_1} < \frac{b_2}{a_2} < \ldots < \frac{b_k}{a_k}, \qquad \frac{b_{k+1}}{a_{k+1}} > \frac{b_{k+2}}{a_{k+2}} > \ldots > \frac{b_d}{a_d}$$

and the asterisk indicates that $(0,1)$ and $(1,0)$ may or may not occur in D. As observed in the previous section, by applying Corollary 29 to V_{UL} the g-values of the points $(0,0), (0,1), \ldots, \left(0, -1 + \sum_{h=1}^{k} b_h\right)$ can be computed. In a similar way we can apply the corollary to V_{LL} and the directions $(a_{k+1}, b_{k+1}), \ldots, (a_d, b_d)$ to conclude that the g-values of the points $(0, n -$

$1), (0, n-2), \ldots, \left(0, n - \sum_{h=k+1}^{d} b_h\right)$ can be computed. In Section 6.1.1 it was observed that the g-values of the points $(0, Q_k), (0, Q_k + 1), \ldots, (0, Q_k + n - N - 1)$ can be freely chosen where $g : A \to \mathbb{R}$ is a function satisfying the line sums. Combining these results we see that all the g-values of the points $(0, 0), (0, 1), \ldots, (0, n-1)$ can be computed or freely chosen, except for the case that $(0, 1) \in D$ and $n = 1 + \sum_{h-1}^{d} b_d$. In the latter case only the g-value of $\left(0, \sum_{h=1}^{k} b_h\right)$ is not determined, but this can be computed by subtracting the g-values of the other points in the leftmost column from the sum of that column. After this all g-values of the points in the leftmost column are fixed. In Section 6.1.1 it was further observed that the g-values of the points $(p, Q_k), (p, Q_k + 1), \ldots, (p, Q_k + n - N - 1)$ for $p = 1, 2, \ldots, m - M - 1$ can be freely chosen. Therefore we can repeat the above procedure successively for columns $p = 1, 2, \ldots, m - M - 1$. Then M columns remain, for which the line sums in the directions of D are known. It is obvious that the computed g-values of the points which do not belong to a switching domain have the original f-value. We are left with a nonvalid case and we can apply an algorithm for that case to compute the remaining g-values.

We have shown that the following theorem holds.

Theorem 31. *Let $A = \{(p, q) \in \mathbb{Z}^2 : 0 \leq p < m,\ 0 \leq q < n\}$ and D a set of primitive directions. Let $f : A \to \mathbb{R}$ be an unknown function. Suppose all the line sums in the directions of D are known. Then we can compute a function $g : A \to \mathbb{R}$ satisfying the line sums in linear time. The points which do not belong to any switching domain get their original f-value.*

Example 32. Consider the situation in Fig. 6.2. We have $m = 26, M = 23, n = 19, N = 17, Q_k = 9$. We can freely choose the g-values of the points $(0, 9)$ and $(0, 10)$ and compute the g-values of the other points $(0, q)$. Next we do so for the columns $p = 1$ and $p = 2$. We are left with a 23 by 19 rectangular grid. Since $m = M = 23$, this is a nonvalid case and we know that the remaining g-values can be computed in linear time. The found g-values of the white and light grey pixels are equal to the original f-values.

Remark 33. In this work we assume that the line sums are correct and that there is no noise. It is easy to check whether this is true afterwards by checking the line sums which have not been used for computing the g-values. In case the line sums are inconsistent, and it is better to use a method which treats the unused line sums in a similar way as the used line sums to obtain a good approximation of the original function.

Remark 34. Theorem 26 states that if $g : A \to \mathbb{R}$ satisfies the same line sums as f, then the associated polynomial g^* is of the form

$$f^* + \sum_{i=0}^{m-1-M} \sum_{j=0}^{n-1-N} c_{i,j} G^*_{i,j}$$

and each such function satisfies the same line sums as f. It is possible to compute the coefficients $c_{i,j}$ as follows. The point $(0, Q_k)$ occurs only in the domain of $G_{0,0}$ and therefore $c_{0,0}$ can be found from the found value for $(0, Q_k)$. The point $(0, Q_k + 1)$ occurs in $G_{0,1}$ and maybe in $G_{0,0}$. Since $c_{0,0}$ is already known, $c_{0,1}$ can be computed. Considering the points with a free choice in the lexicographic order, each time a point occurs in only one new primitive switching domain and hence the corresponding coefficient can be computed.

6.5 An Efficient Algorithm

In this section we present an algorithm to find a function $g : A \to \mathbb{R}$ which satisfies the given line sums of an unknown function $f : A \to \mathbb{R}$. This algorithm is based on ideas and results in [52], to allow for the computation of values within the convex hull of the switching domain using a twofold application of the previous method. Its complexity is studied in the next section.

In this algorithm it is not necessary to compute all weights as in Fig. 6.4. Observe that after the g-values in V_{UL} and V_{LL} have been computed, the g-value of the next point on each row can be computed. The next point of the row will be computed using the same direction, hence everything shifts one place to the right. For this reason, the order in which the new points will be handled will be the same as for the points immediately directly left of them. This process is then continued.

In the example of Fig. 6.4 the g-values of the initial row $q = 1$ are computed first using the direction $(1, -2)$, next the direction $(4, -3)$, and finally the direction $(3, -2)$. Moving more to the right, only the direction $(3, -2)$ would be used. Suppose there would have been seven more columns $p = -1, -2, \ldots, -7$ with g-values equal to 0 in A (cf. Algorithm 1B of [52]). Then for $p \geq 0$ we would have needed only the rightmost direction on each row and the direction needed to compute the g-values of points in the original A would only depend on their row. Therefore it suffices to follow the order in which the rightmost non-border points of V_{UL} and V_{LL} are treated and for each point (p, q) to use the line sum in the direction (a_h, b_h) with h such that the corresponding rightmost point is in V_h. Observe that this h is such that $Q_h \leq q < Q_{h+1}$.

Example 35. Let the directions be $(3, -2), (4, -3), (1, -2)$ as in Fig. 6.4. The border points are $(8, 0), (5, 2), (1, 5), (0, 7)$. Let $m > 8, n = 7$. Then the rightmost points with weight < 1 are

$$(7, 0),\ (6, 1),\ (4, 2),\ (3, 3),\ (2, 4),\ (0, 5),\ (0, 6).$$

If we order them according to increasing weights, then we get

$$(0, 5),\ (0, 6),\ (4, 2),\ (7, 0),\ (3, 3),\ (6, 1),\ (2, 4).$$

If we use the invisible seven columns on the left with g-values 0, then the enumeration of V_{UL} is given by the lower numbers in Fig. 6.5. Observe that it

differs from the enumeration in Fig. 6.4. For computing the g-values in rows 0 and 1 direction $(3, -2)$ is used, in rows 2 to 4 direction $(4, -3)$ and in rows 5 and 6 direction $(1, -2)$.

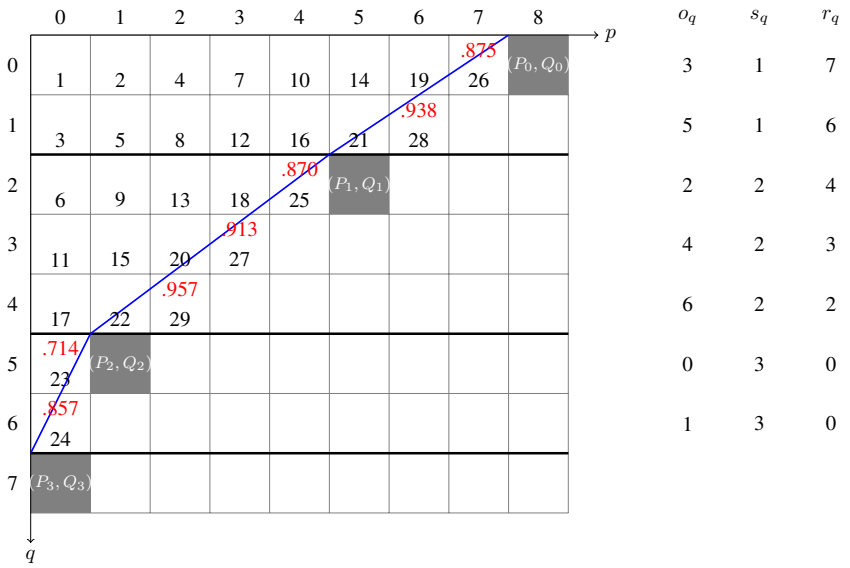


Figure 6.5: The weights (upper red numbers) of the rightmost points in V_{UL} of each row and the ordering of points in V_{UL} (lower numbers) of Example 35. On the right, o_q reports the order of the rightmost points in V_{UL} after increasing weights, $s(q)$ tells which direction has been used in the corresponding row and r_q indicates, for each row, the p-coordinate of the rightmost point in V_{UL}.

The above example illustrates the first seven steps of the algorithm. The above procedure is done for both V_{UL} and for V_{LL}. In between g-values 0 are substituted (or any other values) at the places where the switching domains offer free choice. Now the g-values in the first column are known and we can proceed with the next column and so on until we have treated $m-M$ columns. An M by n grid remains to be handled, but this is a nonvalid case. This can be treated in a similar way, but starting from upper corner regions V_{UL} and V_{UR} and then going downwards. In the algorithm we have $g(p, q) = f(p, q)$ for all the points (p, q) which are not in a switching domain of (A, D).

The order in which g-values are computed is shown in Fig. 6.6 for $m = 21, n = 16, D = \{(0, 1), (1, 0), (1, 1)(-1, 1), (-3, -1), (-1, -3), (5, -1), (7, 5)\}$. The switching domain is shown in grey, and 0 g-values are substituted in the dark grey pixels. The rightmost points with weight < 1 are shown in dark blue, and the ordering is listed beside these points. Beginning at the leftmost shift $t = 1 - \max(P_0, P_d)$. The g-values are computed according to the enumeration order for all points in the grid. The points are then shifted to the left until the process has been completed for $t = m - M - 1$, and the g-values for all blue points in Fig. 6.6 have been computed. A similar process in then repeated in the y-direction with the red points for the remaining grid.

