Note on radon and iradon transforms and Matlab's iradon on the all-at-once call vs. the one-at-time call

Nasser M. Abbasi

July 17, 2008 page compiled on June 30, 2015 at 11:43pm

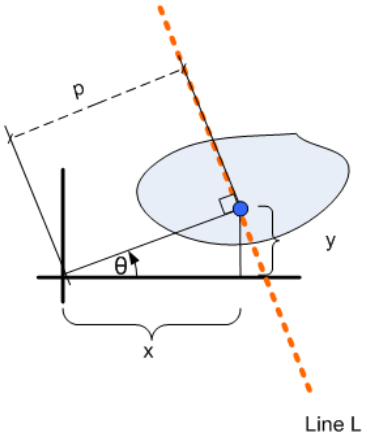
This note was motivated by a strange result found when using Matlab iradon.

In matlab, when using iradon to obtain a backprojection image from a set of projections, a different result is obtained depending if one calls iradon() passing it the set of projections all at once vs. if one calls iradon() once for each projection, and then sum the resulting set of backprojections.

Calling the first method above `all-at-once` method and the method as the `one-at-time` method, then it is found that using the one-at-time method resulted in an image whose intensity levels is $2N$ times that of the image resulting from using the all-at-once method. Here N is the number of projections.

This report is the result of investigation made to determine the reason for this difference.

1 Radon transform introduction



The equation of the line L can be written in 2 ways. The standard way is

$$y = mx + b$$

Where m is the slope and b is the intercept. It can also be written in terms of the parameters p and θ as

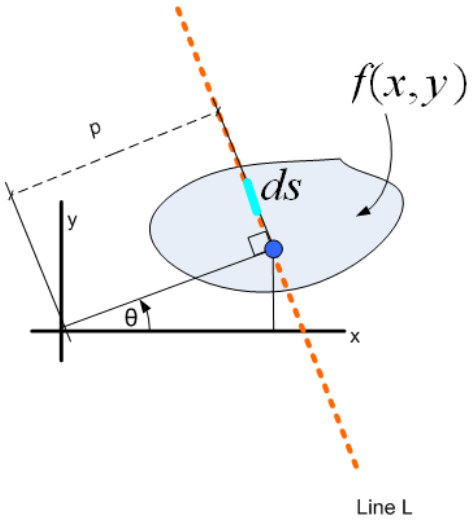
$$L(p,\theta) = \{(x,y) : x\cos\theta + y\sin\theta = p\}$$

Any point (x,y) on the line L with specific p and specific θ satisfy $x\cos\theta + y\sin\theta = p$

Assuming there exist a function $f(x,y)$ defined over the region shown above. The integral of this function over the line $L(p,\theta)$ is

$$\int_L f(x,y)\,ds$$

where ds is a differential element of the line

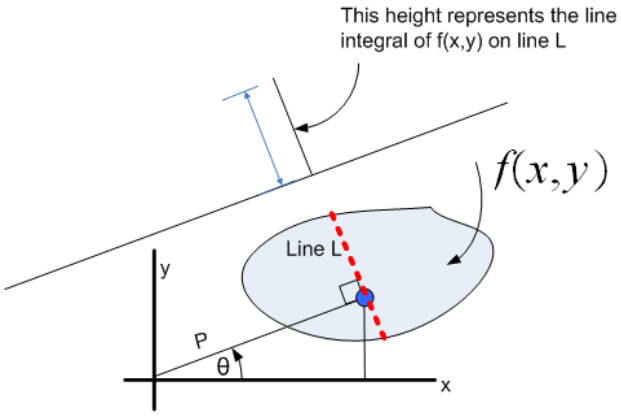


It is simpler to express the above integral in terms of x and y. To do that, a trick is used with the help of the delta function. The above integral can be written as

