Implementing Fast Hierarchical Back Projection

Jason Chang

ECE 558 – Final Project Paper

May 9, 2007

Abstract – Filtered back projection (FBP) is a commonly used image reconstruction scheme for parallel beam tomography. The algorithm implemented in this project, called fast hierarchical back projection (FHBP), was proposed in [1] as a fast approximation to the direct FBP technique. Through derivation, implementation and simulation results, I will show a complete picture on this novel algorithm for a classical image reconstruction problem. FHBP divides an image into smaller images and uses the basic fact that smaller images require less data to reconstruct. While the direct FBP technique has computational complexity $O(N^3)$, the computational complexity of this FHBP is $O(N^2 \log_2 N)$. The gains achieved for a 512 x 512-pixel image for this specific implementation reached upwards of 100 with a worst case root mean squared error (RMSE) of 13.3/255 gray levels.

TABLE OF CONTENTS

I. Introduction

Image reconstruction in many senses deals with a problem where only measurements of the original image are available. In a wide variety of applications, we need to see inside something without actually being able to open it up. In cases such as these, the minimal amount of intrusions usually points to a reconstruction technique known as filtered back projection (FBP). FBP takes an array of line integrals taken at different angles through an image to approximate a reconstruction of the image. This specific approach to image reconstruction is used in many applications, especially in medical imaging.

The direct FBP approach has a computational complexity on the order of $O(N^3)$. Currently implemented algorithms in the field draw on various proportional gains that can be realized with parallelizing the problem. However, these gains are still unable to improve the computation complexity of the algorithm. With the increasing technologies, data acquisition rates for images and the demand for higher resolution imaging are both increasing, and past algorithms for FBP are quickly becoming out of date for the specific applications. The algorithm implemented here, called fast hierarchical back projection (FHBP) [1], is a novel idea in improving reconstruction time without much loss in quality.

II. BACKGROUND OF FILTERED BACK PROJECTION

The direct FBP approach can be broken down into a few basic steps. Figure 1 shows a block diagram of how direct FBP is computed.

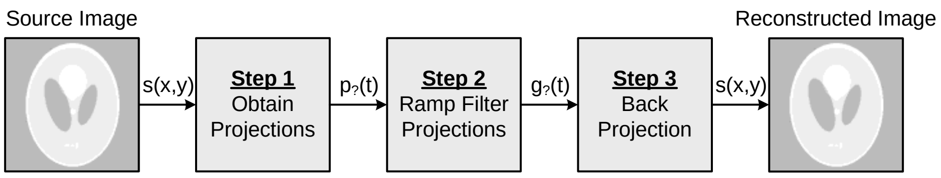


Fig. 1. FBP Block Diagram

FBP has been used for many years already, and the fact that certain parts of this reconstruction technique are slow is not a new development. Many researchers have been working on developing a new and fast algorithm for this process. However, [1] indicates that many of these new methods are visually unsuccessful in reproducing an accurate estimation of the image. In addition, the computation gain from these algorithms pales in comparison to the gains achieved by FHBP. Therefore, I focused on the implementation and optimization of FHBP.

To understand how FHBP works, we need to first understand the intricate details of FBP. We will not go through these details to understand where speed improvements can be made.

II-A. Step 1: Obtain Projections

This step in the real world application is the data acquisition step. In the ideal reconstruction case, this corresponds to taking the continuous Radon Transform of the image to produce an infinite number of line integrals. In this analog case, we integrate perpendicularly to a given angle and all radial distances. However, because this continuous quantity is unrealizable and impractical, a discrete approximation is used. Figure 2 illustrates one angle of the discrete Radon Transform.

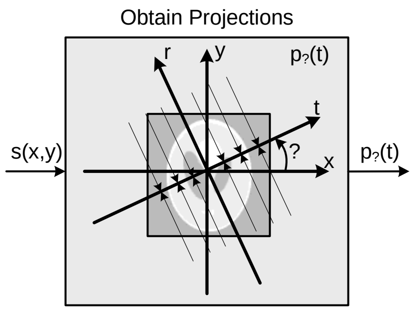


Fig. 2. Discrete Radon Transform

In general, for close to perfect reconstruction, we need to sample at the Shannon-Nyquist sampling rate. In an image of size *N pixels*, this usually corresponds to acquiring $P \approx 2N$ projection angles, and $Np \approx 2N$ points per projection angle. It should be noted that the actual minimal sampling rate is a little less than this, but we usually use a factor of 2 to sample a little bit above the minimum.

II-B. Step 2: Ramp Filter Projections

When we combine all the projection angles, we end up with an image that will be somewhat blurred. This is due to the fact that we have very concentrated samples of the image towards the center, and more sparse samples near the edges. Figure 3 illustrates this point. To compensate for this mismatch in sampling, we apply a filter to the output. It can be easily shown that the optimal ideal filter to be used is just a ramp filter (show in Figure 4).