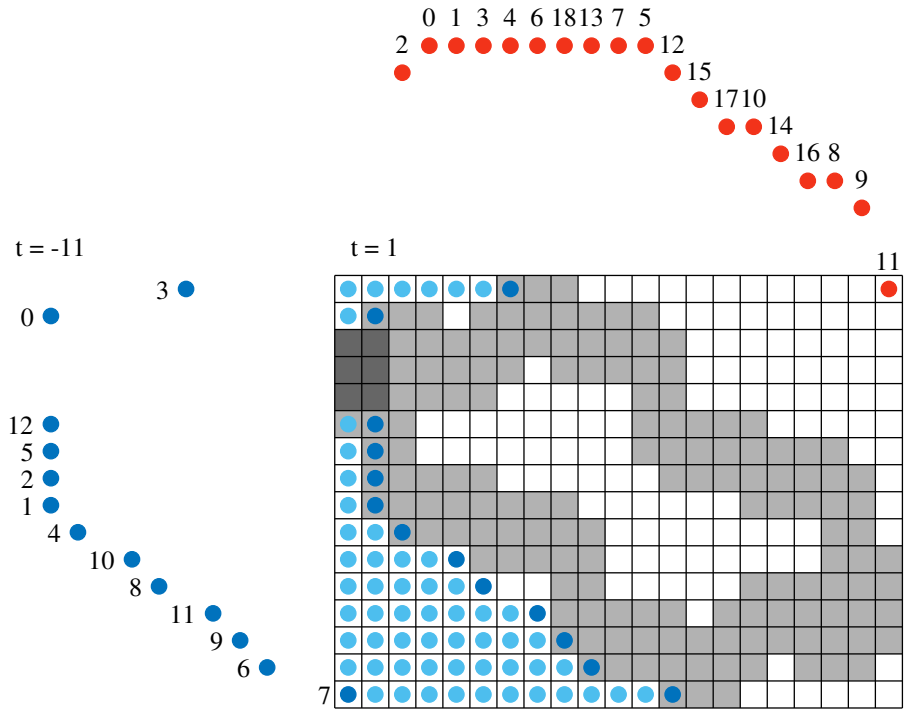


Figure 6.6: The order in which g-values are computed in the algorithm.

The algorithm has 12 steps and is illustrated in Example 37 following it. For Steps 1-7 see Fig. 6.7, for Steps 8-12 see Fig. 6.8. In the algorithm, $\lambda(\mathbf{d}, p, q)$ denotes the line sum containing the point (p, q) in the direction $\mathbf{d}$, and $\lceil r \rceil$ denotes the ceiling of r.

In Step 1 the directions are ordered in such a way that the corner regions become concave. Throughout the algorithm we write b_h instead of $-b_h$ ($h = 1, \ldots, k$) so that b_h is always non-negative. In Step 2 the border points of V_{UL} and V_{LL} are found. In Step 3 a function δ is introduced indicating whether the g-value of a point has been computed. Step 4 provides a shortcut for nonvalid cases. If $n \leq N$, then this shortcut can be used after rows and columns have been interchanged. Step 5 serves to find the grid points left of the border line, measured by the sequence r. The weights of these points are computed as well, together with the sequence s, which indicates the direction of the line used to compute the g-value. In Step 6 the points found in the previous step are ordered after increasing weight by the sequence o. If weights are equal, the order is irrelevant. In Step 7 the g-values of the first $m - M$ columns (and of some more points of A) are computed.

Steps 8-12 are essentially equal to Steps 1-7, but with the columns $p = 0, 1, \ldots, m - M - 1$ omitted, as they have already been treated, and the roles of the rows and columns interchanged. In Step 8 the directions are reordered as now V_{UL} is mirrored and V_{UR} takes over the role of V_{LL}. A similar reordering of border points takes place in Step 9. The weights and the corresponding directions are found in Step 10. The order of the points found in the previous

step are fixed in Step 11. Finally the remaining g-values are computed in Step 12 where the u is introduced to make the necessary shift because of the omitted $m - M$ columns.

Algorithm 4 Linear time reconstruction of line sums.

Input: A set $A = \{(p, q) : 0 \leq p < m, 0 \leq q < n\}$ with positive integers m, n, a finite set of (primitive) directions D and all the line sums in the directions of D of a function $f : A \to \mathbb{R}$.

Output: Function $g : A \to \mathbb{R}$ which satisfies the line sums.

Step 1: Initial values.

1: **for all** $\mathbf{d} = (a, -b) \in D$ (with $a > 0, b > 0$) order the directions such that

$$\frac{b_1}{a_1} < \frac{b_2}{a_2} < \cdots < \frac{b_k}{a_k}.$$

2: **for all** $\mathbf{d} = (a, b) \in D$ (with $a > 0, b > 0$) order the directions such that

$$\frac{b_{k+1}}{a_{k+1}} > \frac{b_{k+2}}{a_{k+2}} > \cdots > \frac{b_d}{a_d}.$$

3: $M \leftarrow \sum_{h=1}^{d} a_h$
4: $N \leftarrow \sum_{h=1}^{d} b_h$
5: **if** $(1, 0) \in D$ **then** $M \leftarrow M + 1$
6: **if** $(0, 1) \in D$ **then**
7: $\quad N \leftarrow N + 1$
8: $\quad \mathbf{d}_0 \leftarrow (0, 1)$

Step 2: Border points.

9: $(P_0, Q_0) \leftarrow \left(\sum_{h=1}^{k} a_h, 0\right)$
10: **for** $h \leftarrow 1$ **to** k **do**
11: $\quad (P_h, Q_h) \leftarrow (P_{h-1} - a_h, Q_{h-1} + b_h)$
12: $(P_k^*, Q_k^*) \leftarrow \left(0, n - 1 - \sum_{j=k+1}^{d} b_j\right)$
13: $(P_{k+1}, Q_{k+1}) \leftarrow (P_k^* + a_{k+1}, Q_k^* + b_{k+1})$
14: **for** $h \leftarrow k + 2$ **to** d **do**
15: $\quad (P_h, Q_h) \leftarrow (P_{h-1} + a_h, Q_{h-1} + b_h)$

Step 3: Fixing switching functions.

16: **for** $p \leftarrow 0$ **to** $m - 1$ **do**
17: $\quad$ **for** $q \leftarrow 0$ **to** $n - 1$ **do**
18: $\quad\quad \delta(p, q) \leftarrow 0$
19: **for** $p \leftarrow 0$ **to** $m - M - 1$ **do**
20: $\quad$ **for** $q \leftarrow Q_k$ **to** $Q_k + n - N - 1$ **do**
21: $\quad\quad g(p, q) \leftarrow 0$

22: $\delta(p,q) \leftarrow 1$

Step 4: Nonvalid case.

23: **if** $m \leq M$ **then** goto Step 8

Step 5: Choosing starting points r_h, weights $w(r_h,h)$ and directions $\mathbf{d}_{s(h)}$.

24: **for** $H \leftarrow 1$ **to** k **do**
25: **for** $h \leftarrow Q_{H-1}$ **to** $Q_H - 1$ **do**
26: $r_h \leftarrow \left\lceil \frac{(Q_H - h)P_{H-1} + (h - Q_{H-1})P_H}{Q_H - Q_{H-1}} - 1 \right\rceil$
27: $w(r_h,h) \leftarrow \frac{b_H r_h + a_H h}{b_H P_H + a_H Q_H}$
28: $s(h) \leftarrow H$
29: **for** $h \leftarrow Q_k^* + 1$ **to** Q_{k+1} **do**
30: $r_h \leftarrow \left\lceil \frac{(h - Q_k^*)P_{k+1}}{Q_{k+1} - Q_k^*} - 1 \right\rceil$
31: $w(r_h,h) \leftarrow \frac{b_{k+1} r_h + a_{k+1}(n-1-h)}{b_{k+1} P_{k+1} + a_{k+1}(n-1-Q_{k+1})}$
32: $s(h) \leftarrow k+1$
33: **for** $H \leftarrow k+2$ **to** d **do**
34: **for** $h \leftarrow Q_{H-1} + 1$ **to** Q_H **do**
35: $r_h \leftarrow \left\lceil \frac{(Q_H - h)P_{H-1} + (h - Q_{H-1})P_H}{Q_H - Q_{H-1}} - 1 \right\rceil$
36: $w(r_h,h) \leftarrow \frac{b_H r_h + a_H(n-1-h)}{b_H P_H + a_H(n-1-Q_H)}$
37: $s(h) \leftarrow H$
38: **if** $(0,1) \in D$ **then** $s(Q_k^*) \leftarrow 0$

Step 6: Ordering the points.

39: Order the points (r_h,h) for $h \leftarrow 0,1,\ldots,Q_k-1,Q_k^*+1,Q_k^*+2,\ldots,n-1$ after increasing values of $w(r_h,h)$ and call these points in this order $(p_0,q_0),(p_1,q_1)\ldots,(p_{N-1},q_{N-1})$.
40: **if** $(0,1) \in D$ **then** $(p_{N-1},q_{N-1}) \leftarrow (0,Q_k^*)$

Step 7: Assignment of f-values.