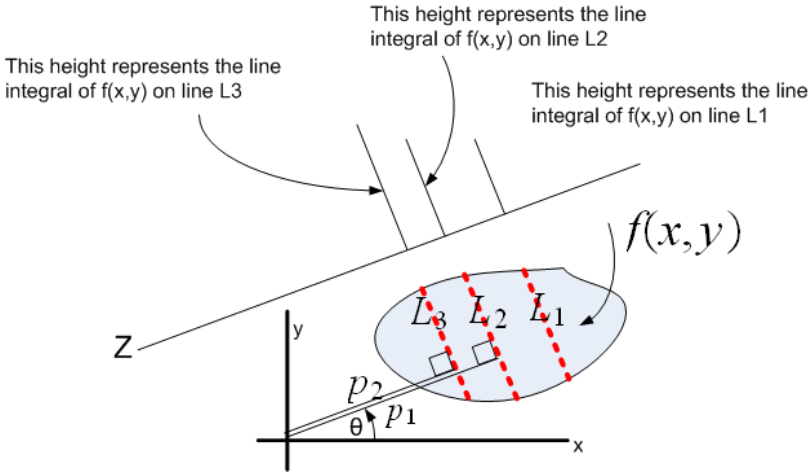
$$\int_{y=-\infty}^{\infty}\int_{x=-\infty}^{\infty} f(x,y)\ \ \delta(x\cos\theta + y\sin\theta - p)\ \ dxdy$$

Hence for a specific p, θ the above will integrate $f(x,y)$ over the line $L(p,\theta)$. The above is the radon transform of $f(x,y)$ over the line $L(p,\theta)$. So the radon transform is really the line integral of a $f(x,y)$

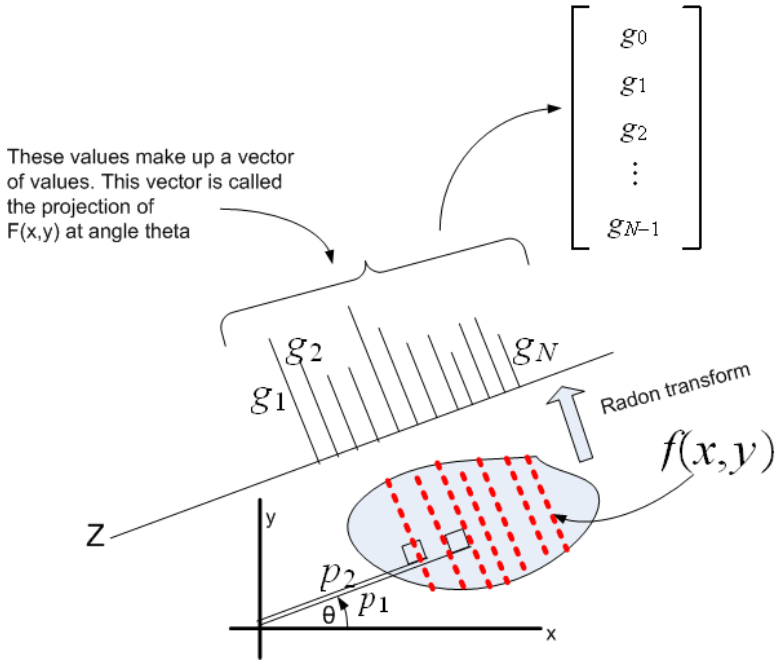
The result of the above radon transform is one numerical value. It is the line integral value. We can imagine a projection line into which we accumulate the result of these line integrals as follows



Now suppose we have many parallel lines to L and we perform the line integral of $f(x,y)$ over each of these lines (since all these lines are parallel to line L, then all of them will have the same θ, but they will have different p each). This will result in many line of the projection line as follows



So, if we do the above over many parallel lines, we obtain many sample points on the projection line z, Notice that the projection line z in the above diagram is some arbitrary line drawn just to collect the result of the line integrals into. It represents a detector which collects the results of each line integral. If we collect many line integrals to cover the whole region. Hence for each specific angle θ_i we obtain a projection vector $\mathbf{g}_i$ as shown in the following diagram



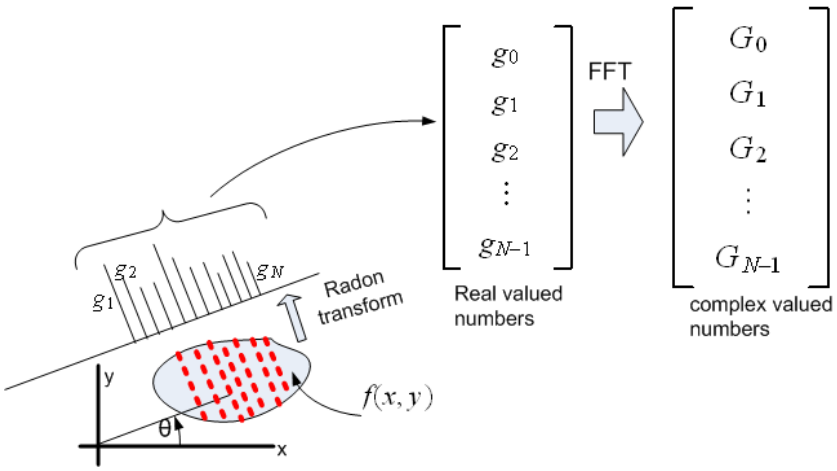
The projection shown above is a discrete function. It is a function of p and θ. Hence we have $g(p,\theta)$. But for the same angle θ, g is a function of p, So some books write $g_\theta(p)$.

