Chapter 5

Reconstruction of Non-Cartesian Data

5.1 Introduction

There are many alternatives to spin-warp, or 2DFT, acquisition methods. These include spiral scans, radial scans, variations of echo-planar, and many other less common acquisitions (rosette, Lissajou, ...). All of these are relatively easy to program on modern commercial imaging systems. Many of these have specific advantages over spin-warp, such as speed, flow performance, and SNR efficiency. The main disadvantage with these methods is the difficulty of reconstructing the resulting data sets. This chapter concerns methods for reconstructing these data sets that do not fall on a regular Cartesian grid in spatial-frequency space. This is a very important topic. Once the reconstruction limitations are removed, there are many more options for collecting MRI data.

One of the most important advantages of spin-warp, is that the reconstruction can be performed with a simple 2D DFT. This is easily implemented, and very efficient. For non-Cartesian data sets there are many options. The first approach is to collect the non-Cartesian data in a way that a previously known reconstruction method, like projection reconstruction, can be applied. While this solves the reconstruction problem, it usually requires compromises in data acquisition. Second, the non-Cartesian data can be demodulated point-by-point with the conjugate phase reconstruction. This works, but is slow. Better approaches involve first resampling the data to a Cartesian grid, and then using a 2DFT for the reconstruction. There are many different approaches to resampling, which will be our main concern here.

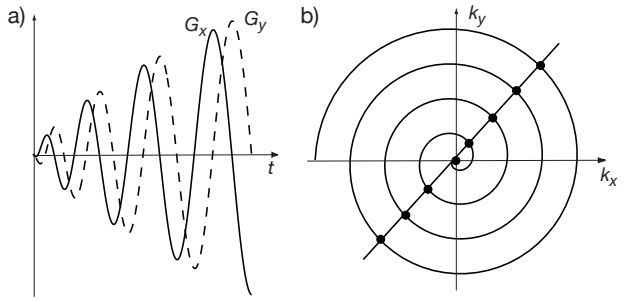

Figure 5.1: Sampling of a constant angular rate spiral (a) at an even integer number of samples per rotation results in the samples falling on diameters in k-space. The 1D Fourier transform along each diameter is a projection at that angle, by the central section theorem. The image can then be reconstructed via backprojection.

5.2 Reconstruction Using Specific Geometries

Originally MRI was done with projection-reconstruction, and then later with spin-warp. Hence when methods like spirals came along, they were adapted to fit within the well known projection-reconstruction framework. This was done by using a spiral trajectory that rotates at a constant angular rate. Then, if the number of

samples per rotation is constant and even, all of the samples line up on diameters, as is shown in Fig. 5.1. The data along a diameter is the transform of a projection, by the central section theorem. Hence, a full set of projections can be computed, and the image reconstructed using backprojection, as in X-ray CT. The drawback of this approach is that a constant angular rate spiral does not make efficient use of the gradient system performance, and typically results in acquisitions that are $\sqrt{2}$ longer than would be required for a constant slew rate spiral.

Another example where an acquisition is fit into a specific geometry for reconstruction is echo-planar imaging. In conventional EPI, the slow phase encode axis is applied continuously as the fast axis oscillates. The result is a sinusoidal trajectory through k-space. For reconstruction convenience it is assumed the trajectory actually follows a raster scan, which results in ghosting artifacts from objects with high spatial frequencies in x. To improve this, blipped EPI uses a blipped phase-encode gradient, so that the k_x–k_y trajectory is in fact a raster scan. However, this still leaves problems with off-resonance, which has to follow the conventional EPI trajectory, since off-resonance precession happens continuously. A potentially better alternative is to use a conventional EPI trajectory, where y and the off-resonance frequency ω are indistinguishable, and reconstruct using the true non-Cartesian reconstruction trajectory. Then off-resonance produces only geometric distortion, without ghosting.

5.3 Sensitive Point and Conjugate Phase Reconstruction

Some of the early imaging methods could not easily be fit into the framework of known reconstruction methods. One of the earliest methods for MRI imaging was called "sensitive point."The basic idea

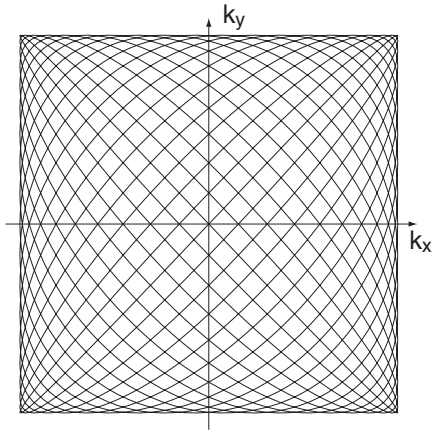

Figure 5.2: Lissajou k-space trajectory.

was to continuously vary the gradients in three dimensions. The point at the isocenter of the gradients sees no change, and continues to produce a constant signal thoughout the experiment. All other points see some time-varying field. If the gradient waveforms are properly chosen, the signal from these other points will integrate to zero, while the signal at the origin will integrate coherently to a value proportional to the magnetization there. In the original implementation, the subject was translated in the magnet to move a new voxel to isocenter, and the process repeated.

For sensitive point imaging, the sinusoidal gradients are applied. In multiple dimensions the frequencies of the sinusoids are chosen to be relatively prime. Otherwise the trajectory can retrace itself, and not all of k-space will be covered with the desired density. A sample Lissajou pattern k-space trajectory is shown in Fig. 5.2

The k-space trajectory is

$$k_x(t) = k_{max} \sin(2\pi\Omega_x t/T) \quad (5.1)$$
$$k_y(t) = k_{max} \sin(2\pi\Omega_y t/T) \quad (5.2)$$

where Ω_x is 19, and Ω_y is 20 in this case. The gradient waveforms that produce this trajectory

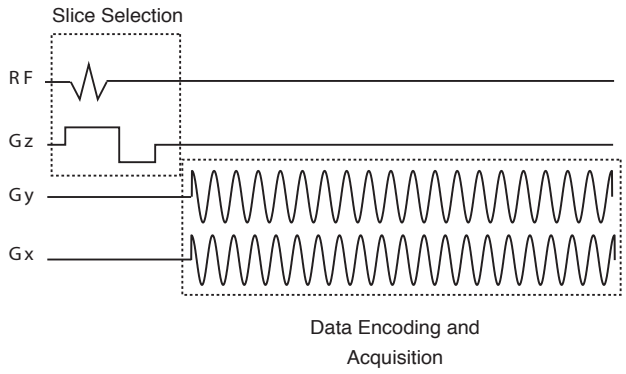

Figure 5.3: Pulse sequence that generates a Lissijou k-space trajectory.

are

$$G_x(t) = \frac{k_{max}T}{(2\pi)^2\gamma\Omega_x}\cos(2\pi\Omega_x t/T) \quad (5.3)$$

$$G_y(t) = \frac{k_{max}T}{(2\pi)^2\gamma\Omega_y}\cos(2\pi\Omega_y t/T) \quad (5.4)$$

These waveforms are plotted in the pulse sequence shown in Fig. 5.3.