41: **for** $t \leftarrow 1 - \max(P_0,P_d)$ **to** $m - M - 1$ **do**
42: **for** $h \leftarrow 0$ **to** $N-1$ **do**
43: **if** $0 \leq p_h + t < m$ **and** $\delta(p_h+t,q_h) = 0$ **then**
44: $g(p_h+t,q_h) \leftarrow \lambda(\mathbf{d}_{s(q_h)}, p_h+t, q_h)$
45: **for all** $\mathbf{d} \in D$ **do**
46: $\lambda(\mathbf{d},p_h+t,q_h) \leftarrow \lambda(\mathbf{d},p_h+t,q_h) - g(p_h+t,q_h)$
47: $\delta(p_h+t,q_h) \leftarrow 1$

Step 8: Start nonvalid case, initial values, cf. Step 1.

48: $((a_1,b_1),\ldots,(a_k,b_k)) \leftarrow ((a_k,b_k),\ldots,(a_1,b_1))$
49: $((a_{k+1},b_{k+1}),\ldots,(a_d,b_d)) \leftarrow ((a_d,b_d),\ldots,(a_{k+1},b_{k+1}))$

50: **if** $(1,0) \in D$ **then** $\mathbf{d}_0 \leftarrow (1,0)$

Step 9: Border points, cf. Step 2.

51: **if** $M > m$ **then** $M \leftarrow m$
52: $((P_0,Q_0),\ldots,(P_k,Q_k)) \leftarrow ((P_k,Q_k),\ldots,(P_0,Q_0))$
53: $((P_k^*,Q_k^*),(P_{k+1},Q_{k+1}),\ldots,(P_d,Q_d)) \leftarrow ((M-P_d-1,n-Q_d-1),\ldots,$
54: $\qquad (M-P_{k+1}-1,n-Q_{k+1}-1),(M-P_k^*-1,n-Q_k^*-1))$

Step 10: Choosing starting points r_h, weights $w(r_h,h)$ and directions $\mathbf{d}_{s(h)}$, cf. Step 5.

55: **for** $H \leftarrow 1$ **to** k **do**
56: $\quad$ **for** $h \leftarrow P_{H-1}$ **to** $P_H - 1$ **do**
57: $\quad\quad r_h \leftarrow \left\lceil \dfrac{(P_H-h)Q_{H-1}+(h-P_{H-1})Q_H}{P_H-P_{H-1}} - 1 \right\rceil$
58: $\quad\quad w(h,r_h) \leftarrow \dfrac{a_H r_h + b_H h}{a_H Q_H + b_H P_H}$
59: $\quad\quad s(h) \leftarrow H$
60: **for** $h \leftarrow P_k^* + 1$ **to** P_{k+1} **do**
61: $\quad r_h \leftarrow \left\lceil \dfrac{(h-P_k^*)Q_{k+1}}{P_{k+1}-P_k^*} - 1 \right\rceil$
62: $\quad w(h,r_h) \leftarrow \dfrac{M-1-h}{M-1-P_k^*}$
63: $\quad s(h) \leftarrow k+1$
64: **for** $H \leftarrow k+2$ **to** d **do**
65: $\quad$ **for** $h \leftarrow P_{H-1}+1$ **to** P_H **do**
66: $\quad\quad r_h \leftarrow \left\lceil \dfrac{(P_H-h)Q_{H-1}+(h-P_{H-1})Q_H}{P_H-P_{H-1}} - 1 \right\rceil$
67: $\quad\quad w(h,r_h) \leftarrow \dfrac{a_H r_h + b_H(M-1-h)}{a_H Q_{H-1} + b_H(M-1-P_{H-1})}$
68: $\quad\quad s(h) \leftarrow H$
69: **if** $(1,0) \in D$ **then** $s(P_k^*) \leftarrow 0$

Step 11: Ordering the points, cf. Step 6.

70: Order the points (r_h,h) for $h \leftarrow 0,1,\ldots,P_k-1,P_k^*+1,P_k^*+2,\ldots,M-1$ after increasing values of $w(r_h,h)$ and call these points in this order $(p_0,q_0),(p_1,q_1),\ldots,(p_{M-1},q_{M-1})$.
71: **if** $(1,0) \in D$ **then** $(p_{M-1},q_{M-1}) \leftarrow (P_k^*,0)$

Step 12: Assignment of f-values, cf. Step 7.

72: $u \leftarrow m - M$
73: **for** $t \leftarrow 1 - \max(Q_0,Q_d)$ **to** $n-1$ **do**
74: $\quad$ **for** $h \leftarrow 0$ **to** $M-1$ **do**
75: $\quad\quad$ **if** $0 \le p_h + u < m$ **and** $0 \le q_h + t < n$ **and** $\delta(p_h+u,q_h+t) = 0$ **then**
76: $\quad\quad\quad g(p_h+u,q_h+t) \leftarrow \lambda(\mathbf{d}_{s(p_h)},p_h+u,q_h+t)$
77: $\quad\quad\quad$ **for all** $\mathbf{d} \in D$ **do**

78: $\lambda(\mathbf{d}, p_h + u, q_h + t) \leftarrow \lambda(\mathbf{d}, p_h + u, q_h + t) - g(p_h + u, q_h + t)$

79: $\delta(p_h + u, q_h + t) \leftarrow 1$

return g

Remark 36. We are assuming that line sums are exact. So, the fact that the output is consistent with the data can be easily checked by observing whether all line sums are equal to zero (Steps 7 and 12 update the line sums by subtracting the value of each point).

Example 37. Let $m = 21$, $n = 16$, $D = \{(0,1), (1,0), (1,1)(-1,1), (-3,-1), (-1,-3), (5,-1), (7,5)\}$ and the line sums in the directions of D of some function $f : A \to \mathbb{R}$ (the function itself is irrelevant for the example). The effect of the steps is the following.

Step 1. $k = 2, d = 6$, $\mathbf{d}_0 = (0,1)$, $(a_1, b_1) = (5,1)$, $(a_2, b_2) = (1,1)$, $(a_3, b_3) = (1,3)$, $(a_4, b_4) = (1,1)$, $(a_5, b_5) = (7,5)$, $(a_6, b_6) = (3,1)$. $M = 19, N = 13$.

Step 2. $(P_0, Q_0) = (6,0)$, $(P_1, Q_1) = (1,1)$, $(P_2, Q_2) = (0,2)$, $(P_2^*, Q_2^*) = (0,5)$, $(P_3, Q_3) = (1,8)$, $(P_4, Q_4) = (2,9)$, $(P_5, Q_5) = (9,14)$, $(P_6, Q_6) = (12,15)$.

Step 3 $\delta(p,q) = 1, g(p,q) = 0$ for $p = 0,1$ and $q = 2,3,4$. $\delta(p,q) = 0$ for all other points (p,q) of A.

Step 4 False.

Step 5. $r_0 = 5$, $r_1 = 0$, $r_6 = r_7 = r_8 = 0$, $r_9 = 1$, $r_{10} = 3$, $r_{11} = 4$, $r_{12} = 6$, $r_{13} = 7$, $r_{14} = 8$, $r_{15} = 11$.
$w(5,0) = .833$, $w(0,1) = .500$, $w(0,6) = .900$, $w(0,7) = .800$, $w(0,8) = .700$, $w(1,9) = .875$, $w(3,10) = .962$, $w(4,11) = .923$, $w(6,12) = .981$, $w(7,13) = .942$, $w(8,14) = .904$, $w(11,15) = .917$.
$s(0) = 1$, $s(1) = 2$, $s(6) = s(7) = s(8) = 3$, $s(9) = 4$, $s(10) = s(11) = s(12) = s(13) = s(14) = 5$, $s(15) = 6$, $s(5) = 0$ (See Fig. 6.7).

Step 6. $(p_0, q_0) = (0,1)$, $(p_1, q_1) = (0,8)$, $(p_2, q_2) = (0,7)$, $(p_3, q_3) = (5,0)$, $(p_4, q_4) = (1,9)$, $(p_5, q_5) = (0,6)$, $(p_6, q_6) = (8,14)$, $(p_7, q_7) = (11,15)$, $(p_8, q_8) = (4,11)$, $(p_9, q_9) = (7,13)$, $(p_{10}, q_{10}) = (3,10)$, $(p_{11}, q_{11}) = (6,12)$, $(p_{12}, q_{12}) = (0,5)$.

Step 7. See Fig. 6.7 for the order in which the f-values are computed, indicated by the black numbers. After this step the f-values of the first $m - M = 2$ columns are known and $M = 19$ columns are left.

Step 8. $\mathbf{d}_0 = (1,0)$, $(a_1, b_1) = (1,1)$, $(a_2, b_2) = (5,1)$, $(a_3, b_3) = (3,1)$, $(a_4, b_4) = (7,5)$, $(a_5, b_5) = (1,1)$, $(a_6, b_6) = (1,3)$.

Step 9. $(P_0, Q_0) = (0,2)$, $(P_1, Q_1) = (1,1)$, $(P_2, Q_2) = (6,0)$, $(P_2^*, Q_2^*) = (6,0)$, $(P_3, Q_3) = (9,1)$, $(P_4, Q_4) = (16,6)$, $(P_5, Q_5) = (17,7)$, $(P_6, Q_6) = (18,10)$.

Step 10. $r_0 = 1, r_1 = r_2 = \cdots = r_5 = 0$, $r_7 = r_8 = r_9 = 0$, $r_{10} = 1$, $r_{11} = 2$, $r_{12} = r_{13} = 3$, $r_{14} = 4$, $r_{15} = r_{16} = 5$, $r_{17} = 6$, $r_{18} = 9$.

$w(0,1) = .500$, $w(1,0) = .167$, $w(2,0) = .333$, $w(3,0) = .500$, $w(4,0) = .667$, $w(5,0) = .833$, $w(7,0) = .917$, $w(8,0) = .833$, $w(9,0) = .750$, $w(10,1) = .904$, $w(11,2) = .942$, $w(12,3) = .981$, $w(13,3) = .885$, $w(14,4) = .923$, $w(15,6) = .962$, $w(16,5) = .865$, $w(17,6) = .875$, $w(18,9) = .900$.
$s(0) = 1, s(1) = s(2) = \cdots = s(5) = 2$, $s(7) = s(8) = s(9) = 3$, $s(10) = s(11) = \cdots = s(16) = 4$, $s(17) = 5$, $s(18) = 6$, $s(6) = 0$.