Notice that z is parallel to p, and can be called the axis of the projection.

Once the projection $g_\theta(p)$ is obtained, then it is converted to the (discrete) Fourier domain using FFT. The discrete Fourier transform is

$$G_k = \sum_{n=0}^{N-1} g_n \exp\left(-\frac{2\pi i}{N} kn\right) \quad k = 0, 1, \cdots, N-1$$

Hence we obtain the vector $\mathbf{G}$ which is the discrete Fourier transform of the projection $\mathbf{g}$, This is illustrated in the following diagram. Notice that the numbers G_k are complex numbers and hence have phase and magnitude. In the implementation of the discrete Fourier transform, the FFT is used for performance.



So, why do we do the above? The reason is to filter the projection data. Filtering the projection produces better backprojection (sharper) than without filtering (more blurred). It is easier to apply filtering in the frequency domain than in the spatial domain (multiplication vs convolution). Now that the FFT is done and **G** obtained, a filter is selected. Consider the ram-Lak filter. This filter has a frequency response as follows[1]

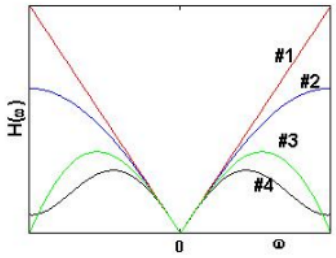


Fig. 2. Magnitude response of backprojection filters.
1=Ram-Lak (ramp), 2=Shepp-Logan, 3=Cosine, and 4=Hamming.

So, assume the filter frequency spectrum is given by the vector **H**, (this is a complex vector, since it is the frequency spectrum of the filter). Hence the filtered backprojection is given by

$$\tilde{\mathbf{G}} = \mathbf{GH}$$

Where I use the tilde symbol to represent a filtered frequency response.

Now that we filtered the projection, we need to return back to the spatial domain. Hence we obtain $\tilde{\mathbf{g}}$ by inverse discrete fourier transform of $\tilde{\mathbf{G}}$ and this is given by

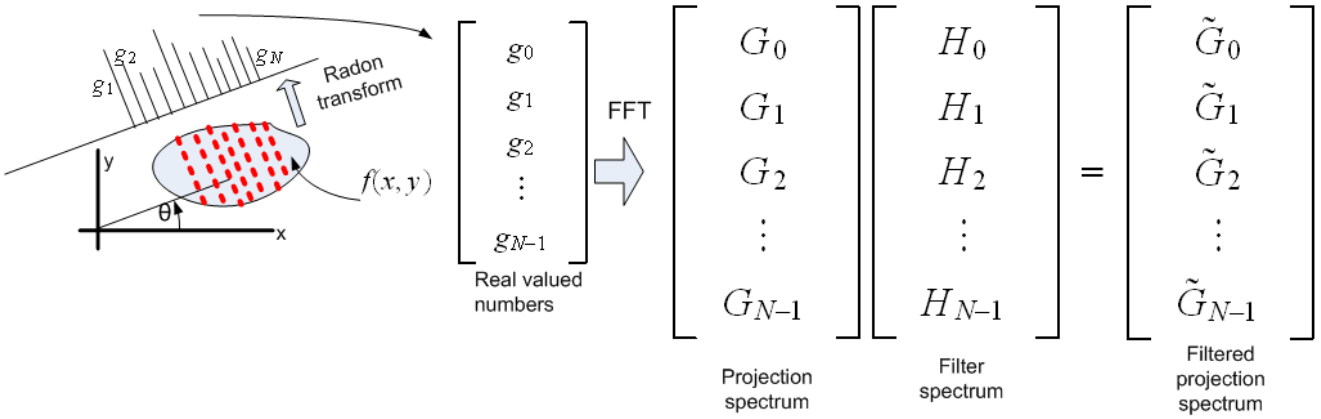
$$\tilde{g}_n = \frac{1}{N}\sum_{k=0}^{N-1} \tilde{G}_k \exp\left(\frac{2\pi i}{N} kn\right) \qquad n = 0, 1, \cdots, N-1$$

