ICASSP'05, IEEE International Conference on Signal Processing, vol. V, Pages:153-156, Philadelphia, PA, March 18-23, 2005.

PERFORMANCE ANALYSIS OF THE FILTERED BACKPROJECTION IMAGE RECONSTRUCTION ALGORITHMS

Thammanit Pipatsrisawat, Aca Gačić, Franz Franchetti, Markus Püschel, and José M. F. Moura

Department of Electrical and Computer Engineering
Carnegie Mellon University, Pittsburgh, U.S.A.

ABSTRACT

We investigate performance tradeoffs for a class of filtered backprojection (FBP) image reconstruction algorithms. The recently developed fast hierarchical backprojection asymptotically achieves the same $\mathcal{O}(N^2 \log N)$ cost as Fourier-based methods while retaining many advantages of the FBP technique. In this paper, we provide a detailed cost and performance analysis of the algorithm on a general purpose platform. Based on carefully tuned implementations of both the direct and the hierarchical backprojection, we explore the tradeoffs between distortion and runtime by varying several algorithm and implementation choices. Experimental results show that, given the desired performance, the choice of algorithm parameters is not obvious and largely depends on the image properties and the underlying computer platform.

1. INTRODUCTION

Many imaging techniques are based on reconstructing an image from data that can be interpreted, either directly or after some preprocessing, as a set of projections of the imaged object. Most notable examples are medical imaging techniques such as computed tomography (CT) and spotlight-mode synthetic aperture radar (SAR) imaging. Several approaches to the reconstruction problem have been proposed, of which the transform-based methods have been most common in practical use. The mathematical foundation is provided by the Radon transform (RT) which computes 1-D projections of a 2-D data at different view angles. In CT imaging, for example, the data is obtained by passing a set of narrow X-ray beams through the scanned object and collecting their intensities using an array of sensors. The acquired data represents the Radon transform of the cross-sectional absorbtion densities that form the image [1]. Munson et al. [2] showed that the data collected by the SAR, after demodulation and lowpass filtering, represents the Fourier transform of the projections obtained from the reflectivity density of the targeted ground patch.

For the transform-based approach, the reconstruction problem can be viewed as the problem of inverting the Radon transform. Two main methods have been studied and used most. The direct Fourier methods compute the Fourier transform of the projections, interpolate the data from polar to Cartesian grid, and reconstruct the image using the inverse 2-D FFT with the total cost of $\mathcal{O}(N^2 \log N)$. However, most popular in practice are methods based on the *backprojection* (BP) operation which reduces the distortion by avoiding the interpolation step. The BP approach is also more suitable to handle other problems such as wavefront curvature effects in SAR imaging. In the popular filtered backprojection (FBP) algorithm, the projections are first filtered and then backprojected to reconstruct the original image (e.g., [1]). The FBP algorithm typically provides better accuracy at a higher $\mathcal{O}(N^3)$ cost. Basu and Bressler [3] developed a new method for the parallel-geometry FBP based on hierarchical decomposition and angular downsampling of the backprojection operation (the HBP method), which asymptotically reduces the cost to the desired $\mathcal{O}(N^2 \log N)$. However, the cost reduction achieved in the HBP algorithm comes at the expense of an increased level of distortion. Furthermore, because of the higher complexity of the data flow, it is not clear when the lower computational cost translates to a more efficient implementation.

This work was supported by NSF through awards 0234293, 0310941, and 0325687. Franz Franchetti was supported by the Austrian Science Fund FWF's Erwin Schroedinger Fellowship J2322.

In this paper, we focus on the backprojection (BP) operation, the computational bottleneck of the FBP algorithm. First, we provide a detailed cost analysis of both the direct BP and a parameterized HBP. We then present optimized implementations of these algorithms which provides the flexibility to investigate different algorithm choices. The experimental results are aimed at showing that careful performance analysis is needed to determine the efficiency of the backprojection algorithms, w.r.t. both runtime efficiency and distortion. Necessary steps include careful tuning of code, exploring various algorithmic and implementation degrees of freedom, and performing tests on a diverse set of images.

