Tomographic reconstruction algorithms
Implemented with SciPy by Tim Day

Objective

- This programme arose out of a desire to better understand more of the visualisation pipeline (figure 1), and the broader context within which volume rendering exists.
- The reconstruction stage in particular, in which raw scanner sinograms are processed to more familiar volume data, seemed deeply mysterious. By learning enough to actually implement a variety of reconstruction algorithms, I hoped to gain some insight into this area.
- All the results shown are obtained from my own implementations using SciPy (a MATLAB-like toolkit based on Python). Execution times reported are from a quad-core (hyperthreaded) 2.67GHz Intel i7.

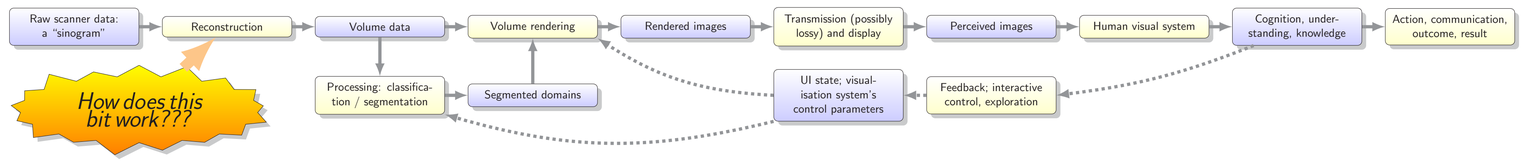


Figure 1: The visualisation pipeline. Processing steps are yellow, blue represents data/artefacts/state.

Simulated scanning and the Radon transform

For the purposes of obtaining some test data, parallel beam imaging is simulated (see figure 2). This keeps the mathematics as simple as possible without adding the considerable complications associated with real scanners' fan beam geometry (see figure 3) which arises due to their point x-ray sources illuminating a line array of sensors.

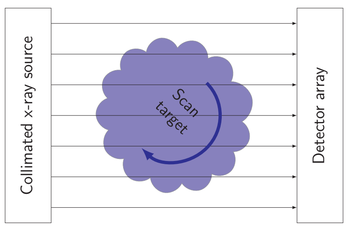


Figure 2: Parallel beam scan geometry.

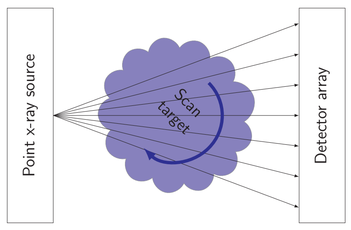


Figure 3: Fan beam scan geometry.

The scanner model used projects x-rays along the x-axis through a target. The attenuated signal is detected by a row of sensors parallel to the y-axis on the far side of the target. Repeated measurements ("exposures") are made with the target rotated by various angles relative to the measurement system.

With the target T rotated to angle θ, the measurement by detector y is the original ray strength attenuated by a factor $e^{-\sigma(\mathcal{R}T)(y,\theta)}$ where $\mathcal{R}(y,\theta)$ is the Radon transform operator (named for the Austrian mathematician Johann Radon who first studied its properties in 1917) and σ is a constant describing the rate at which target density attenuates the signal.

The Radon transform (in 2 dimensions) simply describes line integrals through the rotated target:

$$(\mathcal{R}T)(y,\theta) = \int_{-\infty}^{\infty} T(x\cos(\theta) - y\sin(\theta), x\sin(\theta) + y\cos(\theta))dx$$

The 256×256 pixel "phantom" target used in these experiments, and a resulting simulated 359×256 pixel sinogram are shown in figure 4. Regions of high value (high density) in the target image correspond to dark, attenuated signals in the sinogram. Really the negative-exponential term simply serves to map the indeterminate range of accumulated density to a more manageable $[0 \ldots 1]$ range for saving to a sinogram image. All the reconstruction algorithm implementations here simply invert this immediately on loading such an image to recover the measured Radon transform of the target and work with $S(y,\theta) = (\mathcal{R}T)(y,\theta)$.

There must be some concern that quantisation to a limited number of gray-levels in the saved sinogram image introduces some error (particularly if a bad choice of attenuation coefficient is made, limiting the dynamic range in the sinogram). However, repeating the experiments with a 16-bit image representation of the sinogram does not provide any evidence that this is a problem in any results shown here.

An odd number of angles are used because $\mathcal{R}(y,\theta+\pi) \equiv \mathcal{R}(y,\theta)$ and so no "new information" would be gained from directly opposite scan exposures.