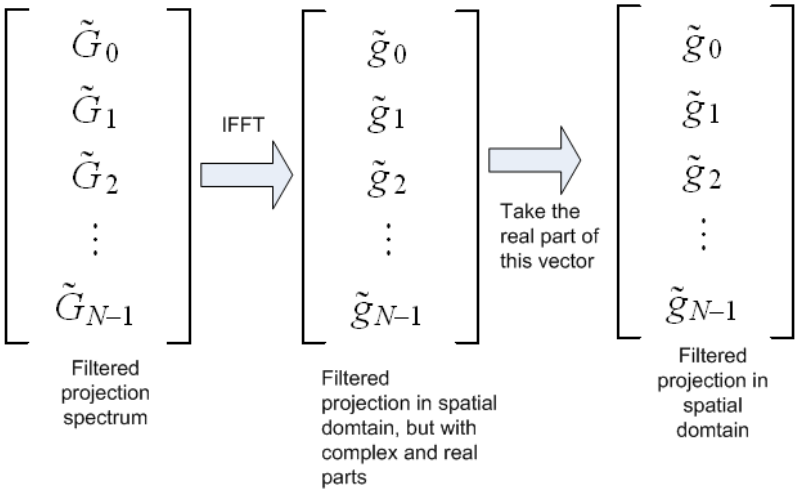
Now we have obtain the spatial representation $\tilde{\mathbf{g}}$ of the projection **g** after being filtered. However, this contains both complex and real components (since it is complex valued as result of the IFFT). Then we need to take its real part

$$\tilde{\mathbf{g}} = \text{real}\,(\tilde{\mathbf{g}})$$

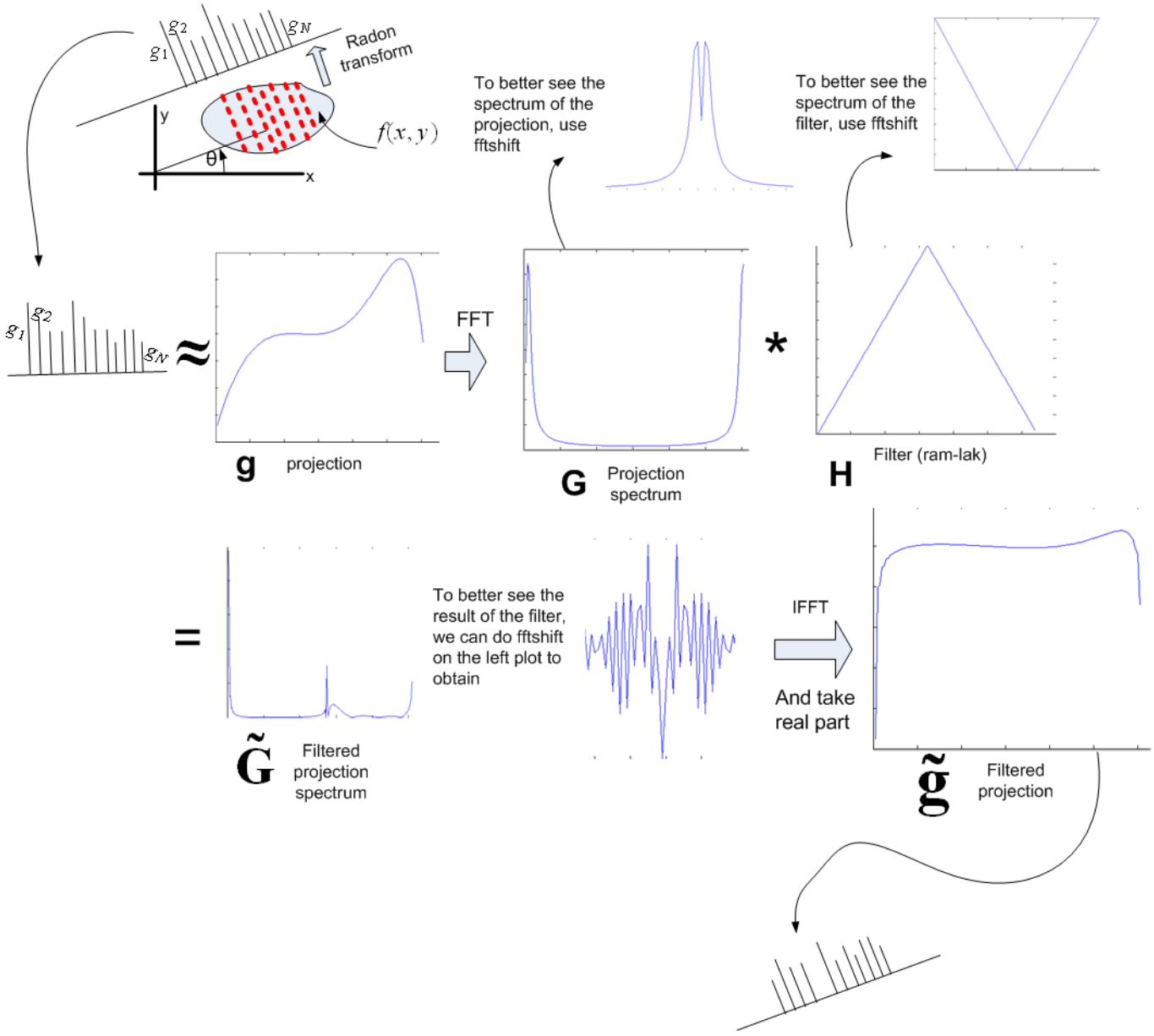
The above is illustrated by the following diagram