Step 11 $(p_0, q_0) = (1,0)$, $(p_1, q_1) = (2,0)$, $(p_2, q_2) = (0,1)$, $(p_3, q_3) = (3,0)$, $(p_4, q_4) = (4,0)$, $(p_5, q_5) = (9,0)$, $(p_6, q_6) = (5,0)$, $(p_7, q_7) = (8,0)$, $(p_8, q_8) = (16,5)$, $(p_9, q_9) = (17,6)$, $(p_{10}, q_{10}) = (13,3)$, $(p_{11}, q_{11}) = (18,9)$, $(p_{12}, q_{12}) = (10,1)$, $(p_{13}, q_{13}) = (7,0)$, $(p_{14}, q_{14}) = (14,4)$, $(p_{15}, q_{15}) = (11,2)$, $(p_{16}, q_{16}) = (15,5)$, $(p_{17}, q_{17}) = (12,3)$, $(p_{18}, q_{18}) = (6,0)$.

Step 12. See Fig. 6.8 for the order in which the g-values are computed, indicated by the black numbers. This has to be continued in the obvious way. When this step has been completed all the g-values are known.

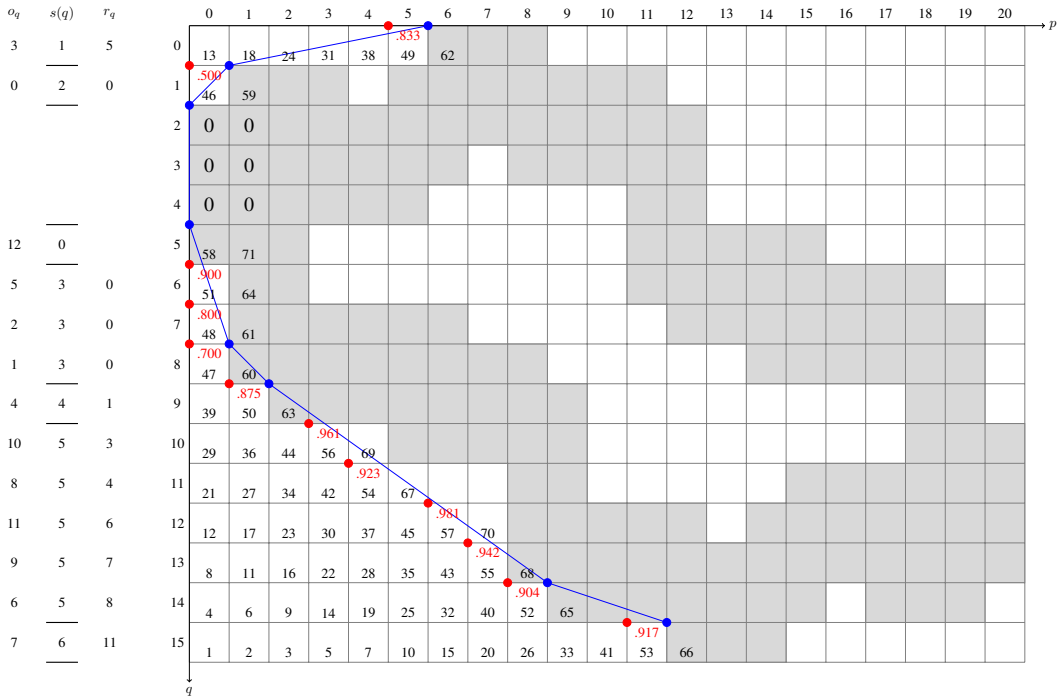


Figure 6.7: Steps 1-7 of Example 37. Grey pixels show the union of switching domains, and white pixels are uniquely determined.

Figure 6.7 Steps 1-7 of Example 37. The light grey pixels indicate the union of the switching domains. The white pixels indicate the pixels of which the f-values are unique, and therefore equal to the computed g-value. There are six primitive switching functions and their lexicographic smallest elements have a 0. In Step 3 their g-values are fixed as 0, but this may be replaced by any other values. Note that if these g-values are chosen to not be zero, then their contribution to the appropriate line sum must be subtracted. Step

1 guarantees the concavity of the upper left corner region V_{UL} and the lower left corner region V_{LL}, left of the blue border line. The border points for V_{UL} and V_{LL} are indicated by blue dots. They are found in Step 2. Step 4 provides a shortcut in case of a nonvalid case; if $m \leq M$, then the coordinates can be switched. For each row the grid point just left of the border line is computed in Step 5. The weights of these points are indicated in red. The red points are ordered after size as indicated in the column o_q (see Step 6). The function s indicates the directions which are used for the grid points in that row. Finally, in Step 7, the g-values are computed for the first $m - M$ columns and some more pixels. The order in which they are calculated is given in black.

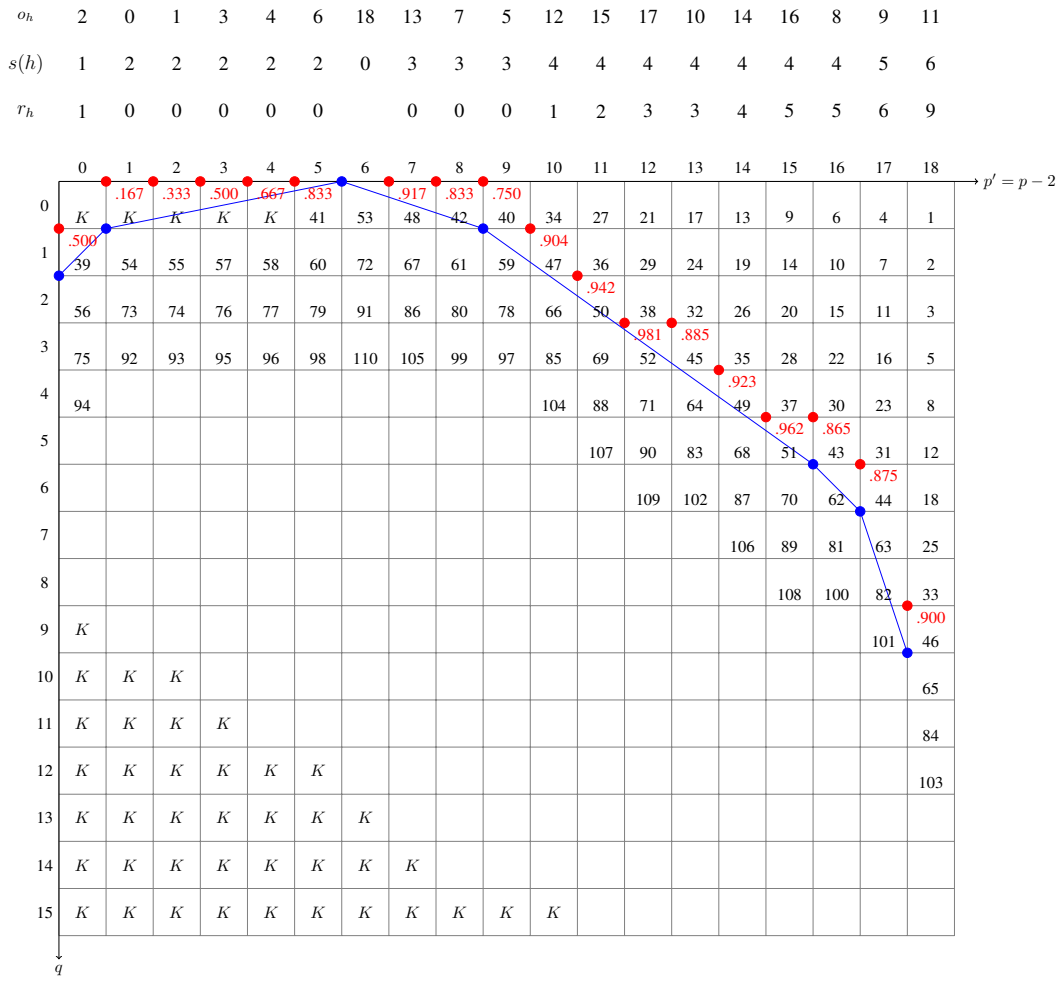


Figure 6.8: Steps 8-12 of Example 37, where the K's indicate the pixels of which the g-values have already been calculated in Step 7.

Figure 6.8 illustrates Steps 8-12 of Example 37. We have omitted the first $m - M = 2$ columns so that the column numbers indicate $p - 2$. The K's indicate the pixels of which the g-values have already been calculated in Step 7. In this stage the roles of rows and columns are interchanged. Step 8 serves to adjust the order of the directions. In Step 9 the border points in V_{UL} are

reordered, those in V_{UR} are those of V_{LL} mirrored. Again the blue border line connects them. This time the red points are the integer points immediately *above* the blue border points. Their Q-values, their weights (indicated in red) and the sequence s are computed in Step 10. They are ordered in Step 11 and the order is indicated in row o_h. Finally in Step 12 the remaining g-values are computed where by using u the original p-values are used instead of $p-2$. The order of the way the g-values are found is indicated by black numbers. This has to be completed downwards to find all g-values.