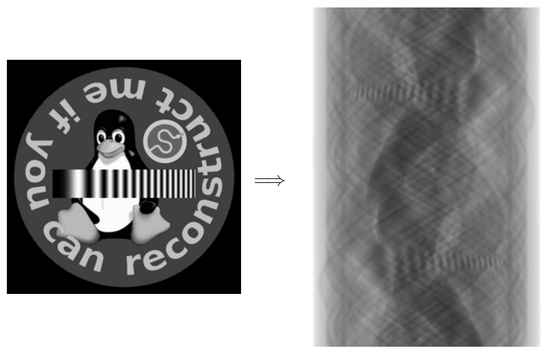


Figure 4: Simulated scan target and resulting sinogram. Rows of the sinogram image correspond to varying θ.

Simulated sinogram images are trivially created with SciPy by rotating the target image with `scipy.ndimage.interpolation.rotate` and applying `np.sum` over one axis.

FBP: Filtered Back-Projection

One approach to recovering the target from a sinogram is to back-project the observations. Bearing in mind that each point in the sinogram corresponds to a unique ray through the target area, the back-projection at a point is defined as the average of all the rays intersecting that point:

$$(\mathcal{B}S)(x,y) = (\mathcal{B}\mathcal{R}T)(x,y) = \frac{1}{2\pi}\int_{-\pi}^{\pi}(\mathcal{R}T)(x\sin(\theta) + y\cos(\theta),\theta)d\theta$$

Applying this to my simulated sinogram yields the image in figure 6; some evidence of the target is (just!) visible. The blurriness suggests that with some suitable filtering a better image might be recovered, and in fact by application of some mathematics involving the central slice theorem (also known as the Fourier slice theorem or projection slice theorem; published in 1956 by Robert Bracewell working in the field of radio astronomy) the filtered back projection formula can be derived:

$$T(x,y) = \frac{1}{2}\mathcal{B}\left\{\mathcal{F}^{-1}\{|s|(\mathcal{F}\mathcal{R}T)(s,\theta)\}\right\}(x,y)$$

where $\mathcal{F}$ represents a Fourier transform. Intuitively, the application of $|s|$ in the Fourier domain represents a boosting of the higher frequencies. However the higher frequencies are also responsible for much of the noise in the reconstructed image and it can be useful to also suppress these.

In the discrete implementation, we use the Ram-Lak (Ramachandran and Lakshminarayanan) filter; this has a parameter $\epsilon \in [0 \ldots 1]$ which allows some control of the frequency response (see figure 5). Using these filters the reconstructions in figures 7, 8 and 9 are obtained.

The influence on image quality of varying the number of sinogram angles is show in figure 10.

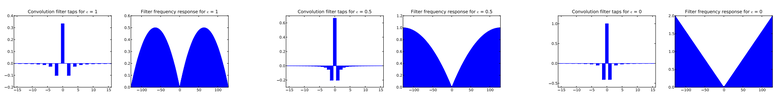


Figure 5: Ram-Lak filter convolution coefficients and frequency responses for various ϵ

Figure 6: Unfiltered back projection.

Figure 7: Filtered back projection with $\epsilon = 1.0$

Figure 8: Filtered back projection with $\epsilon = 0.5$

Figure 9: Filtered back projection with $\epsilon = 0.0$

Figure 10: Increasing reconstruction quality with increasing number of sinogram angles (1,2,3,5,8,13,21,34,55,89,144)

The core of my implementation consists of a back-projector which can filter a sinogram exposure and project it across the reconstruction image. Filtering is applied in frequency space. Parallelism is achieved by map-reduce over sinogram exposure angles. Reconstruction requires only a few seconds.

Key numpy/SciPy functions used are `np.meshgrid`, `np.fft.ifft`, `np.fft.fft`, `scipy.interpolate.interp1d`.

Fourier reconstruction method

The central slice theorem basically states that the (1D) Fourier transform of the Radon transform (at some given angle) of a target image is equal to the values along a line (a slice) at that same angle passing through the centre of the 2D Fourier transform of the target image.

This immediately suggests the following reconstruction algorithm:

- Fourier transform (1D) the data for each of the sinogram angles; example shown in figure 11.
- Arrange them appropriately in 2D Fourier space, where they form a radial "spokes of a wheel" pattern.
- Interpolate from that polar pattern to a regular Cartesian grid over the 2D Fourier space. Figure 12 shows the result of this for the test sinogram.
- Reconstruct the target image by inverse 2D FFT of the interpolated Fourier space. Results are shown in figure 13).

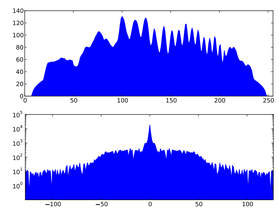


Figure 11: Top: sinogram data for one exposure angle. Bottom: Corresponding FFT (magnitude).