[1] reference: http://dukemil.egr.duke.edu/XRAY/CT/Simulation/ct_in.html





The above can also be represented using plots of the spatial and frequency spectrum of the above vectors. For these plots, I used a projection signal made up from some simple function of x. The plots shown are the actual FFT and IFFT and the filter ram-lak used to obtain the filtered backprojection



Now that the filtered projection is obtained, then backprojection is done. Notice that the filtering, when we talk about filtered backprojection, is carried on the projection itself, and not on the backprojection. I am not sure if it is possible to do backprojection on the projection first, then apply filtering on the resulting backprojection image.

The filtered projection is $\tilde{g}_\theta(p)$, where the tilde indicates this is a filtered projection. Now that we have calculated $\tilde{g}_\theta(p)$, we can obtain the backprojection, which will be a 2D function. Assume the original function (which we do not know in practice, was $f(x,y)$, then let the backprojection function be $f_B(x,y)$. Hence

$$f_B(x,y) = \int_{\theta=0}^{\pi} \tilde{g}_\theta(p)\, d\theta$$

But $p = x\cos\theta + y\sin\theta$, hence the above becomes

$$f_B(x,y) = \int_{\theta=0}^{\pi} \tilde{g}_\theta(x\cos\theta + y\sin\theta)\, d\theta$$

What the above is saying, is that to find $f_B(x,y)$ at some x,y position, the angle θ is changed to cover all the angles from zero to 180^0, and for each angle in this interval, the function $\tilde{g}_\theta(p)$ is evaluated at $p = x\cos\theta + y\sin\theta$.

The sum of all these gives f_B at that x, y values. We do these for each x, y value in the region to obtain all the values of f_B. Assume there are N projections made. Hence N angles (since each projection corresponds to one angle). Then the discrete version of the above integral becomes

$$f_B(x,y) = \left\{ \sum_{m=0}^{N-1} \tilde{g}_{\theta_m} (x\cos\theta_m + y\sin\theta_m) \right\} \Delta\theta$$

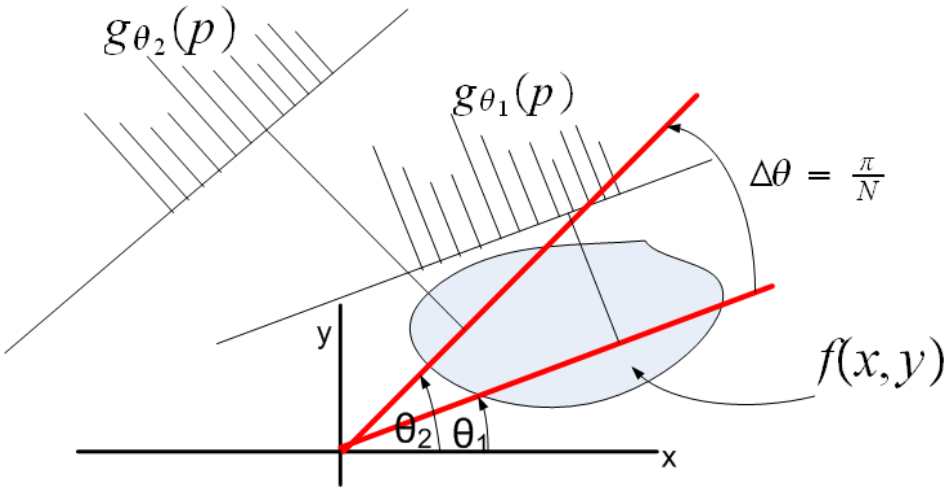
Since there are N angles, then we divide π by N to obtain each specific angle in the range. Hence