2. FILTERED BACKPROJECTION ALGORITHMS

The Radon transform (RT) represents a set of parallel line integral projections of a 2-D function $f(x, y)$ at different angles θ. The continuous Radon transform is defined by

$$\hat{f}(r, \theta) = \int\int f(x, y)\delta(r - x\cos\theta - y\sin\theta)dx\, dy, \qquad (1)$$

where r and θ are polar coordinates and δ is the unit impulse. The projections $\hat{f}(r, \theta)$ are also referred to as the data *sinogram*. In their original form, filtered backprojection (FBP) algorithms are based on the well-known inversion formula for the RT:

$$f(x, y) = \mathcal{B}\tilde{f}(\rho, \theta), \quad \tilde{f}(\rho, \theta) = \mathcal{F}_r^{-1}|\omega_r|\mathcal{F}_r\hat{f}(r, \theta). \qquad (2)$$

Here, $\hat{F}(\omega_r, \theta) = \mathcal{F}_r\hat{f}(r, \theta)$ represents a 1-D Fourier transform in the variable r, and $\mathcal{B}$ is the continuous backprojection operator

$$f(x, y) = \mathcal{B}\tilde{f}(\rho, \theta) = \int_0^\pi \tilde{f}(x\cos\theta + y\sin\theta)\, d\theta. \qquad (3)$$

The projections $\hat{f}(r, \theta)$ are first filtered using the ramp filter $|\omega_r|$ and then backprojected to reconstruct the image.

Discrete direct backprojection. In practice, the number of projections P and the sampling distribution are determined by the data acquiring equipment, and the reconstructed image is discrete. We will assume that the projection angles θ are evenly distributed in the interval $[0, \pi)$ and that all images are square with $N \times N$

pixels. To implement the FBP algorithm on a computer, the backprojection operation is discretized and the ramp filter is windowed and sampled. The discrete backprojection is performed for each pixel $f(m,n)$ as a sum of projected values over all angles θ:

$$f(m,n) = \sum_{\theta} \tilde{f}(m\cos\theta + n\sin\theta, \theta). \qquad (4)$$

Interpolation with a kernel $\phi(\rho)$ in the radial direction is required to compute sampled $\tilde{f}$ at non-integral values. Better approximation to the continuous backprojection can be achieved by introducing the image sampling operator to model the physical properties of the sensing equipment [3]. In our implementation, however, we use the ideal sampling kernel.

Hierarchical backprojection. The hierarchical backprojection (HBP) algorithm uses the fact that the angular bandwidth of the sinogram is proportional to the size of the image [4]. Basu and Bresler [3] showed that, under certain assumptions on the image, the problem can be recursively decomposed into a problem of backprojecting onto sub-images using a proportionally smaller number of projections. If the image is, for example, separated into four quadrants of size $N/2 \times N/2$, then they can be backprojected using only $P/2$ available projections. Assuming that $N = 2^n$, the algorithm can be recursed n times to reach the base case with only one remaining pixel ($B = 1$). This leads to an algorithm with $\mathcal{O}(N^2 \log N)$ arithmetic cost. Anti-aliasing with a filter $\psi(\theta)$ is applied after angular downsampling to reduce the distortion. However, some aliasing occurs at each step of the recursion and the effect largely depends on the spectral properties of the image. To investigate this cost/distortion tradeoff, we design a flexible recursion strategy (see schematics in Figure 1) described by the following pseudo-code:

Algorithm 1 (Parameterized HBP Algorithm)

```
BACKPROJECTION ($N, \tilde{f}(\rho,\theta)$)
    if $N \leq B$ then
        return DIRECT BACKPROJECTION($N, \tilde{f}(\rho,\theta)$)
    else
        for $i = 1 \ldots 4$ do
            if $N \leq D$ then
                $\tilde{f}(\rho,\theta) \leftarrow$ DOWNSAMPLE ($\tilde{f}(\rho,\theta), \phi(\rho), \psi(\theta)$)
            endif
            $\tilde{f}_i(\rho,\theta)) \leftarrow$ SEGMENT($\tilde{f}(\rho,\theta)), i$)
            $b_i(x,y) \leftarrow$ BACKPROJECTION ($N/2, \tilde{f}_i(\rho,\theta)$)
        endfor
        return TILE BLOCKS($b_0(x,y), \ldots, b_3(x,y)$)
    endif
```

First, the recursion simply segments the original image sinogram into sub-sinograms corresponding to image blocks of size D without downsampling. This is really a form of angular oversampling suggested in [3] to further decrease the distortion. From that point on, the angular smoothing and downsampling is applied recursively until a base case block size B is reached. At that moment, the direct backprojection is applied to all sub-sinograms, and the resulting image blocks are tiled to form the reconstructed image. For example, for $D = B = N$, the algorithm is equivalent to the direct BP, and for $D/B = 1$ no cost savings occur.