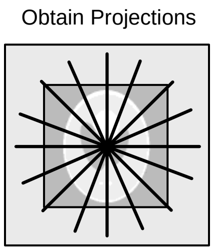


Fig. 3. Complete Radon Transform

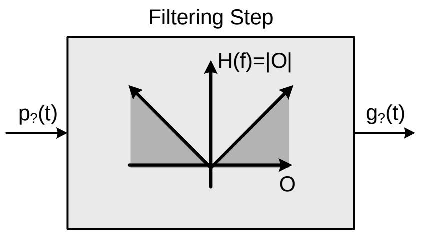


Fig. 4. Ramp Filter

II-C. Step 3: Back Projection

After the ramp filtering, the projections are now weighted correctly. We can now back project the projections to reform the image. We can think of this back projecting as a back "smearing", where we take each radial sample at each angle, and smear it along the path that we integrated. Summing all these smears over all radial distances and angles will give an approximation to the original image. In a more mathematical formulation, the back projecting has the following form for each pixel.

$$\hat{s}(x,y) = \sum_{p=1}^{P} g_{\theta_p}\left(x\cos\theta_p - y\sin\theta_p\right) \tag{1}$$

In other words, to compute the back projection for the entire image, we need to loop over each pixel and compute (1) for it. This is just a summation over *P* elements a total of N^2 times. Therefore, we can conclude that

$$O\left(PN^2\right) \tag{2}$$

Knowing that *P* is only a scalar factor from *N*, we see that the computation complexity of the back projection step is $O(N^3)$.

III. Background of Radon Transforms

One of the most important things to know before developing the FHBP algorithm is a complete understanding of the Radon Transform in detail. Specifically, we will consider the spectral support, or Discrete Time Fourier Transform (DTFT), of the Radon Transform here. From [2], we see that the DTFT of the Radon Transform can be approximated as a bow-tie shaped support (pictured in Figure 5), where *B* is the bandwidth of the image, *W* is the radial support of the image, and *P* is the number of projection angles taken.

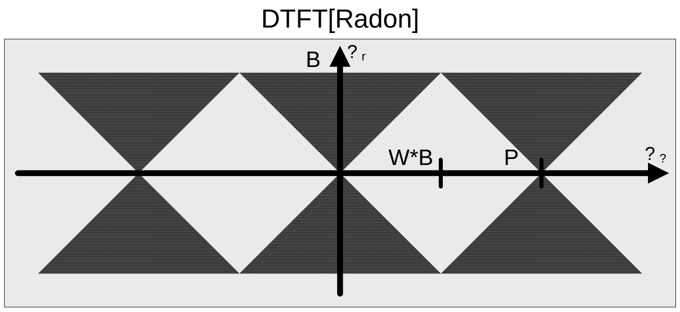


Fig. 5. Discrete Time Fourier Transform of a Radon Transform

W is easily found by looking at the size of the original image.

$$W = \frac{N\sqrt{2}}{2} \tag{3}$$

As stated previously (Step 1 of FBP), the Shannon-Nyquist sampling theorem shows that we only need to sample at twice the highest frequency for perfect reconstruction. In other words, because the highest frequency in the angular direction is *WB*, we have

$$P > 2BW \tag{4}$$

$$P > BN\sqrt{2} \tag{5}$$

With the knowledge of FBP and spectral supports of Radon Transforms, we can not derive the fast algorithm, FHBP.

IV. FAST HIERARCHICAL BACK PROJECTION

From now on, we will refer to the direct filtered back projection approach as the direct approach, and the fast hierarchical back projection approach as the FHBP approach. As stated previously (Step 3 of FBP), the back projection step of the direct approach is the most time consuming (ramp filtering can be done in $O(Nlog_2N)$ time with Fast Fourier Transforms). Therefore, FHBP focuses on optimizing this step. In this section, we will focus on the development on this algorithm. Specifically, we will consider viewing the derivation from multiple ways. In Section IV-A, we will consider out intuition with FBP. Then, in Section IV-B, we will then evaluate the gains that can be achieved more concretely with Fourier Domain analysis. Finally, in Section IV-C, we will consider assumptions used in deriving FHBP that make it only an approximation.

IV-A. Intuition

Examining (5) closely reveals that the minimum number of projection angles, *P*, needed to reconstruct the image is directly proportional to the image size, *N*. This intuitively makes sense because smaller images require less data to reconstruct. However, a more important conclusion that we can draw from the Shannon-Nyquist Criterion is that, if we could somehow find the projections to only a small portion of an image, then we can reconstruct that small portion of an image with fewer angles than needed for the entire image. Here we make our first assumption that we can indeed obtain projections of a small portion of an image from the entire set of projections. We will see in a later section that this assumption is fairly accurate. With this, we can then divide an image into smaller parts, reconstruct each part with fewer angles, and then combine them to form the entire estimated image.

To decrease the number of projection angles we have, all we need to do is downsample the projections. Each downsampling creates a significant amount of savings. If we apply this dividing of the image recursively (until we reach a single pixel), then we can downsample a total of log_2P times. Using (2) we see that the computational complexity of this algorithm is just $O(N^2log_2P)$. We will now discuss the formulation of this argument in both the spectral and spatial domain.

IV-B. Fourier Domain Analysis

We can analyze these concepts more concretely by looking at the spectral support of various Radon Transforms. Referring back to Figure 5, we know that the spectral support of the Radon Transform is in the form of a bow-tie. Figure 6 shows the spectral support of the Radon Transform with the same number of projection angles but on a smaller image of half the length.