$$\theta_m = \frac{\pi}{N} m$$

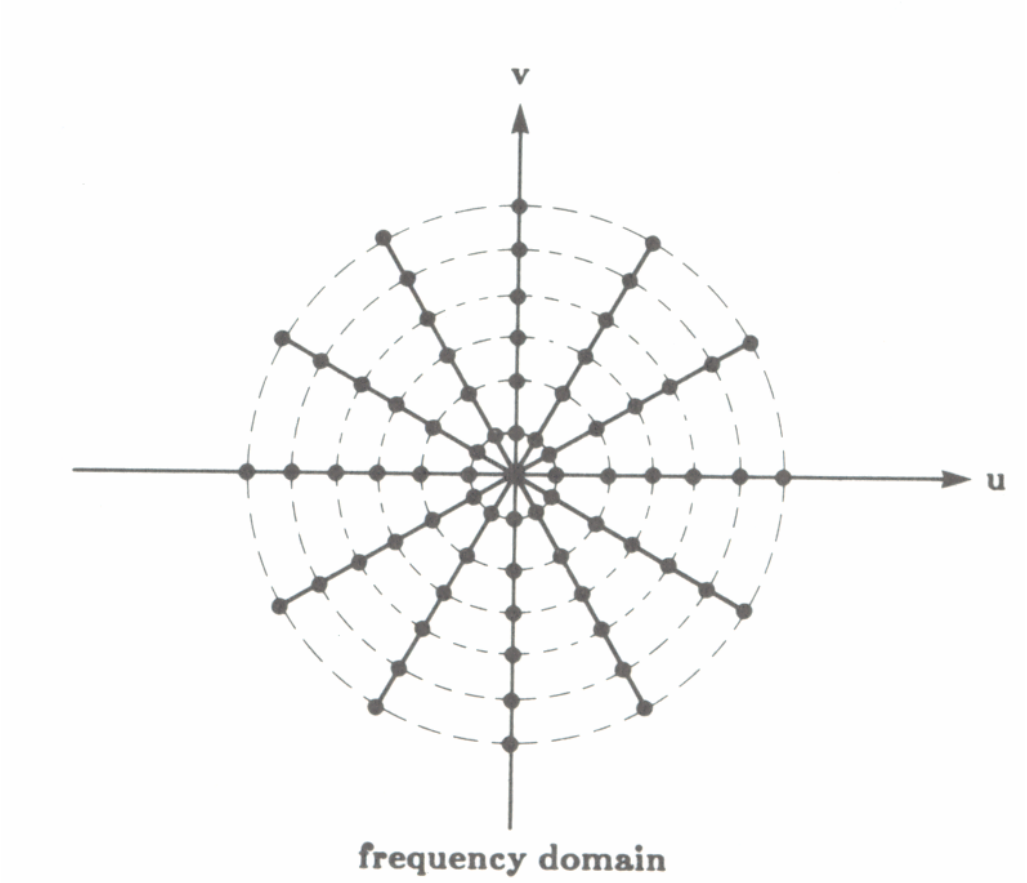
And also we see that $\Delta\theta = \frac{\pi}{N}$ So the sum becomes

$$f_B(x,y) = \frac{\pi}{N} \sum_{m=0}^{N-1} \tilde{g}_{\theta_m} \left(x\cos\left(\frac{\pi}{N}m\right) + y\sin\left(\frac{\pi}{N}m\right) \right) \tag{1}$$

And the above is the equation used for the backprojection. The following diagram illustrates the above.