Savings in arithmetic cost occur only for sub-image sizes in the range between D and B (dotted line in Figure 1), where the projections are downsampled. We vary both D and B to examine the effects on performance.

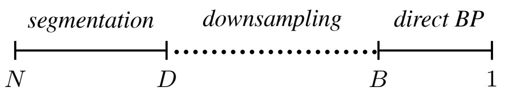

Fig. 1. Recursion strategy for the HBP algorithm; cost savings occur in the dotted range.

Cost of direct and hierarchical backprojection. The number of collected projections P is typically chosen to be proportional to the image size $P \sim N$. The direct backprojection operation dominates the cost since it requires a summation of P values for all N^2 pixels and is hence $\mathcal{O}(N^3)$, where we assume an interpolation cost of $\mathcal{O}(1)$. The cost of the direct BP is

$$\mathbf{C}^{BP}(N,P,\phi) = \big(10 + c(\phi)\big)N^2P + N^2 + 4. \qquad (5)$$

The constant $c(\phi)$ depends on the chosen radial interpolator ϕ (e.g., $c(\phi) = 4$ for linear and $c(\phi) = 10$ for cubic spline).

The exact arithmetic cost (counting additions, multiplications, absolute value, and rounding equally) of our parameterized HBP algorithm is given by

$$\begin{aligned}\mathbf{C}^{HBP}&(N,P,D,B,\phi,\psi) = \\ &c_1(\phi,\psi)P\lceil\sqrt{2}N\rceil \tfrac{D}{N}\log_2\left(\tfrac{D}{B}\right) + c_2(\phi)N^2P\tfrac{B}{D} \\ &+ c_3P\left(\tfrac{N}{D}\right)^2 + c_4\left(\tfrac{D}{B}\right)^2 + c_4P\tfrac{D}{B} + N^2 + \mathcal{O}(N) \qquad (6)\end{aligned}$$

In the limit case when $D = N$ and $B = 1$, the cost approaches $\mathcal{O}(N^2\log N)$ or, precisely,

$$\mathbf{C}^{HBP} = c_1(\phi,\psi)PN\log_2 N + \mathcal{O}(N^2). \qquad (7)$$

The constants c_1 and c_2 depend on the choice of the interpolator and the anti-aliasing filter $\psi(\theta)$. We choose $\psi(\theta) = [0.5\ 1\ 0.5]$ for all experiments. Then, for example, $c_1 = 40,\ c_2 = 20$ for the linear and $c_1 = 88,\ c_2 = 46$ for the cubic spline interpolator. Note that in (6) the second term dominates the cost when the window size D/B is small and the third term dominates when D is chosen small. In both cases the cost approaches $\mathcal{O}(N^3)$.

3. IMPLEMENTATION AND OPTIMIZATION

This section describes our C implementation of algorithm 1. Our goal was fast and portable code that enables investigation of performance tradeoffs as well as implementation choices.

System Design. Our system implements algorithm 1 by the following functions: 1) BACKPROJECTION() is the main recursive function. Depending on the current state of the recursion and the parameters B and D, it applies the recursive algorithm with or without downsampling or calls the direct method; 2) DIRECT-BACKPROJECTION() implements the direct BP algorithm; 3) The functions DOWNSAMPLE(), TILEBLOCKS() and SEGMENT() are helper functions and a part of BACKPROJECTION(); The functions $\phi(\rho)$ and $\psi(\theta)$ can be runtime or compile-time parameters trading speed versus generality. Allowing for automated investigation of program configurations and algorithm parameterizations, we make heavy use of the C preprocessor. To avoid data copying in TILE-BLOCKS() and SEGMENT() we used dynamic memory allocation involving pointer arithmetic instead of 2D arrays and sub-arrays.