6.6 Complexity

In this section we claim that the complexity of the algorithm presented in Section 6.5 has linear time complexity in dmn for the majority of relevant cases. In some special cases, this is not strictly true. However, we present a modification to the algorithm in these cases whereby we can still achieve linear time reconstruction. We state the complexity of each step of the algorithm, where we count every addition, subtraction, multiplication, division and determining of the larger of two explicit quantities an elementary operation. We take the complexity of sorting a list of n elements to be $\mathcal{O}(n \log n)$. The order of complexity of the algorithm for each step is as follows:

- Step 1: $\mathcal{O}(d \log d)$;
- Step 2: $\mathcal{O}(d)$;
- Step 3: $\mathcal{O}(mn)$;
- Step 4: $\mathcal{O}(1)$;
- Steps 5 and 10: $\mathcal{O}(n)$ and $\mathcal{O}(m)$, respectively;
- Steps 6 and 11: $\mathcal{O}(n \log n)$ and $\mathcal{O}(m \log m)$, respectively;
- Steps 7+12: $\mathcal{O}(dmn)$.

Without loss of generality we may assume $m \leq n$. It follows that the complexity of the algorithm is $\mathcal{O}(dmn)$ unless both $d = \mathcal{O}(n)$ and $m = \mathcal{O}(n)$. In this exceptional case only the complexity of Step 6 has to be adjusted. This can be achieved by remembering in Step 5 for every H the location of the point (r_h, h) with $s(h) = H$ and minimal weight for direction (a_H, b_H). This has complexity $\mathcal{O}(dn)$.

Now Step 6 proceeds as follows. For each direction (a_H, b_H) let (r_h, h) be the point with $s(h) = H$ and minimal weight. For $H \leq k$ we start with the vectors

$$(r_h,\ h,\ b_H(P_H - 1) + a_H Q_H,\ b_H P_H + a_H Q_H,\ R_H,\ S_H,\ T_H,\ U_H),$$

where $(R_H, S_H) = (Q_{H-1}, Q_H - 1)$ and (T_H, U_H) is the unique pair satisfying $0 \le T_H < a_H$ and $b_H T_H + a_H U_H = 1$. For $k < H \le d$ we start with the vectors

$$(r_h,\ h,\ b_H(P_H-1)+a_H(n-1-Q_H),\ b_H P_H+a_H(n-1-Q_H),\ R_H, S_H, T_H, U_H),$$

where $(R_H, S_H) = (Q_{H-1} + 1, Q_H)$ and (T_H, U_H) is the unique pair satisfying $0 \le T_H < a_H$ and $b_H T_H - a_H U_H = 1$. Observe that in each case the quotient of the third and fourth entry is the weight.

We order such vectors on the top line according to increasing weight. At each step we increase by one the third entry of the first (leftmost) vector. If the third entry now is still smaller than the fourth entry, we replace the first two entries (r_h, h) by $(r_h+T_H, h+U_H)$ or $(r_h+T_H-b_H, h+U_H-a_H)$ such that the second entry is in (R_H, S_H). If the third entry becomes equal to the fourth one, we neglect the vector in the sequel. In any case we order the remaining vectors on the line again after increasing weight. This procedure runs until there is no vector left. At every step the two leftmost entries of the leftmost vector give the next value (p_h, q_h).

The computation of the vectors (T_H, U_H) has complexity $\mathcal{O}(d \log n)$, the computation of each row $\mathcal{O}(d \log d)$ and there are n rows. Therefore the total complexity is $\mathcal{O}(nd \log d)$. This is $\mathcal{O}(dmn)$, unless $m = \mathcal{O}(\log d)$. However, we have $a_h \ge 1$ with at most one exception. So if $m = \mathcal{O}(d)$, we can delete $d-m-1$ directions and still have a nonvalid case. After deletion we have, denoting by d the new number of directions, $m = d - 1$ and complexity $\mathcal{O}(dmn)$.

Example 38. [Continuation of Example 37]. The described procedure yields as the first row the vectors

$$(0,1,1,2,1,1,0,1),\quad (0,8,7,10,6,8,0,-1),\quad (5,0,5,6,0,0,1,0),$$
$$(1,9,7,8,9,9,1,0),\quad (8,14,47,52,10,14,3,2),\quad (11,15,11,12,15,15,1,0).$$

Since the last four entries do not change, we do not mention them in the table. Then the table becomes as follows (each row represents one step in the procedure).

(0,1,1,2)	(0,8,7,10)	(5,0,5,6)	(1,9,7,8)	(8,14,47,52)	(11,15,11,12)
(0,8,7,10)	(5,0,5,6)	(1,9,7,8)	(8,14,47,52)	(11,15,11,12)	
(0,7,8,10)	(5,0,5,6)	(1,9,7,8)	(8,14,47,52)	(11,15,11,12)	
(5,0,5,6)	(1,9,7,8)	(0,6,9,10)	(8,14,47,52)	(11,15,11,12)	
(1,9,7,8)	(0,6,9,10)	(8,14,47,52)	(11,15,11,12)		
(0,6,9,10)	(8,14,47,52)	(11,15,11,12)			
(8,14,47,52)	(11,15,11,12)				
(11,15,11,12)	(4,11,48,52)				
(4,11,48,52)					
(7,13,49,52)					
(3,10,50,52)					
(6,12,51,52)					

The sequence $(p_0, q_0) = (0, 1)$, $(p_1, q_1) = (0, 8)$, $(p_2, q_2) = (0, 7)$, $\ldots$, $(p_{11}, q_{11}) = (6, 12)$ can be read from the leftmost two entries. At the end $(p_{12}, q_{12}) = (0, 5)$ has to be added.

6.7 Reconstruction of 3D Boundary Ghosts

In this section, we extend the reconstruction algorithm to three dimensions for the case of boundary ghosts. By exploiting the special structure of boundary ghosts domains, the three dimensional situation can be reduced to the two dimensional one. This is due to the fact that 3D ghosts are comprised of a stack of 2D ghosts. Therefore we can compute the function values of the pixels inside the boundary ghost domain in linear time. We will illustrate this with an example.

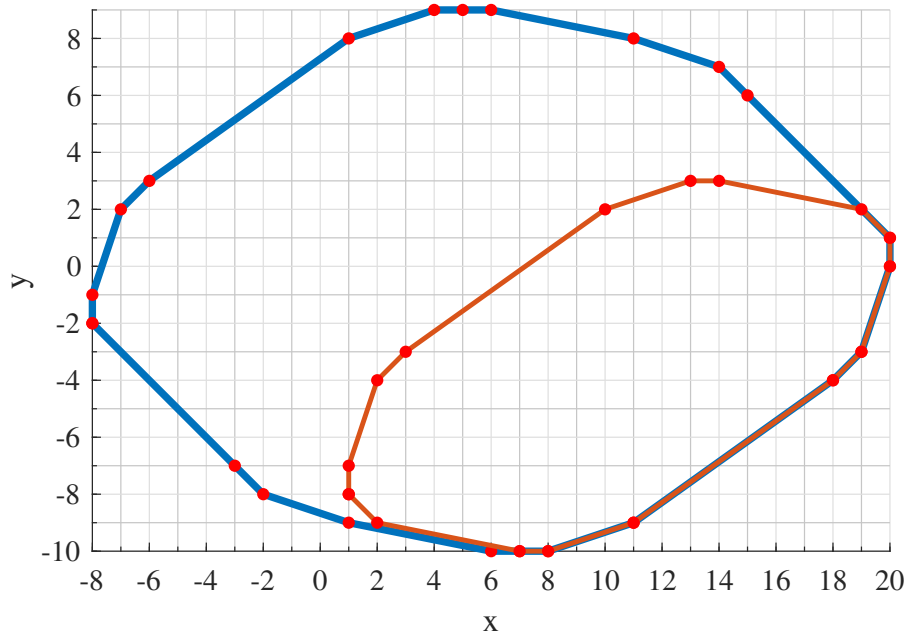


Figure 6.9: Convex hull of a boundary ghost projected onto the x-y plane (blue). Convex hull of the boundary ghost in the plane $z = 5$ (red).

Example 39. Consider the boundary ghost generated by the directions: $D = \{(0,1,0),\ (1,0,0),\ (1,1,0),\ (1,0,1),\ (-1,1,0),\ (-3,-1,0),\ (-3,1,-2),\ (-1,-3,0),\ (5,-1,0),\ (5,-5,4),\ (7,5,0)\}$.

We apply the method to the block $[-8,20]\times[-10,9]\times[-2,5]$ which contains a primitive boundary ghost for D. We give function values 0 to all integer points outside this block. The first observation is that if we have computed all function values for points with $z = 5$ we are left with a similar or even simpler problem and can apply the method to compute all function values for integer points with $z = 4$ and so on. Figure 6.9 shows the convex hull of the boundary ghost seen "from above". The blue contour is the projection of the convex hull parallel to the z-axis. The red contour indicates the intersection of the convex hull with the plane $z = 5$.