Notice that in the above, backprojection is carried out in the spatial domain. This is how the Matlab iradon does it. Other way to do backprojection is to stay in the frequency domain by using the central slice theorem. This is done as follows: One the fourier transform of the projection is found, it is moved to the 2D plane which represents the 2D fourier transform of the image being reconstructed by the backprojection. It goes into a radial line through the center of the 2D fourier transform at the same angle θ of the projection. This is done for each projection. The result is a 2D fourier transform of the image. However, it is filled radially. This diagram from Kak illustrates this:



Then filtering is done now in the frequency domain. Next, gridding is carried out to transform the above to a Cartesian coordinates since IFFT works on these only (by interpolation). Then IFFT is done to obtain the final backprojection image.

But since the purpose of this note is to talk about Matlab iradon(), then we will use (1) as the method of backprojection.

2 Matlab iradon and the all-at-once vs the one-at-time

Now we start the investigation on why when using Matlab to determine backprojection from a set of projections it gives different results depending if the call is made in all-at-once vs. one-at-a-time.

Let consider the all-at-once first. From (1) we see that the backprojection image is

$$f_B(x,y) = \frac{\pi}{N}\sum_{m=0}^{N-1} \tilde{g}_{\theta_m}\left(x\cos\left(\frac{\pi}{N}m\right) + y\sin\left(\frac{\pi}{N}m\right)\right)$$

If we call the sum as img, then the above becomes

$$f_B(x,y) = \frac{\pi}{N} img$$

Notice however, that matlab iradon.m does the following at the end: (type iradon.m to see)

$$f_B = \frac{\pi}{2N} img \tag{2}$$

I am not sure now why Matlab divides by an extra 2. I think this might be because if the angles given at from 0 to 360^0, then there is a double counting involved (since the image can be fully constructed from 0 to 180^0, and hence it assumes the angles given have full range of 2π, and so to compensate for doubling the intensity of the image, it divides by an extra 2 at the end.

So, now that (2) gives the backprojection in the all-at-once method, let consider the one-at-time method. Supposed again we have N angles, and we call iradon N times. But each time we call iradon with one angle, we have to pass the same angle twice, and the same projection twice, and divide the result by 2. From help:

```
Compute the backprojection of a single projection vector. The IRADON
    syntax does not allow you to do this directly, because if THETA is a
    scalar it is treated as an increment.  You can accomplish the task by
    passing in two copies of the projection vector and then dividing the
    result by 2.
```

So, in each call now, $N = 2$, and so we write

$$\begin{aligned}
f_{B_1} &= \left(\frac{\pi}{2\times 2} img_1\right)/2 = \frac{\pi}{8} img_1 \\
f_{B_2} &= \left(\frac{\pi}{2\times 2} img_2\right)/2 = \frac{\pi}{8} img_2 \\
&\vdots \\
f_{B_N} &= \left(\frac{\pi}{2\times 2} img_N\right)/2 = \frac{\pi}{8} img_N
\end{aligned}$$

Now we add all the above and obtain

$$f_{B_1} + f_{B_2} + \cdots + f_{B_N} = \frac{\pi}{8}\left(img_1 + img_2 + \cdots + img_N\right) \tag{3}$$

Now, $img_1 + img_2 + \cdots + img_N$ must be twice the img from the call to iradon in the once-at-time method, since using the same assumption as Matlab, where it assumed 0 to 360^0 range and not 0 to 180^0, we will double add the intensity. Hence (3) becomes

$$\begin{aligned}
f_{B_1} + f_{B_2} + \cdots + f_{B_N} &= \frac{\pi}{8}\ (2img) \\
&= \frac{\pi}{4} img
\end{aligned} \tag{4}$$

Compare (4), which is the one-at-time, with (2), which is the all-at-once, we see that

$$\begin{aligned} f_{B_1} + f_{B_2} + \cdots + f_{B_N} &= \frac{\pi}{4} img \\ f_B &= \frac{\pi}{2N} img \end{aligned}$$

Therefore, we see that

$$\begin{aligned} \frac{f_{B_1} + f_{B_2} + \cdots + f_{B_N}}{f_B} &= \frac{\frac{\pi}{4} img}{\frac{\pi}{2N} img} \\ &= \frac{N}{2} \end{aligned}$$

Hence the one-at-time result is $\frac{N}{2}$ times the all-at-once. And this is what was found.

Therefore, when using iradon() to obtain a backprojection in the one-at-time method, we obtain $\frac{N}{2}$ times the image that would have been obtained if using the all-at-once method.

3 References

1. `http://www.comap.com/product/samples/UMM794.pdf`
2. Principles of computerized tomographic imaging by Kak and Slaney
3. Matlab radon and iradon help.