Optimization. To facilitate debugging and verification, we first implemented algorithm 1 in a straight-forward manner. We then restructured our implementation to optimize for speed. Optimizing compilers featuring inter-procedural optimization support

Optimization Methods	Runtime
Basic implementation	41.20 s
+ Loop invariant code motion and loop fusion	0.64 s
+ Precomputation of constants	0.23 s
+ Function inlining and further loop fusion	0.08 s

Table 1. Impact of optimization methods on the runtime for $N = P = 128$. The fastest version applying all optimization methods is running at 15 % of the machine's peak performance.

most of the optimization techniques we applied. However, the compilers often underachieve when optimizing codes featuring pointer arithmetic so we resort to hand-optimization. In addition we applied some problem-specific optimizations. For a problem size of 128 × 128 pixels and 128 projections, our optimized implementation runs at approximately 15 % of the system's peak performance and is 500 times faster than our basic implementation. Table 1 summarizes the impact of different optimization steps that are detailed below.

Function Inlining. When the functions computing $\phi(\rho)$ and $\psi(\theta)$ are compile-time parameters, we inline them to avoid the function call overhead. In addition, we make sure that mathematical library functions are inlined when supported by the compiler.

Loop Invariant Code Motion. The functions DOWNSAMPLE() and DIRECTBACKPROJECTION() contain loops which compute expressions with loop-invariant sub-expressions. These sub-expressions are moved outside of the loops.

Loop Fusion. We fused the functions DOWNSAMPLE() with SEGMENT() and TILEBLOCKS() with BACKPROJECTION() leading to multiple loops operating on the same data set. In addition, the function DOWNSAMPLE() contains application of both $\phi(\rho)$ and $\psi(\theta)$ translating into a sequence of loops over the same data set. To optimize for locality, we reordered the computation and fused these sequences of loops leading to a small number of loops with more compute-intensive loop bodies.

Precomputation of Constants. We precompute the values of $\sin\theta$ and $\cos\theta$ for all angles θ and store the results in an array. This replaces an evaluation of a transcendental function by a memory access.

4. RESULTS

We evaluated our implementation of algorithm 1 for runtime and distortion performance as a function of several parameters.

Platform. We performed all experiments on a 3.2 GHz Intel Pentium 4 platform with 8 kB L1, 512 kB L2 data cache, and 1GB DDR RAM. The system was running Linux Fedora Core 2 and we were using the Intel C++ compiler 8.0 with compiler flags "-fast" which was experimentally determined to be the best choice.

Performance metrics. As the runtime performance measure we use the execution time of the compiled C code as well as the number of floating point operations per second (FLOPS). To measure the distortion, we use both visual and quantitative measures. For the latter we choose the root mean square error (RMSE) of the reconstructed image $\bar{f}(m, n)$ relative to the original $f(m, n)$:

$$\epsilon_{\text{RMSE}} = \sqrt{\|\bar{f} - f\|_F / \|f\|_F}, \qquad (8)$$

where $\|\cdot\|_F$ is the Frobenius norm.

Experimental Setup. All experiments are performed on images of size 512 × 512 pixels with $P = 1024$ projections. We evaluate different algorithm strategies by varying both D and B, and by choosing our radial interpolator $\phi(\rho)$ as either the nearest neighbor, linear, cubic, or cubic spline. In our experiments we use four images with different spectral characteristics: 1) The standard Shepp-Logan phantom head image, 2) "peppers", a smooth picture featuring soft shapes, 3) "harbor", a highly detailed image with substantial energy in the high frequency range, and 4) "Gaussian", an image created of several 2D Gaussian functions with different means and variances. Both the ramp filtering and the forward 2D RT are performed using MATLAB, where for the RT we use MATLAB's `radon` function.