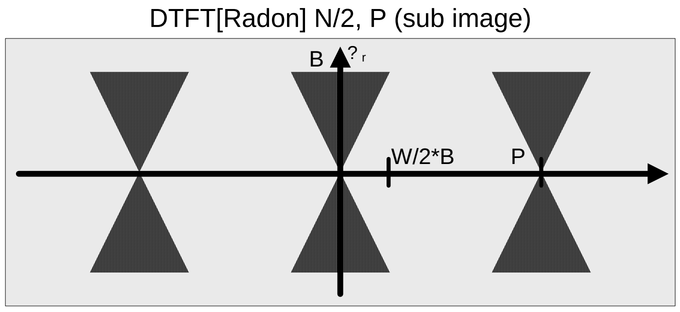


Fig. 6. Spectral Support of Radon Transform of Sub-Image

Clearly, we see that if the spectral support of the image is indeed a perfect bow-tie, then we are oversampling the Radon Transform in the angular direction by a factor of 2. For this reason, we can downsample our projection angles by a factor of two, and in the ideal case, have absolutely no loss. The new spectral support of the downsampled Radon Transform is shown in Figure 7.

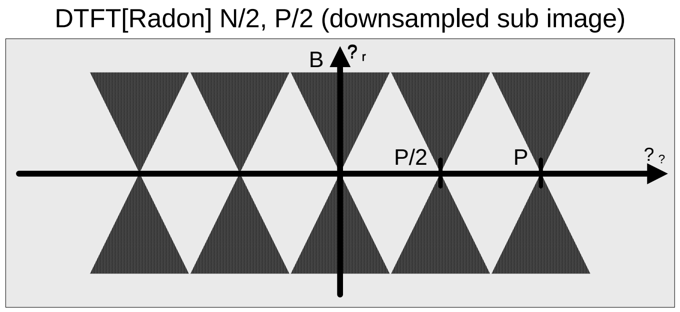


Fig. 7. Spectral Support of Downsampled Radon Transform of Sub-Image

However, downsampling is not actually as simple as this spectral domain analysis may lead us to believe. If we look back at our assumption in the formulation of this algorithm, we assumed that we could perfectly retrieve projections to only a small portion of an image from projections of the entire image. This assumption, although rather good, is actually false. We will now discuss this matter, and explore why the algorithm is an approximation.

IV-C. Assumptions & Approximations

The only important assumption we made in formulating FHBP was that we could perfectly segment projections of the entire image into projections of sub-images. The reason why we need to segment the projection data is because we are splitting the image into sub-images, and need projections through only that sub-image. Figure 8 shows the problem statement on a sample image (where the sub-image we want to back project is the darkened bottom right quadrant). Here, the dotted line of the line integral is the unwanted portion of the entire projection, and the solid line is the wanted segment of the projection for the specific sub-image. Note that the reason why we want to segment the projection is because the dotted line contains data that is irrelevant to reconstructing the sub-image, where as the solid line contains relevant data.

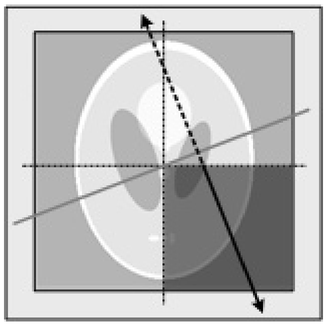


Fig. 8. Desired Segmenting of Projection

Even though the perfect segmentation of a projection is desired, we will show here that this is not possible. However, a simple method can be used to approximate this unrealizable computation.

First, consider the continuous Radon Transform of an image. We will look at a single projection angle, and a single point in that projection. This single projection is represented in Figure 9.

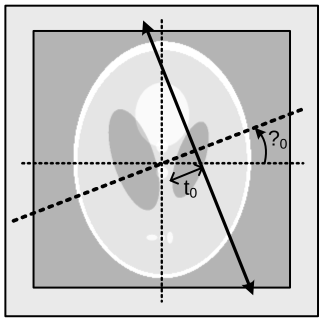


Fig. 9. Single Projection Point on Radon Transform

The Radon Transform can be expressed by the following mathematical expression:

$$RT\left[s(x,y)\right] = S_R(\theta,t) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} s(x,y)\,\delta(x\cos\theta + y\sin\theta - t)\,dxdy \qquad (6)$$

Here, we are considering only one angle and one radial point limited to the confines of the image dimensions. This simplifies (6) to be:

$$S_R(\theta_0,t_0) = \int_{-\frac{N}{2}}^{\frac{N}{2}}\int_{-\frac{N}{2}}^{\frac{N}{2}} s(x,y)\,\delta(x\cos\theta_0 + y\sin\theta_0 - t_0)\,dxdy \qquad (7)$$

We see from (7) that we are representing an entire vector of luminosities by just one scalar value. Clearly, there is a loss of data incorporated with this line integral. Thus, it should be intuitively apparent that we can never perfectly retrieve a line integral over a portion of the image from the scalar value representing the line integral over the entire image. In fact, there is nothing we can do to retrieve a portion of this line integral when only considering one angle and one radial distance. However, we can use information from adjacent angles to estimate a segment of the line integral.

If we consider one radial point on two adjacent angles (shown in Figure 10), we can take advantage of the correlation between two pixels that are close together.