The interesting thing about sensitive point is that it was one of the first imaging methods that didn't fall under one of the previously known reconstruction methods. The samples don't fall on a rectilinear grid, so a simple 2DFT is inadequate for reconstruction. The samples also don't fall on radial lines, so projection-reconstruction can't be used. This required a new perspective on reconstruction.

The new idea was to demodulate the signal for each individual output voxel. Effectively the receiver is made to track the phase of a particular voxel so that its signal coherently adds over the duration of the acquisition. Ideally, the signal from other voxels do not add coherently, and integrate to zero. By selectively tuning to one voxel after another, an image can be built up. And, since the demodulation can be performed in software after the data is acquired, a single acquisition can be used to resolve all of the voxels in an image by postprocessing.

5.3.1 Conjugate Phase Reconstruction

The method developed to reconstruct sensitive point data was called a *conjugate phase reconstruction*. From Chapter 1, the signal from an acquisition is

$$s(t) = \int_{\mathbf{X}} M_{xy}(\mathbf{x}) e^{-i2\pi\mathbf{k}(t)\cdot\mathbf{x}} d\mathbf{x}. \quad (5.5)$$

For a conjugate phase reconstruction of the signal at some point $\mathbf{x}_0$ we take this acquired signal and multiply it by the conjugate of the phase at the point throughout the acquisition, and then integrate over the duration of the acquisition,

$$m(\mathbf{x}_0) = \int_0^T s(t) e^{i2\pi\mathbf{k}(t)\cdot\mathbf{x}_0} dt. \quad (5.6)$$

If we substitute for $s(t)$ and change the order of integration,

$$m(\mathbf{x}_0) = \int_{\mathbf{X}} M_{xy}(\mathbf{x}) \left(\int_0^T e^{-i2\pi\mathbf{k}(t)\cdot(\mathbf{x}-\mathbf{x}_0)} dt \right) d\mathbf{x}. \quad (5.7)$$

This is the spatial convolution of $M_{xy}(\mathbf{x})$ with the integral in parentheses. At $\mathbf{x} = \mathbf{x}_0$ this clearly does the right thing. The inner integral becomes the integration time T, and the image value $m(\mathbf{x}_0)$ is simply $TM_{xy}(\mathbf{x}_0)$. At other values of $\mathbf{x}$ the results are less clear, and depend on the characteristics of the particular k-space trajectory. In fact, the inner integral is the impulse response of an imaging system using that particular k-space trajectory,

$$A(\mathbf{x}) = \int_0^T e^{-i2\pi\mathbf{k}(t)\cdot\mathbf{x}} dt. \quad (5.8)$$

The goal is that the impulse perform as a delta function, sifting out only the value of the image at $\mathbf{x}_0$. Interestingly, this does not work well for Lissajou trajectories, which leads to the next development in image reconstruction.

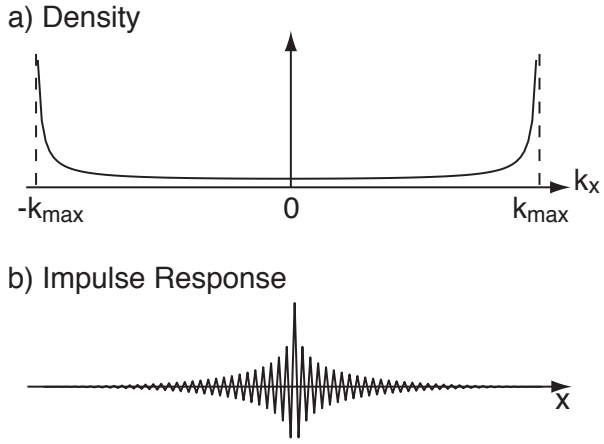

Figure 5.4: Sampling density of a Lissajou trajectory (a), and the impulse response that results from this density (b).

5.3.2 Density Correction

The problem with a Lissajou pattern can be immediately appreciated by examining Fig. 5.2. That is, the pattern is not at all uniform. The sampling density is much higher at the edges than in the middle. The result is an impulse response that "rings" significantly, and does not do a very good job of localization. This is illustrated in Fig. 5.4. This led to the realization that conjugate phase reconstruction by itself was inadequate, and that some weighting must be included to account for the sampling density.

For convenience we will just concern ourselves with the x axis for the time being. The impulse response is

$$A(x) = \int_0^T e^{-i2\pi k_x(t)x} dt \tag{5.9}$$

where

$$k_x(t) = k_{x,max} \sin(2\pi\Omega_x t/T) \tag{5.10}$$

If the time T is an integer number of cycles at Ω_x,

$$A(x) = \int_0^T e^{-i2\pi x k_{x,max} \sin(2\pi\Omega t/T)} dt \tag{5.11}$$

is the one dimensional projection of a two dimensional delta ring. The impulse response is then

$$A(x) = J_0(\frac{\pi x}{\Delta x}) \tag{5.12}$$

which is not a particularly good localization function.

The reason can be seen by rewriting the expression for the impulse response as an integral in k_x. If we look at one half cycle of the sinusoid, say from $3\pi/2$ to $5\pi/2$, and change variables so that

$$dk_x = \left|\frac{dk_x}{dt}\right| dt = |\gamma G_x(t)|\, dt, \tag{5.13}$$

$$\begin{aligned} A(x) &= \int_{3\pi/2}^{5\pi/2} e^{-i2\pi k_x(t)x} dt & (5.14)\\ &= \int_{-k_{max}}^{k_{max}} \frac{e^{-i2\pi k_x x}}{|\gamma G_x(t(k_x))|} dk_x. & (5.15) \end{aligned}$$

The problem is that $G_x(t)$ goes to zero at $k_x = \pm k_{max}$, resulting in impulses in the k-space weighting. In order to make the impulse response behave, it is necessary to add an additional weighting factor

$$W(t) = |G_x(t)| \tag{5.16}$$

so that

$$A(\mathbf{x}) = \int_0^T W(t) e^{-i2\pi \mathbf{k}(t)\cdot\mathbf{x}} dt. \tag{5.17}$$

This produces the desired impulse for an impulse response.

The conjugate phase reconstruction is then

$$m(\mathbf{x}_0) = \int_0^T s(t) W(t) e^{i2\pi \mathbf{k}(t)\cdot\mathbf{x}_0} dt. \tag{5.18}$$

This is a solution to the problem. Unfortunately it is very slow to compute. If the image size is $N \times N$ pixels, there are on the order of N^2 data samples in the acquisition. The reconstruction of each point requires an N^2 element inner product, and this must be repeated for each of the N^2 pixels, requiring on the order of N^4 operations for a reconstruction. The next step is to look for approaches that perform the same functions, but more efficiently.