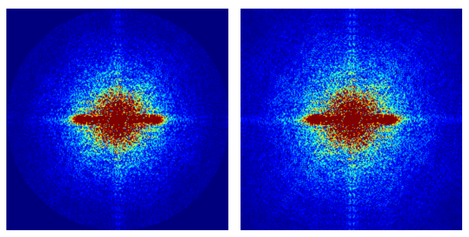


Figure 12: Left: Fourier space interpolated from Fourier-transformed sinogram exposures. Right: FFT of the original target image, shown for comparison.

Incidentally, this gives some insight into the FBP formula's $|s|$ term, which can be seen as a factor compensating for the increased density of contributing samples in the central low-frequency region of the 2D Fourier space. The Fourier method itself avoids any need for such compensation by simply using the additional samples to obtain higher accuracy interpolation of the low-frequency coefficients.

The quality of the results obtained can be improved by increasing the resolution of the 2D FFT grid, as shown in figure 13. However, because of the increased computational demands of this the method is not generally considered competitive with FBP and is consequently little used.

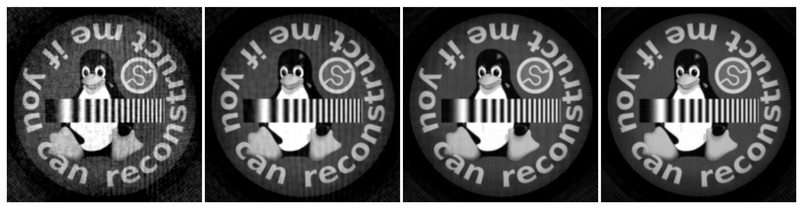


Figure 13: Images obtained by Fourier method with increasing resolution of interpolated Fourier space ($\times 1$, $\times 2$, $\times 3$, $\times 4$)

My implementation uses various 1D & 2D FFT functions from `scipy.fftpack`. The interpolation is performed by the convenient (but far more general than is actually necessary here) `scipy.interpolate.griddata`, which can interpolate arbitrary points to a mesh. Reconstruction only requires a few seconds.

Algebraic Reconstruction Technique (ART)

ART was the reconstruction method used by Hounsfield in the first commercial scanner (EMI, 1972).

Derivation of the algebraic family of reconstruction algorithms begins with the observation that a vector of the unknown target's pixel values $T_i \equiv T(x_i, y_i)$, are related to the vector of measurements $S_i \equiv S(y_i, \theta_i)$ in the linear relation

$$S_i = \sum_j P_{ij}T_j \qquad \text{(or simply } S = PT\text{)}$$

Rows of P describe how pixel values in the target contribute to a specific sinogram pixel. Since sinogram pixels correspond to a single ray through the target (missing most of the target's pixels), P is extremely sparse; further, P it depends only on the scanner geometry and can be computed a-priori of any actual scanning.

ART attempts to solve this system of linear equations using a technique using Kaczmarz method. Given an initial estimate of the target $T^{(0)}$ (typically all zeros) this produces a series of updated estimates

$$T^{(k+1)} = T^{(k)} + \lambda\frac{S_i - \sum_j P_{ij}T_j^{(k)}}{|P_i|^2}P_i$$

where $i = k \mod n$ where n is the number of sinogram pixels, and λ is a relaxation parameter. Note that the sparseness of P contributes greatly to computational efficiency.

This can be understood as, for each iteration (which corresponds to considering a particular sinogram pixel), simply comparing the observation predicted by the estimated T with that actually measured, and applying a proportionate correction to the elements of T influencing that result. Consult the sources mentioned below for an analysis of why this actually works and converges to a useful result.

Results from applying this algorithm to the synthetic sinogram of figure 4 are shown in figure 14; the images show the estimate of the target at 5120 iteration intervals, corresponding to each 20 rows of the sinogram processed.