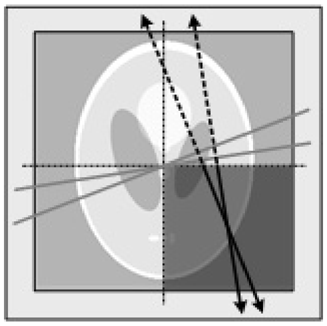


Fig. 10. Adjacent Projections

Notice that the dotted lines are much father apart in distance, especially as they reach the top of the image. Also, notice that the solid lines are fairly close in distance comparatively. Using prior background knowledge that pixels close in distance are highly correlated, we can assume that the pixels along the solid line are highly correlated, while the ones along the dotted lines are uncorrelated. Correlation can be related to frequency components directly, where high correlation relates to strong low-frequency components, and no correlation relates to strong high-frequency components. Therefore, this tells us that we can use a low-pass filter to eliminate the uncorrelated samples of the projections outside of the sub-image.

In the perfect case, we can use an ideal low pass filter, *ψ(n)*, to basically eliminate all projections outside of the sub-image [1]. This ideal low-pass filter is just a rectangle function defined as follows.

$$rect(t) = \begin{cases} 1 & |t| < \frac{1}{2} \\ 0 & else \end{cases} \tag{8}$$

$$\Psi(\omega) = rect\left(\frac{\omega}{\pi}\right) = \begin{cases} 1 & |\omega| < \frac{\pi}{2} \\ 0 & else \end{cases}$$

To find the spatial domain response of the filter, we take the Inverse Discrete Time Fourier Transform given by (9).

$$DTFT^{-1}\{H(\omega)\} = h(n) = \frac{1}{2\pi}\int_{-\pi}^{\pi} H(\omega) e^{j\omega n} d\omega \tag{9}$$

$$\psi(n) = DTFT^{-1}\{\Psi(\omega)\} = \frac{1}{2\pi}\int_{-\pi}^{\pi} \Psi(\omega) e^{j\omega n} d\omega = \frac{1}{2\pi}\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} e^{j\omega n} d\omega$$

$$\psi(n) = \frac{1}{\pi n}\sin\left(\frac{\pi n}{2}\right) = \frac{1}{2}\text{sinc}\left(\frac{\pi n}{2}\right) \tag{10}$$

We know that the sinc function is infinitely long. Thus, we will only use an approximation of *ψ(n)*, with a finite number of filter taps. This is the first approximation that we make, and is one reason why the algorithm is not an exact solution.

In addition to this approximation for the angular smoothing function, we also note that there are a few other approximations that have not been mentioned thus far. In the formulation with the spectral domain of the Radon Transforms, we stated that the spectral support of the Radon Transform is shaped as a bow-tie. In reality, this bow-tie shape is only an approximation to the projections. Thus, this approximation means that when we downsample and low-pass filter, we may be eliminating high frequency components of the actual image.

The final approximation that we make is when we change from the projections of the image as a whole to the sub-image. As noted previously, we already considered the fact that projections may contain data from outside the desired sub-image support. However, we are also using a discrete Radon Transform, and as such, we lose data in this step in another way. When we consider a sub-image, we must first shift the projections so that they correspond (centered at) the new sub-image. Each shift depends directly on the image size and the projection angle. In other words, we must shift each projection by a separate amount. Figure 11 illustrates this varying radial shift.

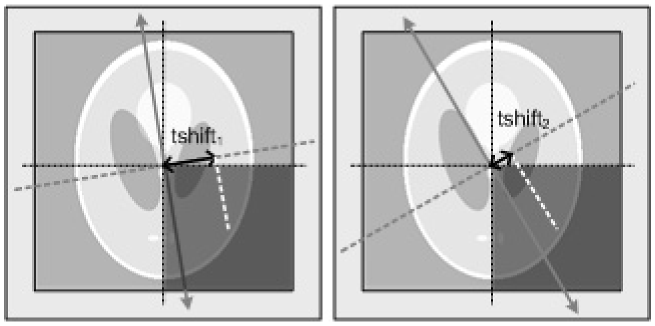


Fig. 11. Varying Radial Shifts

The equation to find the amount of radial shift can be easily found geometrically to be the following (where *p* is the index of the angle, and *q* is the quadrant shifting to).

$$tshift_{p,q} = \begin{cases} \frac{N}{2\sqrt{2}}\cos\left(\frac{\pi}{4}-\theta_p\right) & ,q=1 \\ -\frac{N}{2\sqrt{2}}\cos\left(\frac{\pi}{4}+\theta_p\right) & ,q=2 \\ -\frac{N}{2\sqrt{2}}\cos\left(\frac{\pi}{4}-\theta_p\right) & ,q=3 \\ \frac{N}{2\sqrt{2}}\cos\left(\frac{\pi}{4}+\theta_p\right) & ,q=4 \end{cases} \tag{11}$$

The approximation that comes with these radial shifts is that these shifts are seldom integers or even rational values. Thus, we usually need to shift the projections by a number of projections that is not an integer. Consider a simplified example to illustrate how the shifting works.