5.4 k-Space Resampling Methods: Overview

A much faster approach is to resample the k-space data onto a 2D Cartesian grid first, and then use a 2D DFT to reconstruct the image. There are many options for resampling algorithms. These will be grouped into three broad areas. The first is " grid driven" interpolation. Here the value at each grid point is interpolated from the neighboring k-space data. The second is "data driven" interpolation, where the contribution from each data point is added to the adjacent grid points. This includes the *gridding* algorithm that we will be examining in detail. Finally, there are a number of approaches that try to compute local approximations to optimum interpolators for the specific locations of the sample points and grid points.

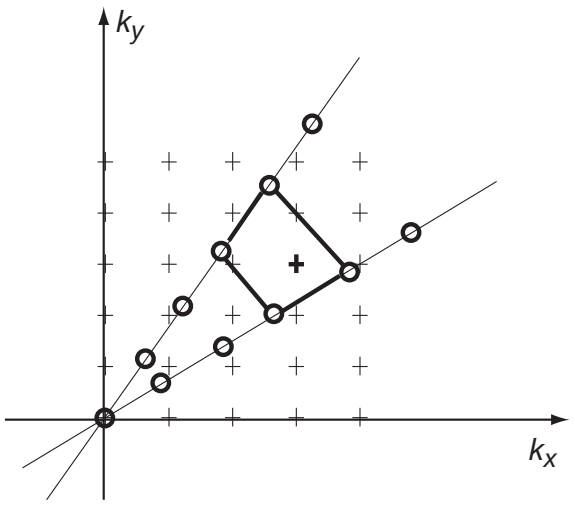

Figure 5.5: Grid-driven interpolation for a projection data set. Data samples lie on diameters in k-space. In this example the surrounding four data samples (o's) are located for each grid point (+'s), and a value at the grid point determined by bilinear interpolation.

5.4.1 Grid-Driven Interpolation

The idea of grid-driven interpolation is to estimate the value at each grid point based on the immediately surrounding data. An example is illustrated in Fig. 5.5

One advantage of this approach is that it is easy to implement if the location of the neighboring data points can be determined analytically. One example is projection-reconstruction. For other trajectories the locations of the neighboring data points can be found in a initialization stage.

It does have the disadvantage that it won't in general use all of the input data, and hence won't be as SNR efficient as some of the other approaches we will consider. This isn't as much of a drawback as it might appear, though, because the unused excess data is usually at low spatial frequencies where the SNR is already high, and the eye is relatively insensitive to errors in low spatial frequencies. Because it doesn't use all the data, this approach doesn't require a density estimate, which can be a significant advantage.

The fidelity of image reconstructions using this approach are a tradeoff between interpolator complexity and k-space oversampling. If a simple bilinear interpolator is used, image quality is poor if the k-space grid is the same as sampling density required to support the desired image FOV. This is called a "1X" grid. However, if the k-space grid is twice as finely sampled, a "2X" grid, the reconstructions are quite acceptable. Sampling artifacts are below the noise floor for typical MRI images. Alternately, the use of a higher-fidelity interpolator could be used with a 1X grid, at the cost of some complexity.

In practice, this approach is seldom if ever used. Mostly this is due to the general convergence of the MRI community on the data-driven interpolation method called gridding, which we will discuss next. However, grid-driven interpolation is a reasonable alternative that can produce high-fidelity

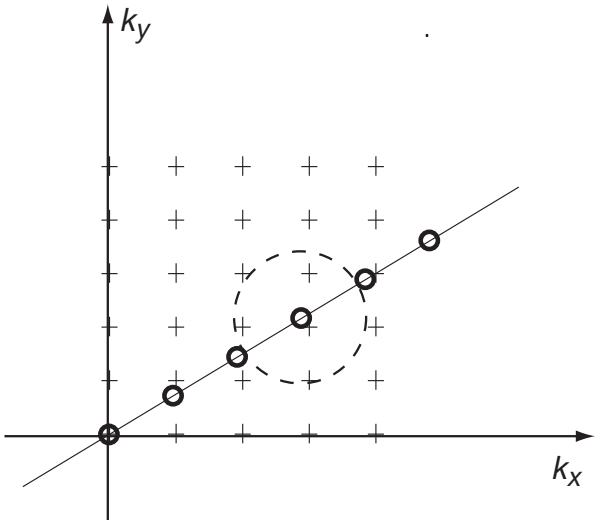

Figure 5.6: Data-driven interpolation for a projection data set. Again, data samples lie on diameters in k-space. Each data point is conceptually considered to be convolved with a small kernel, and the value of that convolution added to the adjacent k-space grid points.

reconstructions. This is particularly true when the sampling density is varying, such as undersampled PR, difficult to compute, or continually changing.

5.4.2 Data-Driven Interpolation

The idea of data-driven interpolation is to take each data point, and add its contribution to the surrounding grid points. There are a number of ways this can be done. We consider here the case where each data point is handled uniformly. The next section on optimum interpolators concerns the case where each data point is considered specifically as a special case.

Each input sample is conceptually considered to be convolved with a small kernel, which is chosen to be wide enough to extend to the neighboring grid points. In this way, each data point is "resampled" at the adjacent grid points. This approach has the advantage that all of the data is used, so it is more SNR efficient than the grid-driven approach described above. However, as a result, it needs a density estimate to correct for the fact that the samples may be concentrated in particular areas of k-space.

As for the grid-driven approach, reconstruction fidelity is a tradeoff between interpolator complexity and k-space oversampling. Using a large kernel (e.g. 4 sample radius) on a 1X grid produces a high-fidelity reconstruction. Usually a good tradeoff is a simpler kernel on a 2X grid, where the k-space oversampling allows for a faster interpolation. We will examine this in more detail below.

5.4.3 "Optimum" Interpolators

The data-driven interpolators use the same convolution kernel for each k-space sample. Better performance can be achieved if each data sample is considered independently, and has its own unique interpolation function. This is typically computed over a local neighborhood to keep the computation within reason, so it is only an approximation to the optimum interpolators. The computation of these functions for each data sample requires a significant amount of setup time. However, once these have been computed they can be used repeatedly. The main advantage of this approach is the ability to produce high fidelity reconstructions on a 1X grid. There are two examples we will examine. One is called BURS, and the other is the nuFFT algorithm that has been increasingly of interest.

5.5 Gridding Reconstruction

The reconstruction method that we will consider in the greatest detail is called gridding. After introducing the basic idea, we will examine the three main design concerns in implementing a gridding reconstruction. These are the choice of a convolu-

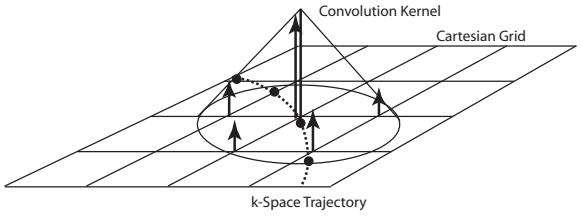

Figure 5.7: Basic gridding idea. The data samples line on some trajectory through k-space (dashed line). Each data point is conceptually convolved with a gridding kernel, and that convolution evaluated at the adjacent grid points.

tion kernel, the density of the reconstruction grid, and the estimation of the sample density. Then we will consider the problem of inverting the gridding operation, and going from Cartesian image space data to non-Cartesian k-space data.