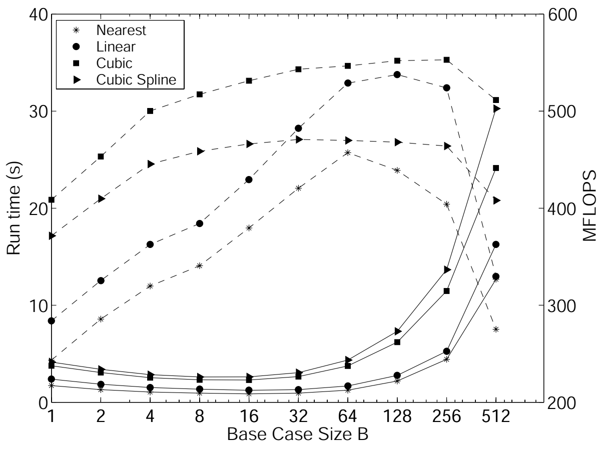

Fig. 2. Runtime (contiguous line) and FLOPS (dashed line) for various base case sizes and interpolators.

Runtime performance experiments. In the first experiment we analyze the runtime behavior of algorithm 1 without oversampling ($D = N$) by varying the parameter B from the fully recursive $\mathcal{O}(N^2 \log N)$ case where ($B = 1$) to the direct BP method ($B = N$). The results in Fig. 2 show that the runtimes using various interpolators rise proportionally with their cost, i.e., cubic spline is slowest while nearest neighbor is fastest. On the other hand, the cost savings arising from the recursion and downsampling (see Figure 1) do not always provide the fastest turnaround time since more recursion steps mean more complicated data access patterns. The full recursion is 50 % slower than the optimal runtime obtained for $B = 16$. The algorithms perform at 10 % to 15 % of the peak performance and show best performance for medium to large base case sizes where data access patterns and the amount of arithmetic operations are best balanced. For the direct method the performance drops significantly due to cache effects.

In the second experiment, we include the angular oversampling controlled by the size D as we explained in Section 2. Figure 3 (top) shows the runtime plots for all combinations of B and D, where we use the cubic spline interpolator. Different lines correspond to different window sizes (D/B) (see Figure 1) whereas the x-axis determines the window position B. For the maximum execution speedup, it is clearly advantageous to use B as large as possible for a fixed window length. It remains to explore how these choices affect the reconstruction accuracy, which we do next.

Distortion analysis experiments. Using the same setup as for the runtime experiments, we create error plots for the phantom head and the harbor image, shown below the runtime plot in Figure 3. For the phantom head, a smaller base case size always gives smaller error for the same window length. However, we might choose to trade accuracy for speed or vice versa. For

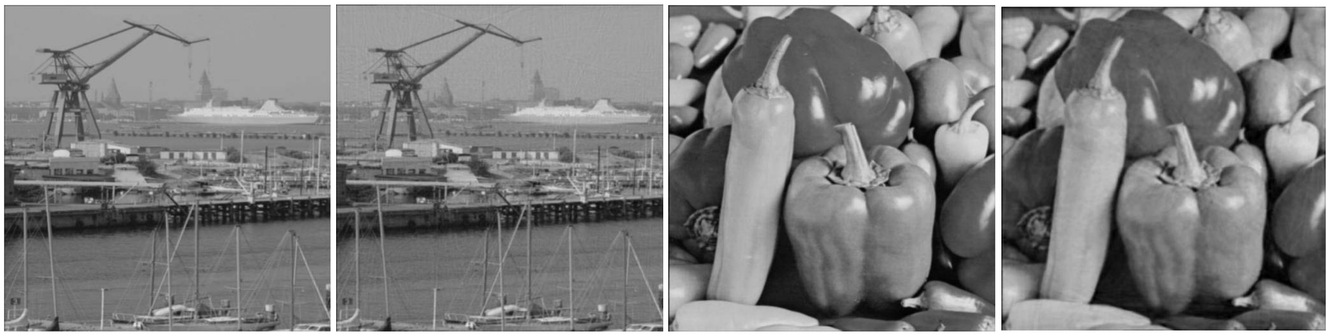

Fig. 4. Harbor image for $B = N$ (left) and $B = 32, D = 128$ (left center); Peppers image for $B = N$ (right center) and $B = 16$ (right).

