3D CT Cone Beam Reconstruction Via Fan to Parallel Rebinning

Gregory Handy

---

1. Overview

This semester (Fall 2012), I spent time reading and implementing the paper by Grass et al. titled "3D cone-beam CT reconstruction for circular trajectories". This paper presented a new reconstruction algorithm that modifies the well-known Feldkamp algorithm. Their new method can be broken into two major steps. The first step rebins the standard 3D cone beam geometry into a parallel-beam geometry along the source trajectory. It is possible to perform a filtered backprojection of the data at this step, and this method is known as P-FDK (parallel-FDK). Despite the algorithmic differences with FDK, this method does not differ in image quality/artifacts. The second step interpolates the projections onto a rectangular virtual detector plane and then proceeds with the filtered backprojection. This method is known as T-FDK (tent-FDK) and shows improved image quality.

In this paper I will provide a mathematical formalism of the algorithm, which is not found in the Grass et al. paper, along with directions on how to use the corresponding code. The formulas will be consistent with the variables used in the book chapters written by Fessler, and will reference sections out of these chapters, code found in the irt toolbox, as well as new code. The P-FDK and T-FDK methods have been completely implemented, but both should be tested on additional phantoms and real datasets.

2. P-FDK

Assume that we are given cone-beam projection data $p^f(s, t, \beta)$, where f denotes the cone/fan-beam geometry, s is the column position on the detector, t is the row position on the detector, and β is the angular position of the x-ray source. Rebinning to parallel fixes t, and rebins in the s direction. For each fixed t, we are in the geometry presented in Section 3.9.

2.1. Fan to Parallel Rebinning

In Section 3.9.1, fan to parallel rebinning is defined assuming $r_{\text{off}} = 0$. We will also assume that we are using third-generation X-ray CT systems ($D_{\text{fs}} = 0$). This simplifies expression (3.9.11) to

$$p^p(r, \phi) = p^f(D_{sd} \arcsin(r/D_{s0}), \phi - \arcsin(r/D_{s0})),$$

where ϕ denotes the angle between the ray and the y-axis, r is the distance from the center of rotation to the ray, D_{s0} is the distance from the source to the center of rotation, and D_{sd} is the distance from the source to the detector. This geometry can be seen in Figure 1. Extending this to our 3D case yields,

$$p^p(r, t, \phi) = p^f(D_{sd} \arcsin(r/D_{s0}), t, \phi - \arcsin(r/D_{s0})).$$

The code implementing this step simply loops over the t-coordinate and makes repeated calls to the function rebin_fan2par. It should be noted that this function requires a 2D projection matrix and a 2D CT geometry. Both of these can be created using the 3D CT geometry and 3D projection matrix provided. The user has control over the resulting parallel beam geometry (can change orbit, ds, etc.). These changes may offer a slight improvement in image quality. When decreasing ds to make a finer grid, the only requirement is that ds·ns remains constant.

After rebinning, one must define a new 3D CT geometry, and make any necessary changes (new ds, etc.). When doing so, the user should use the function ct_geom_par. This is essentially the same as ct_geom, but allows for one additional variable, called 'rebinned', which should now be set to 1. In the future, the ct_geom code should be rewritten to allow for a new type, called par_rebinned, but for now, the modified code checks if cg.rebinned == 1. This small change allows for our 3D CT geometry to use variables such as D_{sd}, which can only be set to 'inf' for the parallel type currently.

2.2. New Geometry

In order to complete the reconstruction algorithm, the new geometry must be understood. Figure 1 offers an above view of the rebinnned projection data while Figure 2 offers a side view. The distances in these figures will be referenced in the following sections.