The basic idea of gridding is illustrated in Fig. 5.7. Data points lie along some trajectory through k-space. Each data point is convolved with a gridding kernel, and the result sampled and accumulated on the Cartesian grid. After all the data samples have been processed, a 2D DFT^{-1} produces the reconstructed image.

This simple version of gridding can be described mathematically. The non-Cartesian sampling function $S(k_x, k_y)$ is

$$S(k_x, k_y) = \sum_i {}^2\delta(k_x - k_{x,i}, k_y - k_{y,i}). \quad (5.19)$$

The sampled data is then $M(k_x, k_y)S(k_x, k_y)$. This is convolved with the gridding kernel $C(k_x, k_y)$, and then sampled on the Cartesian grid,

$$\begin{aligned} \hat{M}(k_x, k_y) &= [(M(k_x, k_y)S(k_x, k_y)) * C(k_x, k_y)] \\ &\quad \times \text{III}\left((\frac{k_x}{\Delta k_x}, \frac{k_y}{\Delta k_y} \right) \end{aligned} \quad (5.20)$$

After the Fourier transform, this becomes

$$\hat{m}(x, y) = [(m(x, y) * s(x, y))\, c(x, y)]$$

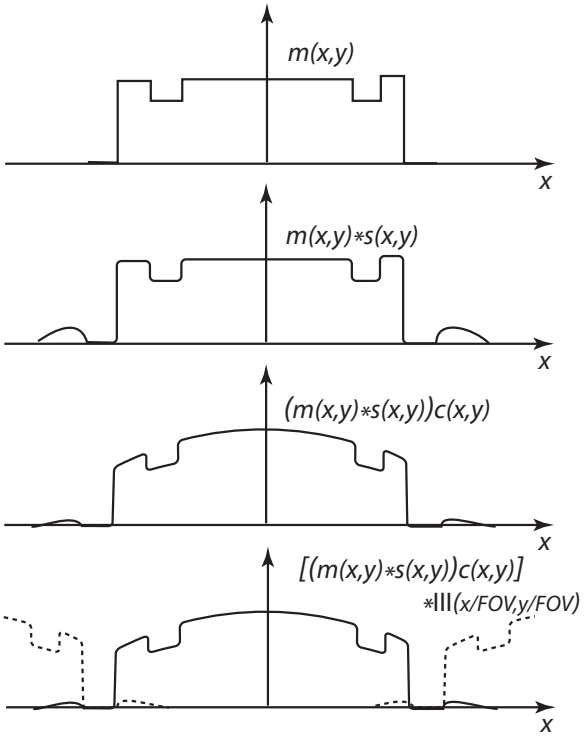

Figure 5.8: Effects of the various terms in the image domain gridding expression given in Eq. 5.21.

$$* \text{III}\left(\frac{x}{FOV_x}, \frac{y}{FOV_y} \right). \quad (5.21)$$

The effects of the various elements of this equation are illustrated in Fig. 5.8. The ideal image $m(x, y)$ is first blurred by convolution with the transform of the sampling function. In addition, sidelobes are usually created due to the pattern of the samples in k-space. For example, for spirals there is a spiral ring sidelobe. Next, the image is apodized by the transform of the gridding kernel. While this has the undesireable effect of producing shading in the image, it also has the very desireable effect of suppressing the sidelobes that were generated by the convolution with the sampling function. Finally, the rectilinear sampling in k-space causes replication in image space. Sidelobes from the first replicas interfere with the desired image.

There are many design issues with the implementation of a gridding reconstruction, that are illustrated in Fig. 5.8. One is the density of the Cartesian grid Δk_x and Δk_y. Since we are impressing this grid, we are free to choose its density to be whatever we like. This has an effect on the amount of aliasing from the adjacent replica lobes, and indirectly on the apodization that the kernel produces. Other important parameters are the shape and size of the gridding kernel $C(k_x, k_y)$, which also effects both the apodization and aliasing. Finally, the sampling pattern $S(k_x, k_y)$ determines the impulse response of the system. This must be corrected for the k-space sampling density, which must be estimated.

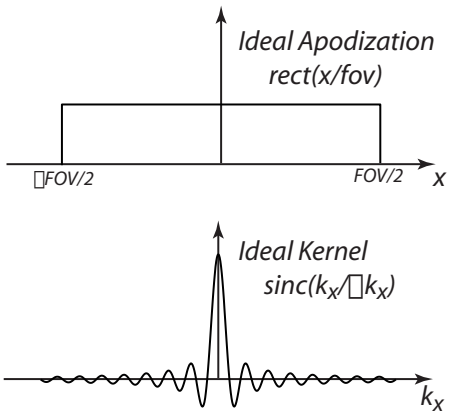

Figure 5.9: Ideal apodization function rect $\left(\frac{x}{FOV}\right)$ and the corresponding ideal kernel function sinc $\left(\frac{k_x}{\Delta k_x}\right)$.

5.5.1 Apodization and $C(k_x, k_y)$

The ideal apodization function would be rect$(\frac{x}{FOV})$rect$(\frac{y}{FOV})$. In this case the ideal gridding kernel function would be

$$C(k_x, k_y) = \text{sinc}\left(\frac{k_x}{\Delta k_x}\right)\text{sinc}\left(\frac{k_y}{\Delta k_y}\right). \qquad (5.22)$$

The x component of the ideal apodization and kernel are illustrated in Fig. 5.9. Unfortunately the ideal kernel is infinite in extent. We need to truncate the kernel at some point. The tradeoffs in deciding this point are how much of the FOV is lost to apodization, and how much aliasing folds back into the image from the first replica side lobes.

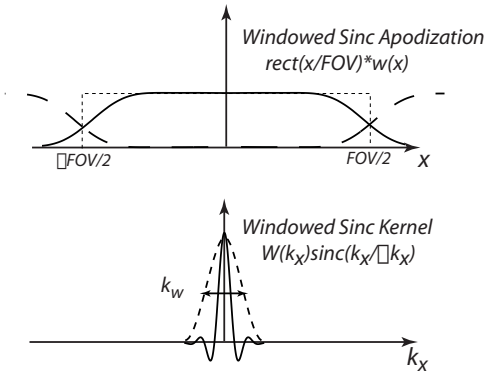

Figure 5.10: Apodization function and kernel for a windowed sinc kernel.