Assume that *N=8*, we are trying to shift to the first quadrant, and we are looking at one specific angle (say *θp=0*). We first need to determine the shift amount, which can easily be computed to be N/4=2. One may think that this would be a nice even shift, but that would be looking over the fact that shifts need are determined not by actual distance, but by the number of points to shift *t* by. Now, we assume that we used *Np=16*, or 16 projections per angle. These points are evenly spaced on the range spanning the radial support of the image.

$$t \in \left[-\tfrac{N\sqrt{2}}{2}, \tfrac{N\sqrt{2}}{2}\right] = \left[-5.66, 5.66\right] \tag{12}$$

Thus, the distance between each radial sample is just

$$tstep = \frac{N\sqrt{2}}{N_p - 1} \tag{13}$$

Therefore, for this example, our vector of radial distances is just

$$\begin{bmatrix}-5.7 & -4.9 & -4.2 & -3.4 & -2.6 & -1.9 & -1.1 & -0.4 & 0.4 & 1.1 & 1.9 & 2.6 & 3.4 & 4.1 & 4.9 & 5.7\end{bmatrix}$$

As computed earlier, we need to shift the radial distances by a distance of 2. Clearly each step is not 2, thus we must interpolate to the following radial distance vector

$$\begin{bmatrix}-3.7 & -2.9 & -2.2 & -1.4 & -0.6 & 0.1 & 0.9 & 1.6 & 2.4 & 3.1 & 3.9 & 4.6 & 5.4 & 6.1 & 6.9 & 7.7\end{bmatrix}$$

Clearly, we will have to interpolate at almost every angle. A fast implementation of interpolation just uses a linear interpolation, and thus, we see the final approximation used in this algorithm.

V. Parameters in Fast Hierarchical Back Projection

FHBP is an approximation to a very slow algorithm. As such, there are a few parameters that the user can choose to trade speed for accuracy. We will briefly go over these parameters and quickly explain its effects on speed and accuracy. For a more complete understanding, please refer to [3].

V-A. Recursive Downsampling Level (Q)

As shown in Section IV, the speed gain in FHBP is from downsampling the number of projection angles recursively. One of the approximations that was made is that the bow-tie structure in Figure 5 is not completely accurate in describing the spectral support of the Radon Transform. Recall that this caused aliasing artifacts in the reconstruction when too much downsampling was used. Therefore, to eliminate some of this reconstruction noise, we can begin downsampling only after a certain number of levels in the recursion. We define this number of non-downsampling recursive levels as the parameter *Q*.

As *Q* is increases, we are basically oversampling the image above the Shannon-Nyquist sampling rate. This will reduce the aliasing artifacts from the bow-tie support approximation. However, it was shown in [2] that as *Q* increases, the speed increases exponentially. The quality of the reconstructed image does increase, but the more significant increase in speed is usually not worth the slight improvement in quality. Therefore, *Q* is usually chosen to be very small (if not 0), and other parameters are change to try and improve speed.

V-B. Radial Interpolation Factor (M)

Another quality gain that we can achieve is to radially interpolate the projections by a factor of *M* prior to backprojecting. This improvement occurs when we apply the ideal low-pass filter when downsampling the projection angles. This improvement in quality relies on the fact that downsampling and then interpolating radially is different from interpolating angularly and then downsampling. It can easily be shown that the latter method provides a better estimate to the projection at the interpolated radial distance. Figure 12 illustrates the difference in the interpolation. "o" marks a point that is obtained from the Radon Transform, and "x" marks an interpolated point. The arrows that point to each interpolated point come from the two points used in the linear interpolation.

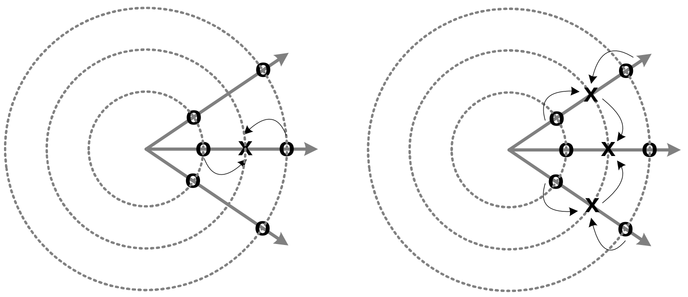


Fig. 12. Interpolation Comparison

Therefore, if we upsample and interpolate the projections radially prior to backprojecting, we can achieve a gain in accuracy. This gain in accuracy is comparable to the gain in choosing *Q*.

However, this method is preferred to improve quality because the speed increases only linearly as *M* is increased.