The first step is to consider the projection of the convex hull of the ghost in the negative y-direction. See Fig. 6.10. The blue lines indicates the convex

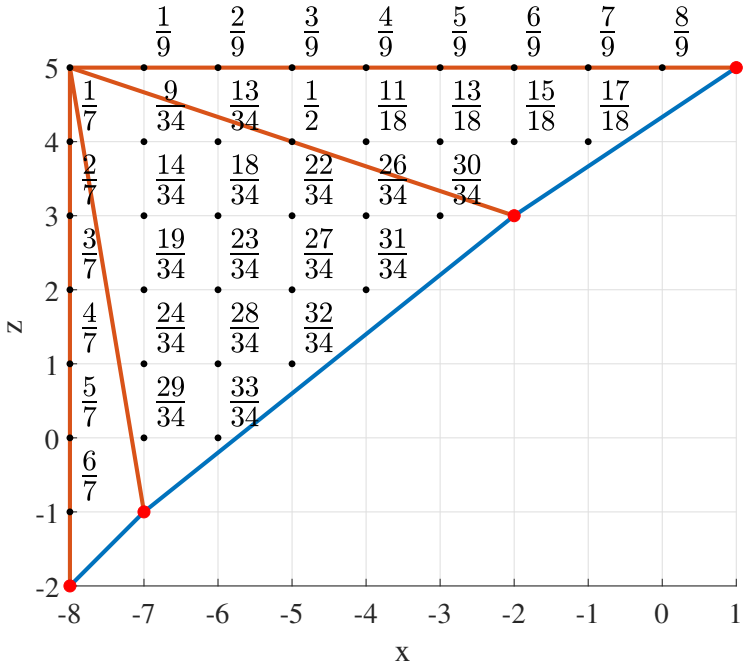


Figure 6.10: Projection of the convex hull onto the x-z plane along the negative y-direction. Weights are given for points outside of the convex hull.

hull of the boundary ghost. It consists of three segments in the directions $(1, 1)$ of $(1, 0, 1)$, $(5, 4)$ of $(5, -5, 4)$ and $(3, 1)$ of $(3, -1, 2)$. The points where these line segments meet, $(-7, -1)$ and $(-2, -3)$, are connected by red straight line segments with the corner point $(-8, -5)$. The weight is given for each integer point P left of the blue lines, which is the quotient of the distance of the corner point $(-8, 5)$ to P and the distance of that corner point to the intersection of the line through the corner point and P and the blue line. For example, $(-5, 4)$ is in the middle of $(-8, 5)$ and $(-2, 3)$ and so its weight is $1/2$. The theory in [52] is applicable here, independent of the value of y: If you order the points P with increasing weight, then each point in turn is the only point with unknown function value with respect to the direction of the corresponding line segment. For instance, the point $(-6, 0)$ with weight $33/34$ is in the red triangle with direction $(5, 4)$, and the other point on the line with this direction is $(-1, 4)$. But its function value is already known since $17/18 < 33/34$ and therefore we can compute the function value of $(-6, 0)$. The point $(-1, 4)$ is in the triangle with direction $(3, 2)$ and on the line through $(-1, 4)$ in this direction we have the points $(-4, 2)$ and $(-7, 0)$. Their weights are smaller than the weight of $(-1, 4)$, and so on. The truth of the claim can be checked numerically in this special case, but can also been seen geometrically by using that the blue lines are convex. In this step all function values of the points $(p, *, r)$ with (p, r) left of the blue lines can be computed. In particular the function values of the points $(p, q, 5)$ with $-8 \leq p \leq 0$ and $*$ any value are known after this step.

The second step is the corresponding operation with respect to the y-z-

plane, see Fig. 6.11. Since we are interested in the function values of points with $z = 5$, we consider the corner region of $(9, 5)$. The blue line consists of line segments with directions $(0, 1)$ from $(1, 0, 1)$, $(-1, 2)$ from $(3, -1, 2)$ and $(-5, 4)$ from $(5, -5, 4)$. The weights of the integer points right of the blue lines are written in red. Note that many of them are the only points on the line with direction $(-5, 4)$ so that their function values are equal to the line sum such that the "weight argument" has to be applied to only few points. We conclude that in this step all function values can be computed of the points $(*, q, 5)$ with $4 \leq q \leq 9$ and $*$ any value.

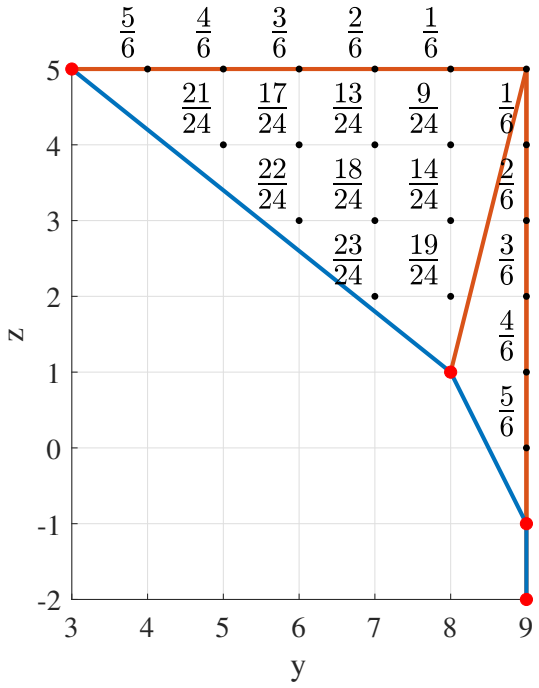


Figure 6.11: Convex hull in the y-z-plane, with weights.

The third and last step is two-dimensional, see Fig. 6.12. The black points indicate the points with already known function values. What is left is the rectangle $\{(p, q, 5) : 1 \leq p \leq 20, -10 \leq q \leq 2\}$. The directions left have all z-value 0, which are $(0, 1, 0)$, $(1, 0, 0)$, $(1, 1, 0)$, $(-3, -1, 0)$, $(-1, -3, 0)$, $(5, -1, 0)$, $(7, 5, 0)$. Neglecting the z-values, algorithm 4 can be applied to compute the remaining function values for points $(p, q, 5) \in A$.

This method can be applied to any set of directions that generates a stack of 2D ghosts. In this case, we can always reduce the three dimensional problem to a series of problems in two dimensions. This allows us to reconstruct the function in linear time. The same process can be applied in an arbitrary number of dimensions if the k-dimensional ghost consists of stacks of two dimensional ghosts.

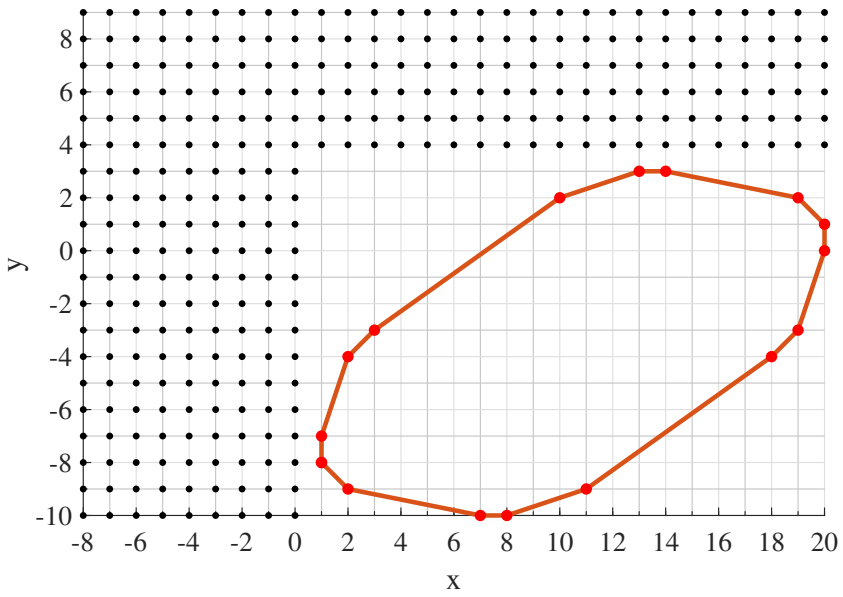


Figure 6.12: Intersection of the convex hull with the plane $z = 5$. The black points have known function values.

6.8 Summary

In this chapter we have addressed the tomographic reconstruction problem for functions with values in a unique factorization domain, such as integers and reals. A key argument is that one may ask for the point values even when many solutions are admissible, since the values of points outside the switching domains are common to all functions satisfying the problem.

Starting from the characterization of the switching functions in [44] and the results in [52], we have shown that all points with uniquely determined value, namely, not belonging to switching domains, can be recovered once we give an arbitrary value to $(m - M)(n - N)$ points, where $(m - M)(n - N)$ is the number of linearly independent switching functions. We have provided an algorithm, running in time linear in dmn, which computes the point values systematically. By the result in [44] our algorithm provides the complete set of solutions with values in the unique factorization domain.

The proposed approach works when line sums are supposed to be exact and therefore not all projections are necessary to recover a solution. It underlines the structural difference between unique factorization domains and other kinds of sets, such as $\{0, 1\}$ (leading to binary images), since in the latter case the reconstruction problem has proven to be NP-hard [32, 33].

Two questions arise from this chapter. Firstly, does there exist a general algorithm for higher dimensions? Secondly, given a system of inconsistent line sums, is there a fast way to find a best approximation of consistent line sums so that the algorithm here can be applied to construct the most likely set of solutions (over $\mathbb{Z}$ or $\mathbb{R}$)?

CHAPTER 7

Conclusion

This thesis demonstrates how to use techniques from discrete tomography to produce highly structured arrays that are well suited to a wide range of applications. We have explored the placement of non-random points on a grid, and the effect these structured arrays can have on correlations and tomography. As discrete tomography may be exact in the absence of noise, the arrays we produce are readily transferable to digital technologies. The Mojette transform is already employed in areas such as network coding, distributed storage, compression and cyber security. Hence, the arrays constructed in this work can be used in communications applications, encryption and watermarking.

The finite Radon transform was employed to construct so-called perfect arrays. These arrays are built on prime $p \times p$ lattices and have ideal correlation properties. The alphabet of these arrays was deliberately confined to the values $\{-1, 0, +1, +2\}$, and the number of times each of these values occur in a $p \times p$ array are $\{(p-1)^2/2, 3(p-1)/2, (p-1)^2/2+1, (p-1)/2\}$ respectively. Thus, the arrays are dominated by ± 1 values, with relatively few zeros present (for greater efficiency and less redundancy). Affine transformations can be applied to periodically shuffle the array values without changing their correlative power, and hence produce families of $p^2 - 1$ arrays which all have ideal auto-correlation, and low cross-correlation.

It was later discovered that the projection process used to construct these arrays produced quadratic residue sequences, through the pattern of the chosen shifts for each projection. This realisation was then used to construct perfect arrays in any number of dimensions. Again, affine transformations can be applied to produce a large family of arrays in higher dimensions. Through an extension of the process applied to the 2D arrays, we can extend nD families of arrays with prime side length p to at least order $\mathcal{O}(p^n)$. However, it may be possible to extend the number of family members further.