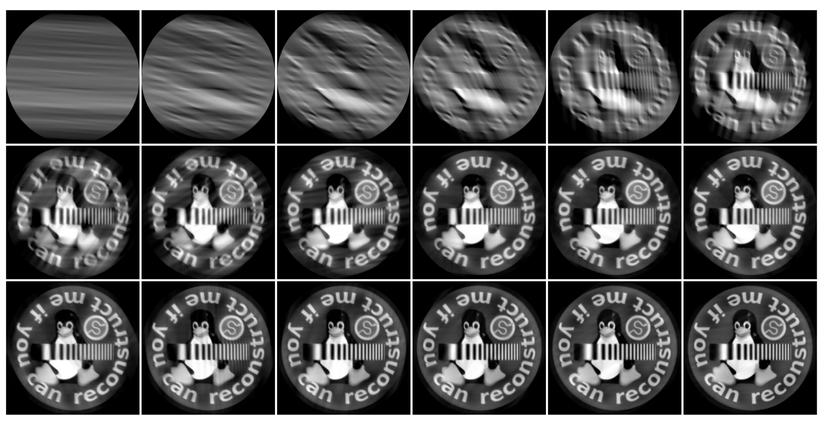


Figure 14: ART progress after an increasing number of complete sinogram exposures have been processed (one image every 20 sinogram angles)

Unsatisfactory aspects of ART are that the result is dependent on the order in which the sinogram pixels are processed: whichever are handled most recently will "overwrite" the contribution of earlier iterations. Work-rounds (untried) include multiple passes, processing in random order and decreasing the λ weighting with successive iterations. More useful aspects of this approach (vs. FBP) are that it can be modified to handle peculiar or incomplete scanning geometries.

Pre-construction of the necessary set of P_i takes a few minutes (parallelised) to create and serialise the sparse projection matrices to a 0.5GByte file. (This calculation would be significantly faster if SciPy's image rotation functions inter-operated with sparse matrices.)

The saved P_i can then be loaded by the reconstruction implementation in a few seconds; reconstruction itself takes around 100 seconds to complete a pass over all sinogram pixels (yielding figure 15).

The reconstruction calculation makes good use of SciPy's sparse matrices, but the implementation has no parallelism due to the tight coupling between $T^{(k+1)}$ and the preceding $T^{(k)}$.

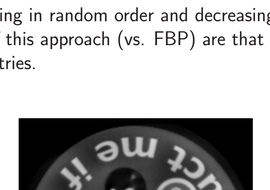


Figure 15: ART result after one complete pass over the sinogram

SIRT & SART -like simultaneous iterative techniques

The fundamental idea behind a broad class of algorithms including Simultaneous Iterative Reconstructive Technique (SIRT) & Simultaneous Algebraic Reconstruction Technique (SART) is to calculate (similarly to ART) corrections to an estimated $T^{(k)}$ due to *all* the S_i independently, and then compute an updated $T^{(k+1)}$ by applying the average of all those corrections. Obviously this eliminates ART's issues with S_i processing order influencing the result. Figures 16 and 17 show results obtained by this method after increasing numbers of iterations.

As originally published, SART's main innovation over SIRT was to add better modelling of the forward projection process allowing fractional contributions of target elements (however, even my ART implementation already included this) and the use of a "heuristic window function" to make corrections near the centre of a ray more strongly than at the ends (not implemented here).

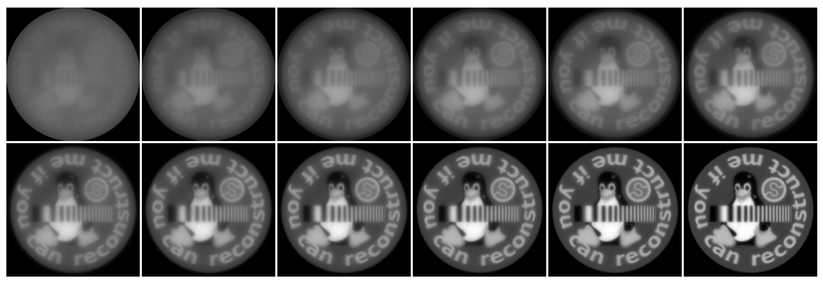


Figure 16: Estimated target image after an increasing number of iterations (1, 2, 3, 4, 5, 7, 10, 14, 19, 26, 36, 50)

My ART code was easily adapted to compute updates to the estimated target image independently, according to the ideas of SIRT/SART.

Since the corrections are calculated independently, the algorithm also becomes trivially parallelisable by map-reduce.

Execution time is approximately 15 seconds per iteration over all sinogram pixels.

Figure 17: Iterative result after 100 iterations

Direct solution of the linear system

It is possible to apply standard linear algebra techniques to the problem of finding the unknown T in $S = PT$. Assuming the system is over-determined, this can be converted to a least-squares problem by multiplying both sides on the left by P^T to obtain $P^TS = P^TPT$ (this reduces the matrix component to a square, and the system becomes a solvable set of simultaneous linear equations, the solution of which minimises $\|S - PT\|^2$). This is a formidable calculation but just feasible for the 256×256 test case used here (see box for details). Figure 18 shows the rather unsatisfactory result obtained by the obvious approach.