V-C. Ideal Low Pass Filter Length

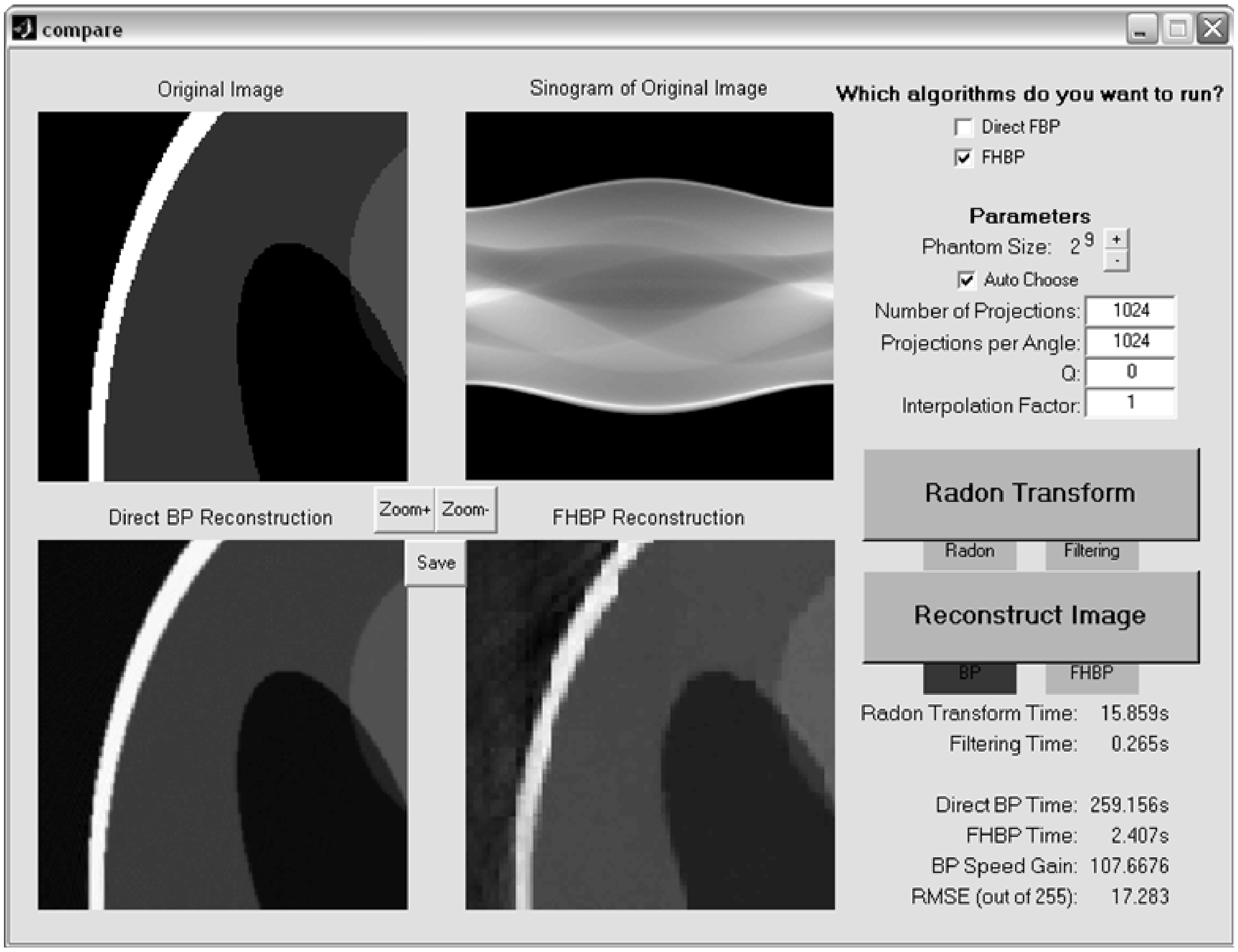
The last parameter that the user can change is the ideal low pass filter length. As stated in Section IV-C, in the perfect case, an ideal sinc filter would be implemented. However, due to the infinite span in the time domain, a small filter length is usually used for fast computation time. If quality needs to be improved, the user can increase the filter length to better approximate the ideal filter. This parameter is much harder to change in the code because of the boundary conditions of the projections, which will be mentioned in the implementation section later. However, increasing filter length could potentially increase the quality (if needed) after a specific *Q* and *M* are chosen.

VI. Implementation of Fast Hierarchical Back Projection

A graphical user interface (GUI) is provided for the user to choose various parameters and examine results of the back projection. The GUI provides the option to do both the direct back projection and the fast hierarchical back projection. In each case, the size of the image, the number of angles, and the number of points per angle can all be chosen. For FHBP, the parameters *Q* and *M* can be specified to produce separate results.

When the program completes the back projection, the original image and reconstructed images will be displayed. To examine the results at a better level, the user can highlight a specific section of any of these three images, and click on the “Zoom +” button to zoom into all the images in the same location. This allows for the user to compare a very detailed image side by side. To return to the entire image, the user only needs to click on the “Zoom –” button. In addition, the user can save the original image and reconstructed images to a file by clicking on the “Save” button.

Figure 13 is a screenshot of a zoomed reconstructed image. The speed gain between the direct approach and the FHBP approach is calculated automatically, along with the root mean squared error (RMSE). The RMSE gives the expected value of the gray level error on a scale of 255.



Which algorithms do you want to run?
Direct FBP [ ]
FHBP [✓]

Parameters
Phantom Size: 2^9
Auto Choose [✓]
Number of Projections: 1024
Projections per Angle: 1024
Q: 0
Interpolation Factor: 1

Radon Transform Time: 15.859s
Filtering Time: 0.265s

Direct BP Time: 259.156s
FHBP Time: 2.407s
BP Speed Gain: 107.6676
RMSE (out of 255): 17.283

Fig. 13. Graphical User Interface for FHBP Comparison

The purpose of FHBP is to be a fast algorithm. Therefore, I decided to implement the algorithm in C++ instead of Matlab for speed, while still using Matlab to interface with the user for its display functions and user interface options. This required compiling C++ files into a dynamic link library (DLL) so that Matlab could call them. [4]-[6] contain tutorials and sample code to be able to interface Matlab with C++.

In the C++ back projection function, I used recursion to split up the image into smaller and smaller images. For each stage, I did the following steps:

1) Shift the projections to correspond to the sub-image
2) Truncate the radial samples to the region of interest covering the sub-image
3) Angularly downsample and low-pass filter
4) If size of image is greater than a pixel, call function recursively on each quadrant
5) Else backproject onto the one pixel

It should be noted that when we low-pass filter the angularly downsampled projections, we need to worry specifically about the boundaries of the projections (θ_0, θ_{P-1}, etc.). When we use a three tap filter on the first angle, we need to wrap around back to the last angle (assuming angles in the range $[0, \pi)$). Also, when we wrap around, we actually want to look at the negative radial distance ($-te^{j\pi} = te^{j\pi}$). Therefore, the length of the filter is not easily changed. For increasing taps, the wrap around to other angles needs to occur for more angles in the beginning and at the end.