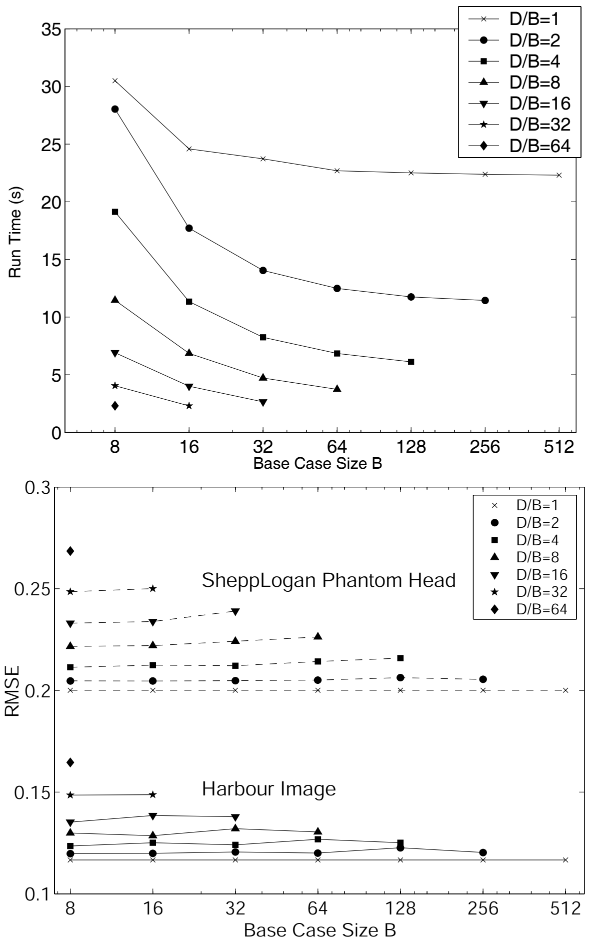

Fig. 3. Runtimes (top) and RMSE (bottom) for different recursion strategies.

example, choosing $B = 64, D = 256$ is slightly less accurate than $B = 64, D = 128$, with almost 40% speedup. In the case of the harbor image, given the fixed window size D/B, it is not clear which choice of B gives best performance. We note that for both images the speedup gains are more prominent than the variation of distortion for the fixed window size, which suggests that varying B is more effective for achieving the desired result than choosing a different D.

Image	Nearest	Linear	Cubic	Cubic Spline
Gaussians (M)	0.06772	0.06731	0.06716	0.06701
Gaussians (A)	0.02939	0.02703	0.02696	0.02697

Table 2. RMSE of the Gaussian image reconstructed by FBP.

At this point we note that the computed RMSE includes errors introduced by the forward RT as well as the ramp filtering, both performed in MATLAB. In Table 4 we show errors of the reconstruction using the direct BP algorithm for the Gaussian image, where the sinogram is obtained: (M) using MATLAB; (A) analytically to avoid forward RT error. For this particular image, the forward RT error contributes for about 60 % of the total error.

Assessment of visual quality. Figure 4 shows the reconstructed images using our implementation of the HBP algorithm. We compare the result for the harbor image using the direct BP and the HBP algorithm with $B = 32$, $D = 28$. The distortion is noticeable only under closer inspection while the runtime improved by about 62%. In the case of the peppers image, even the fastest implementation $(B = 16)$ gives only slightly blurred reconstructed image as shown in Figure 4 (right). The speedup in this case is about 10 times, from 22.3 s for the direct BP to 2.3 s for the fastest HBP algorithm.

5. CONCLUSION

We parameterized the hierarchical backprojection algorithm with several algorithmic choices and optimized the C implementation for best performance. We presented an extensive runtime and distortion evaluation to investigate interesting tradeoffs and provided a detailed cost analysis of the algorithm. The results show that, given the desired performance, the parameter choice depends on image properties and features of the target computer platform.

6. REFERENCES

[1] F. Natterer, *The Mathematics of Computerized Tomography*. New York: Wiley, 1986.

[2] D. C. Munson, J. D. O'Brien, and J. W. K., "A tomographic formulation of spotlight-mode synthetic aperture radar," *Proceedings of the IEEE*, vol. 71, pp. 917–925, August 1983.

[3] S. Basu and Y. Bresler, "$O(N^2 \log_2 N)$ filtered backprojection reconstruction algorithm for tomography," *IEEE Trans. on Image Processing*, vol. 9, pp. 1760–1772, October 2000.

[4] P. A. Rattey and L. A. G., "Sampling the 2-d radon transform," *IEEE Trans. on Acoustics, Speech, and Signal Proc.*, vol. ASSP-29, pp. 994–1002, October 1981.