To prefer smooth solutions, regularization can be used. A Tikhonov matrix Γ is introduced in $P^TS = (P^TP + \Gamma^T\Gamma)T$ and the solution of this minimises $\|S - PT\|^2 + \|\Gamma T\|^2$. If Γ is simply a scaling of the identity matrix, then solutions which minimise the norm of T are favoured. However Γ can also be constructed to minimise the sum of the squares of differences between adjacent pixels of T and thereby favour smooth solutions. The effect of varying the weight placed on such smoothing is shown in figure 19.

Figure 18: Solution without regularization.

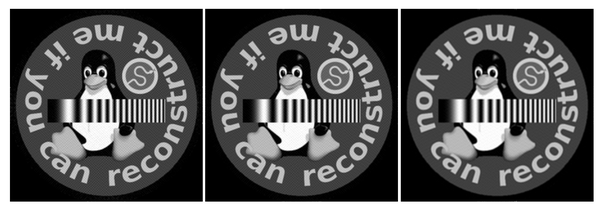


Figure 19: Regularized solution (increasing weighting for smoothness; $\Gamma \propto 1, 4, 16$).

Direct solutions were obtained using `scipy.linalg.solve`; use of double precision was essential. This algorithm is $O(N^3)$ in the number of unknowns (and the number of unknowns in T quadruples when image dimensions are doubled), so while a small 64×64 test case ran in 15s in a small memory footprint, 128×128 required 15 minutes with several gigabytes of RAM and 256×256 required almost 21 hours with over 50GByte RAM (which should make it clear why this method is generally considered to be of no practical value when much more efficient algorithms exist).

An Amazon EC2 "High-Memory Quadruple Extra Large Instance" was used (see figure 20) for the 256×256 solutions to obtain the results shown in figures 19 and figures 18.

Sparse matrices are not very useful as P^TP is fairly dense, and even in the 64-bit version of SciPy they do not support more than 2^{31} non-zero elements.

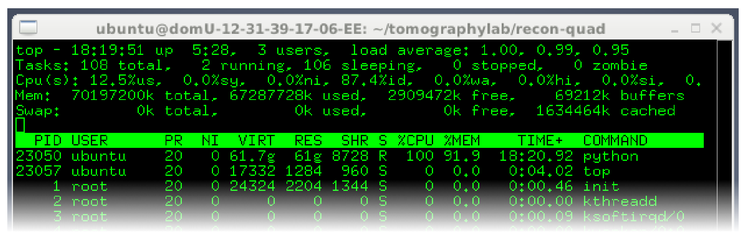


Figure 20: An Amazon EC2 "m2.4xlarge" instance calculating a "direct" solution.

Conclusions, and possibilities for further investigation

An interesting and worthwhile exercise which has considerably de-mystified the reconstruction process. Actually also attempting a "direct" linear-algebra solution gives some appreciation of how ingenious and efficient the other algorithms are.

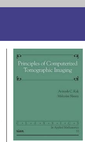
Many more reconstruction algorithms remain to be investigated: linograms, layergrams, MART (Multiplicative Algebraic Reconstruction Technique) and its simultaneous SMART form, plus a large family of reconstruction algorithms derived from statistical considerations such as various EM (Expectation Maximisation) algorithms which are actually closely related to SMART.

It would be interesting to test further how implemented algorithms deal with difficulties such as reducing sinogram resolution (as in figure 10), or adding simulated metal artefacts, noise, scatter and beam hardening.

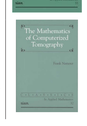
Books and tools

The Mathematics of Medical Imaging: excellent recent undergraduate-level text. Covers FBP and ART (only) in detail, including all the necessary background theory from first principles. Also includes a chapter on MRI imaging. Highly recommended.

Fundamentals of Computerized Tomography: Image Reconstruction from Projections: The biggest and most explicitly medical imaging oriented book. Mathematics is well covered, although with a fair amount of "it can be shown that...".



Principles of Computerized Tomographic Imaging: Available online at `http://www.slaney.org/pct/`. Treatment of algebraic methods extends to SIRT and SART. Includes coverage of such exotica as diffracting and reflected tomography.



The Mathematics of Computerized Tomography: covers FBP, ART and the Fourier method. Very rigorous treatment of the mathematics, including analysis of errors and convergence. Not recommended as a first introduction to this field!

Created on a Debian Gnu/Linux system using Python, SciPy, Matplotlib and LaTeX (beamerposter and TikZ/PGF).


For more information see `http://bitbucket.org/timday/tomographylab` or contact `timday@timday.com`

"It's amazing what you can get by the ability to reason things out by conventional methods, getting down to the basics of what is happening." - Sir Godfrey Hounsfield