The software flow diagram is shown in Figure 14. The code was optimized manually to achieve fastest results in both the direct approach and the FHBP approach.

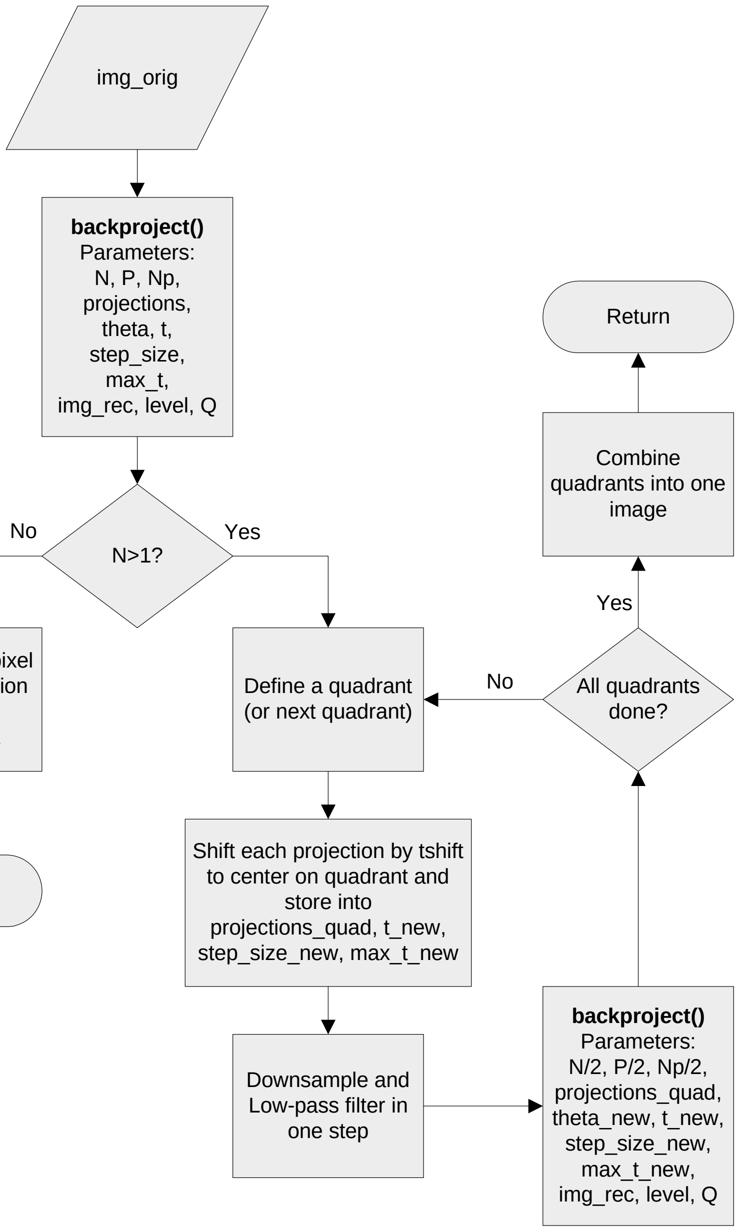

img_orig
backproject() Parameters: N, P, Np, projections, theta, t, step_size, max_t, img_rec, level, Q
N>1?
No
Yes
Loop through pixel with all projection angles and backproject
Return
Define a quadrant (or next quadrant)
Shift each projection by tshift to center on quadrant and store into projections_quad, t_new, step_size_new, max_t_new
Downsample and Low-pass filter in one step
backproject() Parameters: N/2, P/2, Np/2, projections_quad, theta_new, t_new, step_size_new, max_t_new, img_rec, level, Q
All quadrants done?
No
Yes
Combine quadrants into one image
Return

Fig. 14. Software Flow Chart

VII. Results and Comparisons

In general, FHBP reconstructed images did not differ too much from the direct approach. In most cases, parameters could have been chosen to provide both a significant speed gain, and almost no visual defects whatsoever. Throughout the simulations it seemed like the best results (weighting the speed gain and the visual appearance) occurred when *Q=0* and *M=4*. These values were used in the following figures to reconstruct an image of the Shepp-Logan phantom image.