Windowed Sinc Kernels The first approach is to take the infinite sinc and truncate it with a smooth window function, such as a Hamming window. Examples of apodization function, and the windowed sinc kernel are shown in Fig. 5.10. For most of the image the apodization is constant, and doesn't need to be corrected. At the edges of the image there are transition bands where there is increasing apodization, as well as aliasing from the first replicas. The exact width of this region x_{tw} depends on the particular window, but is approximately

$$x_{tw} \approx \frac{1}{k_w} = \frac{1}{n\Delta k_x} = \frac{1}{n}FOV \qquad (5.23)$$

where k_w is the half-amplitude width of the window. The overall window width will be about twice this. Hence, a k_w of 4 as illustrated in Fig. 5.10 results in a loss of 25% of the FOV due to apodization and aliasing. Larger windows require more computation, however, so we must trade off computation for usable FOV. The computation per data point is $\sim (2n)^2$ operations, and there are

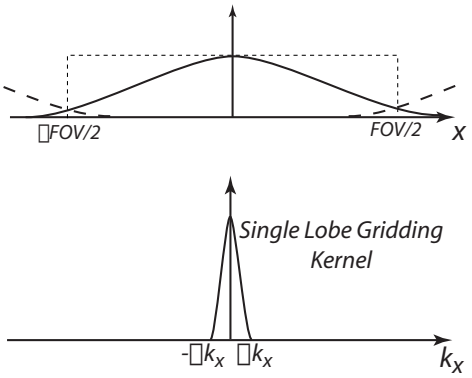

Figure 5.11: Apodization function and kernel for a single lobed gridding kernel.

$\sim N^2$ data points for an $N \times N$ image. As a result, we want to keep n small as we can.

Small Kernels (Gridding) When gridding was first applied to CT and MRI in the early to mid 1980's, computing power was limited, so it was important to keep gridding kernels as small as possible. The emphasis at that time was on single lobe kernels, and that has continued. In general the width of the single lobe kernel is wider than that of the main lobe of the windowed sinc. This makes the apodization function narrower in space, which reduces aliasing at the cost of FOV. The apodization in this case is significant, and must be corrected almost everywhere in the FOV.

Deapodization The apodization function is the transform of the gridding kernel, which can either be calculated analytically for most popular windows, or can be computed numerically. Once it has been computed, the apodization can be corrected completely by dividing by the image by the ideal apodization function. In practice, is often a better idea to divide out only part of the apodization. There are several reasons for this. There are typically significant artifacts at the edge of the FOV, and these get accentuated by the deapodization.

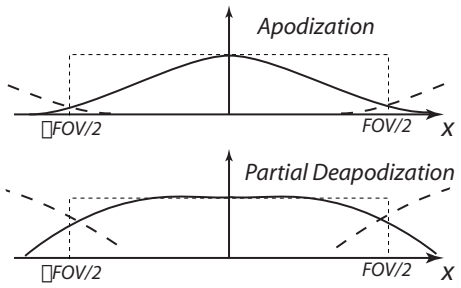

Figure 5.12: Partial deapodization limits artifacts from aliasing at the edges of the FOV.

Also, if a feature is of interest, it is most likely in the middle of the FOV anyway. This is particularly true with real-time interactive imaging. Finally, the eye is not particularly sensitive to low frequency variations, so some residual apodization is not objectionable. An example illustrating partial deapodization is shown in Fig. 5.12. One way to limit the deapodization is to divide by $c(x,y)+a$ instead of $c(x,y)$,

$$\hat{m}(x,y) = \frac{1}{c(x,y)+a}\{[(m(x,y) * s(x,y))\, c(x,y)] * \mathrm{III}(\frac{x}{FOV_x}, \frac{y}{FOV_y})\}. \quad (5.24)$$

Note that the deapodization occurs after the convolution of the III$(\cdot)$ function.

5.5.2 Oversampling

For the examples we have considered so far, we have a serious problem in that the adjacent replica sidelobes have the same amplitude as the desired image at the edge of the FOV. The problem is that we have no space for a transition band between the image and the replica sidelobes. This is a consequence of reconstructing the image using a grid that is the same as the underlying sampling of the k-space data, which is called a "1X" grid. Fortunately, there is nothing fundamental about the grid density that we use to perform the reconstruction. We impose the grid, and we can choose

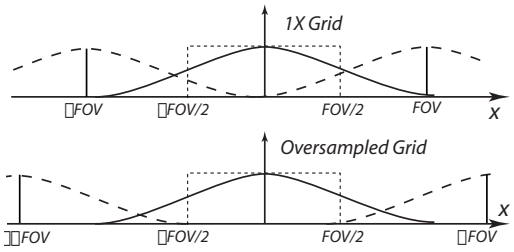

Figure 5.13: Reconstruction on a denser grid (oversampling) moves the replica sidelobes out, reducing aliasing, and allowing less apodization.

the grid density to be anything we would like. By choosing the grid to be denser than the sampling of the underlying k-space data, we can allow for a transition band, and reduce both the apodization and aliasing significantly. If $\alpha > 1$, we reconstruct on a grid

$$\left(\frac{\Delta k_x}{\alpha}, \frac{\Delta k_y}{\alpha}\right) \tag{5.25}$$

The reconstructed image is then

$$\begin{aligned} \hat{m}(x,y) &= [(m(x,y) * s(x,y))\, c(x,y)] \\ &\quad * \text{III}(\frac{x}{\alpha FOV_x}, \frac{y}{\alpha FOV_y}). \end{aligned} \tag{5.26}$$

where we have neglected the deapodization term for simplicity.

The effect of oversampling is illustrated in Fig. 5.13. Increasing the spacing between replica sidelobes reduces aliasing dramatically. Because of this, narrower gridding kernels can be used, which produce less apodization.

2X Grid In practice, a 2X grid is commonly used. This is very forgiving. Almost any reasonable window works well. Optimized parameters for a number of variations are given in [2]. The aliased signal from the adjacent replica sidelobes is typically beneath the noise floor for most MRI images. In addition, the apodization is relatively minor, and can often be ignored. The only disadvantage is the increased computation time for the DFT, and the fact that it becomes expensive in 3D in terms of memory requirements.

2D FFT Size	Time, ms
128	1.96
160	3.23
192	5.37
224	8.28
256	15.63
288	15.50

Table 5.1: 2D FFT times using the FFTW package. Note that smaller non-power-of-two transforms are much faster than the next larger power of two, 256×256. In fact, even the 288×288 FFT is faster than the 256×256 FFT.

Other Oversampling Factors One of the reasons for the 2X grid is the desire to use the next larger power-of-two FFT. However, recently very fast implementations of the FFT have become available for a whole range of FFT lengths. This is due to the FFTW package available from `fftw.org`. Previously, it was usually faster to use the next larger power-of-two FFT, rather than a small non-power-of-two FFT. That has all changed. Table 5.1 show the times for $N \times N$ FFTs on a 700 MHz AMD Athlon. Note that smaller non-power-of-two transforms are much faster than the next larger power of two, 256×256. In fact, even the 288×288 FFT is faster than the 256×256 FFT.