Random matrix theory has yielded many fruitful results in physics, due to the favourable correlation properties of signals that resemble white noise [8]. We have shown that the designed arrays in this work outperform random arrays with respect to correlation. Therefore, perfect arrays may be useful in some fields where random matrix theory is currently applied. Future work will look to find applications for perfect arrays which may supersede random arrays. The primary impediment for using perfect arrays up to this juncture was that it was difficult to produce a large number of unique arrays. However, as we have shown here, it is possible to create families of $p^2 - 1$ arrays.

This thesis also introduced a method of constructing new types of tomographic ghost arrays. The recursion $v_{n+1} = v_n + 2\epsilon_{n+1}v_{n-1}$ with $v_0 = (1, 0)$, $v_1 = (1, 1)$, and $\epsilon_{n+1} \in \{-1, 1\}$, was employed to build a sequence of discrete projection directions that define arrays which sum to zero across all specified directions. This sequence of directions produces maximal ghosts that consist purely of 2^N connected points that have a value of either -1 or $+1$ for N directions. The presence of the ϵ_{n+1} element produces a possible bifurcation of directions at each step, so there are many possible distinct sequences of directions. As the number of points doubles with each additional direction, these ghosts must not have any overlapping points. With the constraint that the adjacent ghost copies cannot have any gaps between them, the question of assembling these points was tackled as a tiling problem. As was proven, the recursion defines a sequence of non-degenerate directions that interlocks self-similar tiles in a shape that resembles a fractal.

Computationally, it was discovered that the interlocking that occurs naturally through this sequence is in fact maximal. This gives credence to the hypothesis that this recursion may arise in nature in the formation of lattices. A famous example of this sort of phenomenon is the spiral configuration of seeds or petals being modelled by the Fibonacci numbers. Future research will look at the physical application of this sequence in the context of crystals and other lattice structures.

Maximal ghosts have a very specific internal structure. Each column alternates between values of -1 and $+1$. Ghost points have value $+1$ for even x and value -1 to those with odd x. This structure can be exploited to create boundary ghosts. When we add the direction $v_{-1} = (0, 1)$ to a maximal ghost, all internal points cancel out, leaving only a thin contour, or boundary, of ghost points. For the purposes of tomography, all ghost points flag the sites of possible image reconstruction errors. Having these errors placed near the boundary of a reconstructed image is favourable when the object of interest lies in the centre of the domain.

The 2D ghost tiles can be stacked along a z-direction to create 3D maximal ghosts. These 3D ghosts therefore carry all of the properties of 2D ghosts along each integer x-y plane. The same process can be used to stack 2D ghosts along any number of dimensions, generating k-dimensional ghosts. The additional degrees of freedom in these higher dimensions allows for a wider range of

boundary ghosts to be derived from each maximal ghost. It is possible to construct 2^{k-2} boundary ghosts for dimension k.

Finally, the process of reconstruction is addressed. Previously, it had been shown that the problem of reconstruction in discrete tomography is NP-hard. However, in this work we proved that the tomographic reconstruction problem can be simplified for functions with values in a unique factorization domain, such as the integers and reals. A reconstruction algorithm is presented that operates in linear time with respect to the grid size and number of directions. The algorithm requires the projections to be free of noise, but does not assume the absence of switching domains or ghosts. Thus, a solution can be obtained in linear time even if it is not unique. Since the 3D ghosts constructed in this work consist of 2D ghosts, it is possible to use this algorithm in higher dimensions for the special case of maximal and boundary ghost directions.

Subsequent research will look to generalise this algorithm to the three dimensional case and beyond. This algorithm can detect the presence of noise, but cannot produce a useful solution in most cases when the line sums are not exact. Hence, it would be useful to investigate a similar algorithm that can deal with noise to obtain a likely solution.

Structured placement of points on a grid has proven to be useful in a vast range of discrete and digital contexts. The properties of discrete projection greatly aid in choosing the placement of these points. Operating in this discrete environment has produced many fruitful results that would not have been possible by simply approximating a continuous domain. And while the results in this thesis are inherently discrete, they may be extrapolated to prove useful in many physical settings, in addition to the digital applications outlined. There is much to be explored in this regard, and this will be the focus of future research.

Bibliography

[1] Alpers, A., and Larman, D. G. The smallest sets of points not determined by their X-rays. *Bulletin of the London Mathematical Society 47*, 1 (2015), 171–176.

[2] Alpers, A., and Tijdeman, R. The two-dimensional Prouhet–Tarry–Escott problem. *Journal of Number Theory 123* (2007), 403–412.

[3] Andersen, A. H., and Kak, A. C. Simultaneous algebraic reconstruction technique (SART): A superior implementation of the ART algorithm. *Ultrasonic Imaging 6*, 1 (1984), 81–94.

[4] Autrusseau, F. *Modélisation psychovisuelle pour le tatouage des images*. PhD thesis, 2002.

[5] Autrusseau, F., Guedon, J. V., and Bizais, Y. Mojette cryptomarking scheme for medical images. In *Medical Imaging 2003: Image Processing* (2003), vol. 5032, International Society for Optics and Photonics, pp. 958–965.

[6] Batenburg, K., and Plantagie, L. Fast approximation of algebraic reconstruction methods for tomography. *IEEE Transactions on Image Processing 21*, 8 (2012), 3648–3658.

[7] Batenburg, K. J., and Sijbers, J. DART: A practical reconstruction algorithm for discrete tomography. *IEEE Transactions on Image Processing 20*, 9 (2011), 2542–2553.

[8] Brézin, É., Kazakov, V., Serban, D., Wiegmann, P., and Zabrodin, A. *Applications of random matrices in physics*, vol. 221. Springer Science & Business Media, 2006.

[9] Brunetti, S., Dulio, P., Hajdu, L., and Peri, C. Ghosts in discrete tomography. *Journal of Mathematical Imaging and Vision 53* (2015), 210–224.

[10] Brunetti, S., Dulio, P., and Peri, C. Discrete tomography determination of bounded lattice sets from four X-rays. *Discrete Applied Mathematics 161*, 15 (2013), 2281–2292.

[11] Cahill, N. D., D Errico, J. R., and Spence, J. P. Complex factorizations of the Fibonacci and Lucas numbers. *Fibonacci Quarterly 41*, 1 (2003), 13–19.

[12] Cavy, B., and Svalbe, I. Construction of perfect auto-correlation arrays and zero cross-correlation arrays from discrete projections. In *International Workshop on Combinatorial Image Analysis* (2015), Springer, pp. 232–243.

[13] Ceko, M., Pagani, S., and Tijdeman, R. Algorithms for linear time reconstruction by discrete tomography II. Submitted for publication.

[14] Ceko, M., Petersen, T., Svalbe, I., and Tijdeman, R. Boundary ghosts for discrete tomography. *Journal of Mathematical Imaging and Vision* (2021), 1–13.

[15] Ceko, M., Svalbe, I., and Petersen, T. A discrete projection analogue to pick's theorem. *Graphical Models* (2020), 101066.

[16] Ceko, M., Svalbe, I., Petersen, T., and Tirkel, A. Large families of "grey" arrays with perfect auto-correlation and optimal cross-correlation. *Journal of Mathematical Imaging and Vision 61*, 2 (2019), 237–248.

[17] Chandra, S., Normand, N., Kingston, A., Guédon, J., and Svalbe, I. Robust digital image reconstruction via the discrete Fourier slice theorem. *IEEE Signal Processing Letters 21*, 6 (2014), 682–686.

[18] Chandra, S., and Svalbe, I. A fast number theoretic finite Radon transform. In *Digital Image Computing: Techniques and Applications, 2009. DICTA'09.* (2009), IEEE, pp. 361–368.

[19] Chandra, S. S., Svalbe, I. D., Guédon, J., Kingston, A. M., and Normand, N. Recovering missing slices of the discrete Fourier transform using ghosts. *IEEE Transactions on Image Processing 21* (2012), 4431–4441.

[20] Cormack, A. M. Representation of a function by its line integrals, with some radiological applications. *Journal of Applied Physics 34*, 9 (1963), 2722–2727.

[21] Cormack, A. M. Representation of a function by its line integrals, with some radiological applications II. *Journal of Applied Physics 35*, 10 (1964), 2908–2913.

[22] Cox, I. J., and Miller, M. L. Review of watermarking and the importance of perceptual modeling. In *Human Vision and Electronic Imaging II* (1997), vol. 3016, International Society for Optics and Photonics, pp. 92–100.

[23] Dalen, B. v., Hajdu, L., and Tijdeman, R. Bounds for discrete tomography solutions. *Indagationes Mathematicae 24*, 2 (2013), 391–402.

[24] Dulio, P., Frosini, A., and Pagani, S. A geometrical characterization of regions of uniqueness and applications to discrete tomography. *Inverse Problems 31*, 12 (2015), 125011.

[25] Dulio, P., Frosini, A., and Pagani, S. Geometrical characterization of the uniqueness regions under special sets of three directions in discrete tomography. In *Discrete Geometry for Computer Imagery*, N. Normand, J. Guédon, and F. Autrusseau, Eds., vol. 9647 of *Lecture Notes in Comput. Sci.* Springer, Cham, 2016, pp. 105–116.

[26] Dulio, P., Frosini, A., and Pagani, S. Regions of uniqueness quickly reconstructed by three directions in discrete tomography. *Fundamenta Informaticae 155*, 4 (2017), 407–423.

[27] Dulio, P., Frosini, A., and Pagani, S. M. A geometrical characterization of regions of uniqueness and applications to discrete tomography. *Inverse Problems 31*, 12 (2015), 125011.

[28] Dulio, P., and Pagani, S. A rounding theorem for unique binary tomographic reconstruction. *Discrete Applied Mathematics 268* (2019), 54–69.