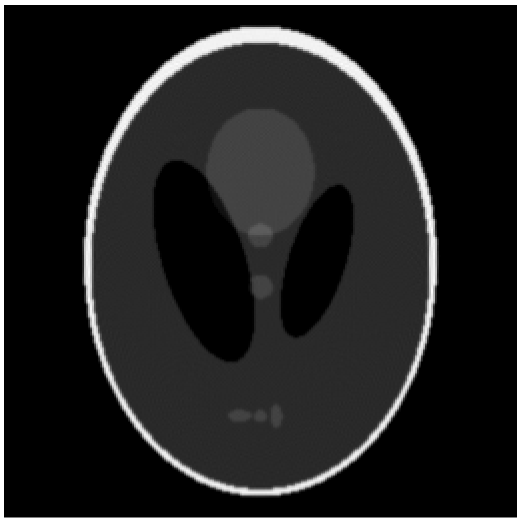


Fig. 15. Direct FBP with
N=256, P=Np=512

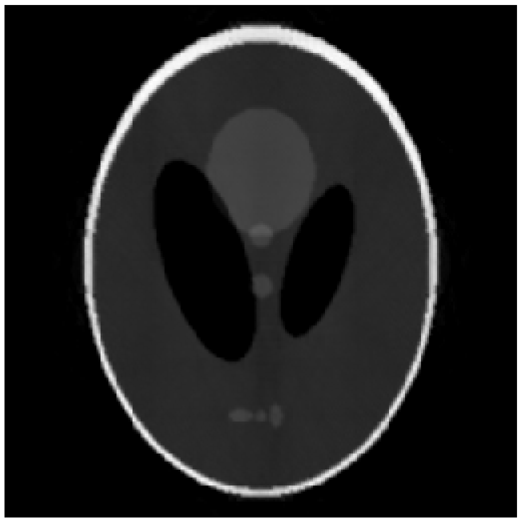


Fig. 16. FHBP with
N=256, P=Np=512, Q=0, M=4

Table 1 summarizes multiple simulations done on the phantom image. Clearly, the parameter *Q* greatly affects the speed gains, just as expected. The gains from increasing *Q* are actually fairly small (on the order of a few gray levels). However, the gains from choosing a good value for *M* are much more beneficial. Increasing *M* from 1 to 4 greatly increases the quality of the reconstruction without too much loss in speed gain. Therefore, this parameter can be optimized to provide better results, and *Q* should stay at fairly low, if not 0. However, if extraordinary quality is needed, the user may still want to increase *Q* slightly for a minor boost in accuracy.

Table 1. Simulation Gain and Error Results

N	P	Np	Q	M	Gain	RMSE
128	256	256	0	1	28.8	28.0
				2	21.4	16.3
				3	18.3	12.2
				4	16.0	10.4
				5	14.3	9.8
			2	1	9.1	28.2
				2	8.2	15.5
				3	7.1	10.2
				4	6.5	8.1
				5	6.0	7.3
			4	1	2.7	28.3
				2	2.5	15.6
				3	2.3	10.3
				4	2.2	7.9
				5	1.9	6.4
256	512	512	0	1	54.1	21.4
				2	44.7	12.8
				3	36.5	10.3
				4	32.7	8.9
				5	29.1	8.1
			2	1	18.0	21.8
				2	15.6	12.4
				3	13.7	8.8
				4	12.2	6.9
				5	11.3	6.3
			4	1	5.1	21.7
				2	4.6	12.1
				3	4.3	8.3
				4	3.8	6.5
				5	3.7	5.3
512	1024	1024	0	1	102.8	17.3
				2	83.5	10.8
				3	71.8	9.0
				4	60.5	7.8
				5	51.7	7.4
			2	1	34.47	17.0
				2	29.3	9.9
				3	25.0	7.1
				4	21.0	6.2
				5	19.4	5.5
			4	1	9.7	16.7
				2	8.5	9.5
				3	7.5	6.6
				4	6.7	5.2
				5	6.2	4.3

VIII. Conclusions and Future Work

Fast hierarchical back projection has proven itself to be a very fast and accurate algorithm to approximate filtered back projection. The gains it achieves in speed surpass other algorithms, and the few artifacts in the reconstructed image are greatly outweighed by the speed gains.

This comparison of FHBP to the direct method was coded completely in C++ for this implementation. However, for a more accurate comparison of FHBP with the current methods of FBP, one would need to work with the parallelization that is used in current technologies for FBP. This could be rather hard to implement with only one computer, but it is something to keep in mind for the future.

Also, much can still be done to improve FHBP. Specifically, FBP may have been slightly parallelizable, but FHBP is completely parallelizable. Because of its divide and conquer strategy, each stage can be split amongst 4 separate processors, and computed simultaneously. The potential gains of parallelizing FHBP are astronomical, and if ever implemented in a real world application, this topic should definitely be exploited.

In addition to parallelizing the computation, FHBP can be improved even more by delving deeper into the code. If the algorithm is programmed in Assembly and parallelized, FHBP will be the fastest possible. By increasing reconstruction time, the gains that would result from these two future methods will greatly change how we think about reconstruction problems for the future.

IX. References

[1] S. Basu and Y. Bresler, "O($N^2\log_2N$) Filtered Backprojection Reconstruction Algorithm for Tomography," *IEEE Trans. Image Processing*, vol. 9, pp. 1760-1773, October 2000.

[2] P. A. Rattey and A. G. Lindgren, "Sampling the 2-D Radon Transform," *IEEE Trans. Acoustic, Speech, and Signal Processing*, vol. ASSP-29, pp. 994-1002, October, 1981.

[3] S. Basu and Y. Bresler, "Error Analysis and Performance Optimization of Fast Hierarchical Backprojection Algorithms," *IEEE Trans. Image Processing*, vol. 10, pp. 1103-1117, July 2001.

[4] P. Carbonetto, "Creating, Compiling and Linking Matlab Executables (MEX files): A Tutorial," December 2006, http://www.cs.ubc.ca/~pcarbo/mex-tutorial/index.html.

[5] L. Ko, "Importing C++ Functions to Matlab," June 2005, http://www2.enel.ucalgary.ca/People/Smith/embeddedTDD/LKoTDD_TutorialSept05/ImportCFunctionsMatlab.htm.

[6] The Mathworks, "Support – MEX-files Guide," May 2007, http://www.mathworks.com/support/tech-notes/1600/1605.html.