The availablity of fast FFT implementations allows great flexibility in choosing the oversampling factor. This must be chosen as a tradeoff with the size and shape of the gridding kernel, the amount of aliasing that can be tolerated, and reconstruction speed. The nature of these tradeoffs is a current research topic.

Oversampling Verses Zero Filling In general when a non-Cartesian acquisition is designed, the

acquisition parameters are the overriding concern. These include duration of the A/D window, how many acquisitions are required, and the performance of the gradient system. As a result, the resolution and FOV of the acquisition seldom works out to convenient numbers. For example, the acquisition might support a 102×102 image at 2.24 ms resolution, for an FOV of 22.85 cm. This is usually reconstructed using a 128×128 FFT (1X grid) or 256×256 FFT (2X grid). However, there are several alternatives as to how this is done.

The two extremes are illustrated in Fig. 5.14. The first, which is more common, is to grid the data to the central part of the larger reconstruction matrix. The data is effectively zero padded, and the image interpolated. The reconstructed image has the same FOV as supported by the original data. The disadvantage of this approach is that the aliasing is that of the 1X grid reconstruction, unless we oversample by a factor of 2.

The other alternative is to grid the data so that it fills the larger reconstruction matrix. The data is oversampled by the ratio of the acquisition matrix size to the reconstruction matrix size. The oversampling increases the FOV, so that the desired FOV is central portion of the reconstructed image. This is seldom done, because the unusual image sizes are not handled well by the scanner data base and display programs.

The advent of fast non-power-of-two FFT's allows these two concerns to be addressed together. We can design both oversampling and zero padding into the reconstruction, so as to produce an image that is a conventional power of two in size. If we start with the $N \times N$ acquisition, want an oversampling factor of α, and a zero filling factor z of M/N (*i.e.* we want an $M \times M$ image), we need to reconstruct on a grid that is

$$N\alpha z = N\alpha(M/N) = M\alpha \qquad (5.27)$$

in size. We do this by gridding the data into the central $N\alpha \times N\alpha$ part of the reconstruction matrix,

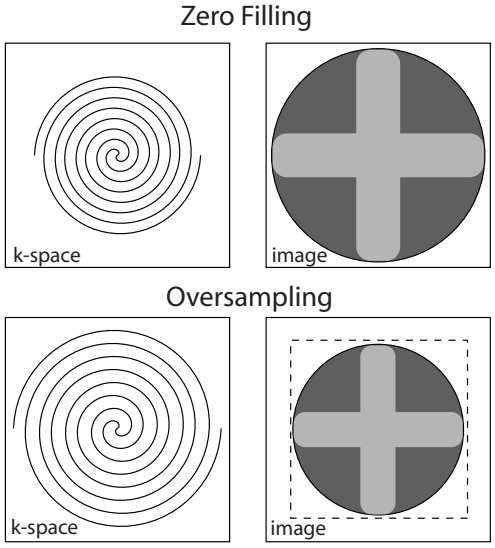

Figure 5.14: Reconstruction on a denser grid (oversampling) moves the replica sidelobes out, reducing aliasing, and allowing less apodization.

and filling the rest of the matrix with zeros. After the 2D FFT, the central $M \times M$ is extracted from the reconstruction matrix.

Returning to the 102×102 example, if we want a 1.5 oversampling factor, and a zero filling factor of 128/102, we construct on a matrix that is $(128)(1.5) = 192$. The data extends over the central 153×153 matrix, and the rest is zero filled. After the reconstruction the central 128×128 matrix is extracted. This has the acquisition FOV, has reduced aliasing due to the 1.5X grid, and has been interpolated up to a convenient size for storage and display.

5.5.3 Impulse Response

The last major issue with gridding is the effect of the sampling pattern $S(k_x, k_y)$. The transform of the sampling pattern $S(k_x, k_y)$ is the impulse response of the system $s(x, y)$. Ideally this should be an impulse, but is not in practice. The reason

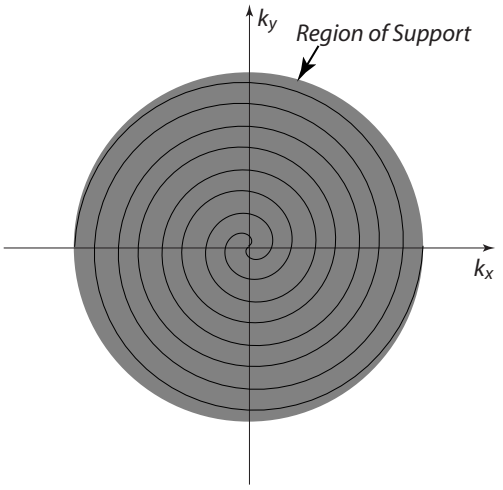

Figure 5.15: Region of support for a spiral trajectory. Ideally, it we grid unity along the trajectory we should have uniform amplitude over the region of support. This wil result in a jinc$(\cdot)$ impulse response, which is the best we could hope for with this acquisition.

can be identified by considering the case where the data is uniformly unity in amplitude. If we grid unity, we should get a uniform amplitude in k-space everywhere that data has been acquired. In this case, the impulse response is just the transform of the region of support, and this is the best we could hope for given that particular acquisition. The region of support for a spiral acquisition is illustrated in Fig. 5.15. Since this a circular disc, the impulse response would be a jinc$(\cdot)$ function.

The problem is that if we grid unity, we don't end up with a k-space weighting that is uniform. This can be due to several factors. One is the rate at which the k-space trajectory is traversed can be non-uniform. This is true at the beginning of a spiral acquisition. Another is that the sample pattern itself can bunch up at particular locations. The most important example is the origin in projection data sets. Another is the edges of k-space in Lissajou acquisitions.

The solution is to correct for the sample density in the gridding operation. There are two options for how this can be done. Ideally the density compensation should be done for each data point before the gridding operation. This is *precompensation*, which may be written

$$\begin{aligned}\hat{M}(k_x,k_y) &= \left[\left(M(k_x,k_y)\frac{S(k_x,k_y)}{\rho(k_x,k_y)}\right) * C(k_x,k_y)\right] \\ &\quad \times \text{III}\left(\frac{k_x}{\Delta k_x},\frac{k_y}{\Delta k_y}\right). \end{aligned} \tag{5.28}$$

The other alternative is to do the density compensation after the gridding operation. This is *postcompensation*, and occurs on an grid point by grid point basis.

$$\begin{aligned}\hat{M}(k_x,k_y) &= \frac{1}{\rho(k_x,k_y)} \\ &\quad \times \left[(M(k_x,k_y)S(k_x,k_y)) * C(k_x,k_y)\right] \\ &\quad \times \text{III}\left(\frac{k_x}{\Delta k_x},\frac{k_y}{\Delta k_y}\right). \end{aligned} \tag{5.29}$$

Postcompensation works well if the rate of change of the density pattern is not too rapid. The gridding kernel convolution blurs the effect of rapid density changes. If these changes happen in distances that are comparable to or smaller than the kernel, they can't be resolved in the postcompensation density correction. One place where this is a problem is at the origin of projection data sets.