[29] Dulio, P., and Peri, C. On the geometric structure of lattice U-polygons. *Discrete Mathematics 307*, 19 (2007), 2330–2340.

[30] Fishburn, P., Lagarias, J., Reeds, J., and Shepp, L. Sets uniquely determined by projections on axes. II. Discrete case. *Discrete Mathematics 91*, 2 (1991), 149–159.

[31] Gardner, R., and Gritzmann, P. Discrete tomography: Determination of finite sets by X-rays. *Transactions of the American Mathematical Society 349* (1997), 2271–2295.

[32] Gardner, R., Gritzmann, P., and Prangenberg, D. On the computational complexity of reconstructing lattice sets from their X-rays. *Discrete Mathematics 202*, 1-3 (1999), 45–71.

[33] Gardner, R., Gritzmann, P., and Prangenberg, D. On the computational complexity of determining polyatomic structures by X-rays. *Theoretical Computer Science 233*, 1-2 (2000), 91–106.

[34] Golomb, S., and Gong, G. *Signal design for good correlation: For wireless communication, cryptography, and radar*. Cambridge University Press, 2005.

[35] Golomb, S., and Taylor, H. Two-dimensional synchronization patterns for minimum ambiguity. *IEEE Transactions on Information Theory 28*, 4 (1982), 600–604.

[36] Golomb, S. W. Replicating figures in the plane. *The Mathematical Gazette 48* (1964), 403–412.

[37] Gómez-Pérez, D., Gómez, A. I., and Tirkel, A. Large families of sequences for CDMA, frequency hopping, and UWB. *Cryptography and Communications* (2019), 1–15.

[38] Gordon, R., Bender, R., and Herman, G. T. Algebraic reconstruction techniques (ART) for three-dimensional electron microscopy and X-ray photography. *Journal of Theoretical Biology 29*, 3 (1970), 471–481.

[39] Gottesman, S. R., and Fenimore, E. New family of binary arrays for coded aperture imaging. *Applied Optics 28*, 20 (1989), 4344–4352.

[40] Guédon, J. *The Mojette transform: Theory and Applications*. ISTE John Wiley & Sons, 2009.

[41] Guédon, J.-P., Parrein, B., and Normand, N. Internet distributed image information system. *Integrated Computer-Aided Engineering 8*, 3 (2001), 205–214.

[42] Gürsoy, D., De Carlo, F., Xiao, X., and Jacobsen, C. Tomopy: a framework for the analysis of synchrotron tomographic data. *Journal of Synchrotron Radiation 21*, 5 (2014), 1188–1193.

[43] Hajdu, L. Unique reconstruction of bounded sets in discrete tomography. In *Proceedings of the Workshop on Discrete Tomography and its Applications* (2005), vol. 20 of *Electronic Notes in Discrete Mathematics*, Elsevier, Amsterdam, pp. 15–25 (electronic).

[44] Hajdu, L., and Tijdeman, R. Algebraic aspects of discrete tomography. *Journal fur die Reine und Angewandte Mathematik 534* (2001), 119–128.

[45] Herman, G. *Fundamentals of computerized tomography: Image reconstruction from projections*, 2nd ed. Springer, 2009.

[46] Kaczmarz, S. Approximate solution of systems of linear equations. *International Journal of Control 57*, 6 (1993), 1269–1271.

[47] Katz, M. Questions of uniqueness and resolution in reconstruction from projections. In *Lecture Notes in Biomathematics* (1978), vol. 26, Springer-Verlag, Berlin, Heidelberg.

[48] Leukhin, A., and Potekhin, E. A Bernasconi model for constructing ground-state spin systems and optimal binary sequences. In *Journal of Physics: Conference Series* (2015), vol. 613, IOP Publishing, p. 012006.

[49] Matúš, F., and Flusser, J. Image representation via a finite Radon transform. *IEEE Transactions on Pattern Analysis and Machine Intelligence 15*, 10 (1993), 996–1006.

[50] Natterer, F. *The mathematics of computerized tomography*. SIAM, 2001.

[51] Normand, N., Kingston, A., and Évenou, P. A geometry driven reconstruction algorithm for the Mojette transform. In *International Conference on Discrete Geometry for Computer Imagery* (2006), Springer, pp. 122–133.

[52] Pagani, S., and Tijdeman, R. Algorithms for linear time reconstruction by discrete tomography. *Discrete Applied Mathematics 271* (2019), 152–170.

[53] Pelt, D., and Batenburg, K. Fast tomographic reconstruction from limited data using artificial neural networks. *IEEE Transactions on Image Processing 22*, 12 (2013), 5238–5251.

[54] Pertin, D. Mojette erasure code for distributed storage (Ph.D Thesis). *University of Nantes* (2016).

[55] Phillipé, O. Image representation for joint source channel coding for QoS networks (Ph.D Thesis). *University of Nantes* (1998).

[56] Pick, G. Geometrisches zur zahlenlehre. *Sitzenber. Lotos (Prague) 19* (1899), 311–319.

[57] Pless, V. S., Huffman, W., and Brualdi, R. A. *Handbook of coding theory*, vol. 1. Elsevier Amsterdam, 1998.

[58] Radon, J. Über die bestimmung von funktionen durch ihre integralwerte längs gewisser mannigfaltigkeiten. *Ber. Verh. Sächs. Akad. Wiss. Leipzig Math.-Phys. Kl.*, 69 (1917), 262–277.

[59] Radon, J. On the determination of functions from their integral values along certain manifolds. *IEEE Transactions on Medical Imaging 5*, 4 (1986), 170–176.

[60] Reeve, J. E. On the volume of lattice polyhedra. *Proceedings of the London Mathematical Society 3*, 1 (1957), 378–395.

[61] Ryser, H. J. Combinatorial properties of matrices of zeros and ones. In *Classic Papers in Combinatorics*. Springer, 2009, pp. 269–275.

[62] Servières, M., Idier, J., Normand, N., and Guedon, J.-P. Conjugate gradient Mojette reconstruction. In *Medical Imaging 2005: Image Processing* (2005), vol. 5747, International Society for Optics and Photonics, pp. 2067–2074.

[63] Stawski, T. M., Van Driessche, A. E., Ossorio, M., Rodriguez-Blanco, J. D., Besselink, R., and Benning, L. G. Formation of calcium sulfate through the aggregation of sub-3 nanometre primary species. *Nature Communications 7*, 1 (2016), 1–9.

[64] Svalbe, I. Sampling properties of the discrete Radon transform. *Discrete Applied Mathematics 139*, 1 (2004), 265–281.

[65] Svalbe, I., and Ceko, M. Maximal N-ghosts and minimal information recovery from N projected views of an array. In *International Conference on Discrete Geometry for Computer Imagery* (2017), Springer, pp. 135–146.

[66] Svalbe, I., Ceko, M., and Tirkel, A. Large families of "grey" arrays with perfect auto-correlation and optimal cross-correlation. In *International Conference on Discrete Geometry for Computer Imagery* (2017), Springer, pp. 46–56.

[67] Svalbe, I., Nazareth, N., Normand, N., and Chandra, S. On constructing minimal ghosts. In *Digital Image Computing: Techniques and Applications (DICTA), 2010 International Conference on* (2010), IEEE, pp. 276–281.

[68] Svalbe, I., and Tirkel, A. Extended families of 2D arrays with near optimal auto and low cross-correlation. *EURASIP Journal on Advances in Signal Processing 2017*, 1 (2017), 18.

[69] Swanson, M. D., Kobayashi, M., and Tewfik, A. H. Multimedia data-embedding and watermarking technologies. *Proceedings of the IEEE 86*, 6 (1998), 1064–1087.

[70] Tirkel, A., Cavy, B., and Svalbe, I. Families of multi-dimensional arrays with optimal correlations between all members. *Electronics Letters 51*, 15 (2015), 1167–1168.

[71] Tirkel, A. Z., Rankin, G., Van Schyndel, R., Ho, W., Mee, N., and Osborne, C. F. Electronic watermark. *Digital Image Computing, Technology and Applications (DICTA'93)* (1993), 666–673.

[72] Van Aarle, W., Palenstijn, W. J., Cant, J., Janssens, E., Bleichrodt, F., Dabravolski, A., De Beenhouwer, J., Batenburg, K. J., and Sijbers, J. Fast and flexible X-ray tomography using the ASTRA toolbox. *Optics Express 24*, 22 (2016), 25129–25147.

[73] Van Aarle, W., Palenstijn, W. J., De Beenhouwer, J., Altantzis, T., Bals, S., Batenburg, K. J., and Sijbers, J. The ASTRA toolbox: A platform for advanced algorithm development in electron tomography. *Ultramicroscopy 157* (2015), 35–47.

[74] Van Aert, S., Batenburg, K. J., Rossell, M. D., Erni, R., and Van Tendeloo, G. Three-dimensional atomic imaging of crystalline nanoparticles. *Nature 470* (2011), 374.

[75] Van Schyndel, R. G., Tirkel, A. Z., and Osborne, C. F. A digital watermark. In *Proceedings of 1st international conference on image processing* (1994), vol. 2, IEEE, pp. 86–90.

[76] Varga, L., Balázs, P., and Nagy, A. Projection selection dependency in binary tomography. *Acta Cybernetica 20* (2011), 167–187.

[77] Verbert, P. *Sur la redondance des transformations Mojette en dimension n et en ligne*. PhD thesis, Nantes, 2004.

[78] Viterbi, A. J. *CDMA: Principles of spread spectrum communication*, vol. 122. Addison-Wesley Reading, MA, 1995.

[79] Waldron, S. A sharpening of the Welch bounds and the existence of real and complex spherical t–designs. *IEEE Transactions on Information Theory 63*, 11 (2017), 6849–6857.

[80] Welch, L. Lower bounds on the maximum cross correlation of signals. *IEEE Transactions on Information Theory 20*, 3 (1974), 397–399.