In practice, both precompenstation and postcompensation can be employed. For example, precompensation can be performed based on the assumption of an ideal acquisition. The postcompensation can be estimated at reconstruction time based on imperfections in the acquisition, such as off-resonance effects, eddy currents, and gradient system errors. As long as the major variations in density have been precompensated, the postcompensation should work well.

Estimating $\rho(k_x,k_y)$ For precompensation the density $\rho(k_x,k_y)$ must be precomputed, since it is required prior to the gridding operation. For

postcompensation, the density can be estimated as part of the gridding process. There are several different methods that can be used to estimate the density. For simple cases, like spirals, projection, and Lissajou, the density can be computed using simple geometry. For more complex cases, there are several approaches. One uses gridding itself, either in a single operation, or iteratively. Another is a numerical approach based on assigning an area in k-space for each sample.

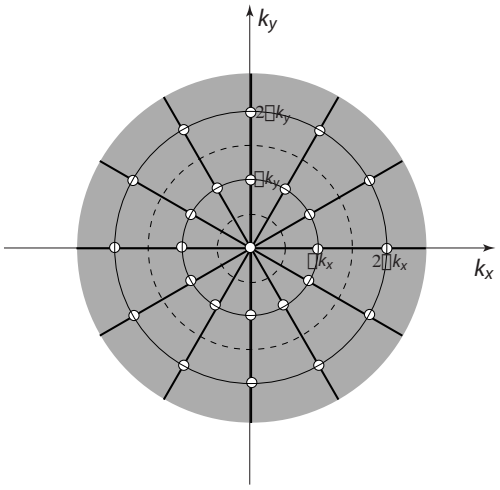

Figure 5.16: Calculation of the density for a projection data set.

Geometry For simple k-space trajectories, the area associated with each sample can be computed geometrically. One example is for projection data sets, as is illustrated in Fig. 5.16. Samples are located along radii at multiples of $\Delta k_x = \Delta k_y = \Delta k_r$.The weighting we will apply during the gridding operation is the inverse of the sample density $w(k_x, k_y) = 1/\rho(k_x, k_y)$. If there are N projections, then the central sample is acquired N times. For these the weighting $\frac{1}{N}$ multiplied by the area of the central disk that is closest to the origin.

$$w_0 = \frac{1}{N}\pi\left(\frac{\Delta k_r}{2}\right)^2. \quad (5.30)$$

$$= \frac{2\pi}{N}(\Delta k_r)^2\frac{1}{8}. \quad (5.31)$$

The weighting for the first samples is given by the area in the next annular ring outside the central disk divided by the number of samples,

$$w_1 = \frac{1}{N}\pi\left[\left(\frac{3\Delta k_r}{2}\right)^2 - \left(\frac{\Delta k_r}{2}\right)^2\right] \quad (5.32)$$

$$= \frac{2\pi}{N}(\Delta k_r)^2. \quad (5.33)$$

For n greater than 1,

$$w_n = \frac{2\pi}{N}(\Delta k_r)^2 n. \quad (5.34)$$

This is the well known *rho* filter from projection-reconstruction. Note that the weighting is linear as expected, but does not go to zero at the origin. This is important in order to get the DC value right for projection reconstruction. Similar geometrical arguments can be used to derive the weighting function for other trajectories, such as spirals [5,3].

Gridding Density If the density cannot be easily computed based on geometry, or it is changing, another alternative is to compute the density using gridding itself. Essentially, this consists of gridding a unity data vector. This can easily be performed in parallel with gridding the actual data by gridding ones into a density matrix along with the gridding of the k-space data matrix. The estimate of the density is then

$$\hat{\rho}(k_x, k_y) = [S(k_x, k_y) * C(k_x, k_y)]\,\text{III}\left(\frac{k_x}{\Delta k_x}, \frac{k_y}{\Delta k_y}\right). \quad (5.35)$$

Note that this is on the *grid* points, so is directly suitable for postcompenstation only. If the density gridding is being performed in parallel with the data gridding, the compenstation is performed by dividing the two matrices, with some provision to avoid dividing by zero outside the area where data has been collected. This can be done by defining a mask where the density is above some thresh-

old and only compensating over that region, or by adding a small constant to the density matrix.

If the sampling function $S(k_x, k_y)$ is slowly varying, the estimate $\hat{\rho}(k_x, k_y)$ will be good, and post-compensation will work well. However, if $S(k_x, k_y)$ varies rapidly, the gridded density estimate will be inaccurate because it has been blurred by the gridding kernel. One important place where this is a problem is at the origin of projection data sets. In the Jackson paper [2] the density is assumed to be computed by gridding, and is written as $S(k_x, k_y) * C(k_x, k_y)$.

Inverse Gridding Ideally, we would like to know the density on the data points, so that we can precompensate for the density. The problem is how to go from a function that is defined on the grid points, back to a function defined on the data sample points. This is known as *inverse gridding* [6]. This is useful for many applications beyond density estimation. Some uses are the simulation of MRI data, iterative partial k-space algorithms for non-Cartesian data, time-series reconstructions, and non-Cartesian SENSE (undersampled multicoil) acquisitions.

The basic idea in inverse gridding is to reverse the steps in gridding. We assume we start with image data, and we want to estimate the corresponding data samples on a non-Cartesian sampling pattern. The first step is pre-emphasis,

$$m_p(x, y) = \frac{m(x, y)}{c(x, y)}. \tag{5.36}$$

This compensates for the apodization from the convolution with a gridding kernel in k-space that will follow. In k-space, the pre-emphasis can be written

$$M_p(k_x, k_y) = M(k_x, k_y) * C^{-1}(k_x, k_y) \tag{5.37}$$

where $C^{-1}(k_x, k_y) = \mathcal{F}\left\{\frac{1}{c(x,y)}\right\}$. This is a deconvolution in k-space by $c(x, y)$. At this point $M_p(k_x, k_y)$ is on a Cartesian grid. To estimate the values at the non-Cartesian sampling points, the Cartesian data is convolved with the gridding kernel, and sampled at the desired sample locations,

$$\hat{M}(k_x, k_y) = [M_p(k_x, k_y) * C(k_x, k_y)]S(k_x, k_y). \tag{5.38}$$

This is exactly the same operation as in gridding, except that at each data point we *accumulate* the contributions from all of the neighboring grid points to estimate the value at that sample. This differs from gridding where the value at that sample is *distributed* over the neighboring grid points. One important point is that there is no need of density compensation for inverse gridding, since the density of the Cartesian grid is uniform. This is an important distinction.

For density estimation we can take our gridded density estimate on the grid points, and compute what the density estimate is on the data samples. The first step is to computed a pre-emphasized estimate of the density. This is $\hat{\rho}(k_x, k_y)$ that has been compensated once for the apodization of the gridding operation, and a second time for the apodization in the inverse gridding operation,

$$\hat{\rho}_p(k_x, k_y) = \mathcal{F}\left\{\frac{\mathcal{F}^{-1}\{\hat{\rho}(k_x, k_y)\}}{c^2(x, y)}\right\} \tag{5.39}$$

The pre-emphasized gridded density is then convolved with the gridding kernel and resampled to give the density on the data samples

$$\hat{\rho}_d(k_{x,i}, k_{y,i}) = [\hat{\rho}_p(k_x, k_y) * C(k_x, k_y)]S(k_x, k_y). \tag{5.40}$$

In practice this will still be an imperfect estimate, and iteration may be required. There are several different approaches to iteration that have been proposed to improve numerical stability [7].

Numerical Computation Another alternative is to numerically compute the area associated with each data sample, and use that as the density estimate. The area is approximated by the spacing

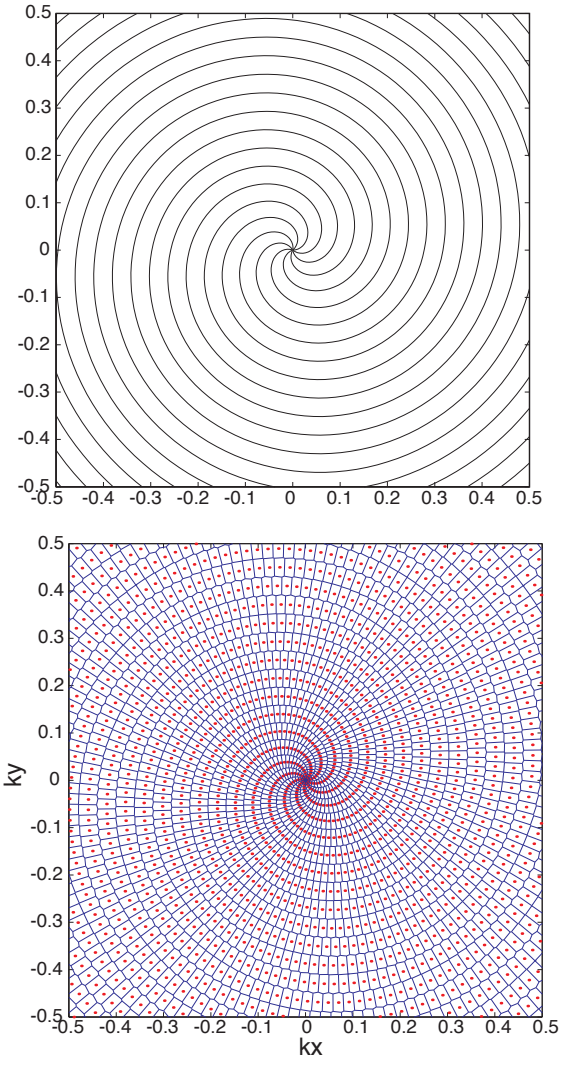

Figure 5.17: Voronoi diagram divides the plane into regions that are closest to any individual data point. On the top is the central portion of a nin-interleave spiral k-space trajectory. On the bottom the Voronoi plot shows k-space region associated with each sample. The density is the inverse of the area of these regions.

of samples along the trajectory multiplied by the spacing between adjacent trajectories. This is easy to compute for simple trajectories, but is in general a hard problem.

The paper by Rasche *et al.* [6] describes a numerical algorithm for this problem called the Voronoi diagram. This starts by a triangular tiling of the plane by associating each sample with it's closest neighbors, which is called a Delaunay triangulation. Bisecting the connections between nearest neighbors divides the plane into regions that are closest to each data point, which is the Voronoi diagram. The remarkable thing about the Voronoi diagram is that there are very efficient algorithms for its computation, which make it very attractive for density estimation.

In matlab6, the `voronoin` routine returns a list of cells, with each cell defined by its vertices in the Voronoi diagram. A density estimate would be the area of each of the cells. A matlab routine that returns the area associated with each sample is

```
function area = voronoidens(kx,ky);

% function area = voronoidens(kx,ky);
% input: kx, ky are k-space trajectories
% output: area of cells for each point

[row,column] = size(kx);
kxy = [kx(:),ky(:)];

% returns vertices and cells of
%     voronoi diagram
[V,C] = voronoin(kxy);

% unpack cell array, compute
%    area of each ploygon
area = [];
for j = 1:length(kxy)
  x = V(C{j},1);
  y = V(C{j},2);
  lxy = length(x);
  A = abs(sum(0.5*(x([2:lxy 1])-x(:)).* ...
        (y([2:lxy 1])+y(:))));
  area = [area A];
end

area = reshape(area,row,column);
```

This routine is that it takes much longer to unpack the cell array and compute the areas of the polygons, than it does to compute the Voronoi diagram in the first place! In general, though, it is pretty fast, typically requiring a few 10's of seconds.

Pre/Post Compenstation Although ideally we would always like to perform only precompensation, in practice this can be difficult. The calculation of the density can require significant computation. Changes in acquistion parameters can cause changes the k-space trajectory, and make it desirable to recompute the density. Some factors that can effect the density include off-resonance, eddy currents, gradient delays, and gradient system fidelity. One altenative is to use an estimated ideal density correction for the precompenstation, but also grid the density along with the data gridding, and then perform postcompensation also. This has some significant benefits. The precompensation should take care of the areas with rapidly changing densities, where postcompensation can fail. As long as the precompensation is reasonably close, the postcompenstation should perform well.

References

1. J. OSulllivan, "A fast sinc function gridding algorithm for Fourier inversion in computed tomography," *IEEE Trans. Med. Imag.*, vol. MI-4, no. 4, pp. 200207, 1985.

2. J. Jackson, C.H. Meyer, D.G. Nishimura, and A. Macovski. "Selection of a Convolution Function for Fourier Inversion Using Gridding," *EEE Trans. Med. Imag.*, vol. 10, no. 1, pp. 473–478, 1991.

3. C. H. Meyer, B. S. Hu, D. G. Nishimura, and A. Macovski, "Fast spiral coronary artery imaging," *Magn. Reson. Med.*, vol. 28, pp. 202-213, 1992.

4. H. Schomberg and J. Timmer, "The gridding method for image reconstruction by Fourier transformation," *IEEE Trans. Med. Imag.*, vol. 14, no. 3, pp. 596–607, 1995.

5. R. D. Hoge, R. K. S. Kwan, and G. B. Pike, "Density compensation functions for spiral MRI," *Magn. Reson. Med.*, vol. 38, pp. 117-128, 1997.

6. V. Rasche, R. Proska, R. Sinkus, P. Boernert, and H. Eggers. "Resampling of Data Between Arbitrary Grids Using Convolution Interpolation," *IEEE Trans. Med. Imag.*, vol. 18, no. 5, pp 385–392, 1999.

7. J.G. Pipe and P. Menon. "Sampling density compensation in MRI: rationale and an iterative numerical solution." *Magn Reson Med.* vol. 41, no. 1 pp. 179–186, 1999.