It is important to note that the total distance from the source to the detector (D_{sd}) is kept constant. As a result, the 'new' detector has an inverted

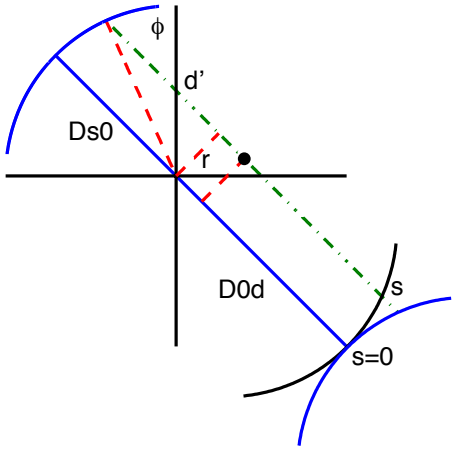

Figure 1: New geometry after rebinning. View is from above, and the x-y axis is provided. Note that the total distance D_{sd} is preserved for the rebinned data, which causes the new detector to curve outwards. Point of interest is the black dot.

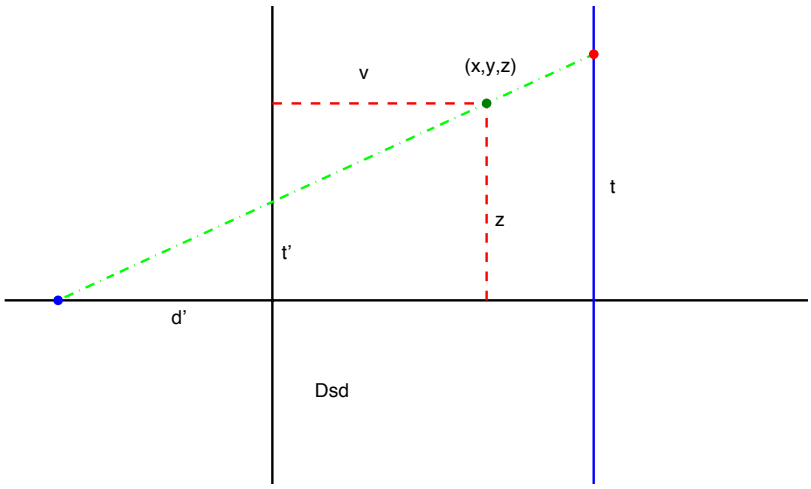

Figure 2: The side view of the projection ray through the point of interest for the new geometry. The v-z axis is provided. The total horizontal distance from the source to the detector is D_{sd}.

arc. Figure 2 is given relative to the v-z axis, where $v(x, y, \phi) = y_\phi$, which can be found in Equation (3.9.16), and is reproduced below:

$$v(x, y, \phi) = (-x \sin(\phi) + y \cos(\phi)).$$

2.3. Filtered Backprojection

The pre-weighting and filtering step proceed as normal with

$$\begin{aligned} w^p(t) &= \frac{D_{\text{sd}}}{\sqrt{D_{\text{sd}}^2 + t^2}} \\ \hat{p}^p(r, t, \phi) &= \Big(w^p(t) p^p(r, t, \phi)\Big) * g^p(r). \end{aligned}$$

The cosine weighting function can be derived from Figure 2, and $g^p(r)$ is the usual ramp filter used in the parallel-beam case (Equation (3.4.14)).

The backprojection now follows with

$$f_p(x, y, z) = \int_0^{2\pi} \hat{p}^p(r(x, y, \phi), t(x, y, z), \phi) d\phi,$$

where

$$\begin{aligned} r(x, y, \phi) &= x \cos(\phi) + y \sin(\phi) \\ t(x, y, z, \phi) &= \frac{D_{\text{sd}} z}{\sqrt{D_{\text{s0}}^2 - r(x, y, \phi)^2} + v(x, y, \phi)}. \end{aligned}$$

The formula for $r(x, y, \phi)$ follows directly from the parallel beam geometry. In order to derive $t(x, y, z, \phi)$, note that by the similar triangles seen in Figure 2, we have

$$\frac{t}{D_{\text{sd}}} = \frac{z}{d' + v},$$

where

$$d' = \sqrt{D_{\text{s0}}^2 - r^2},$$

can be derived from Figure 1. It is interesting to note that additional weighting is not required during the backprojection step. Figure 3 shows the reconstruction results using the P-FDK and FDK algorithms.

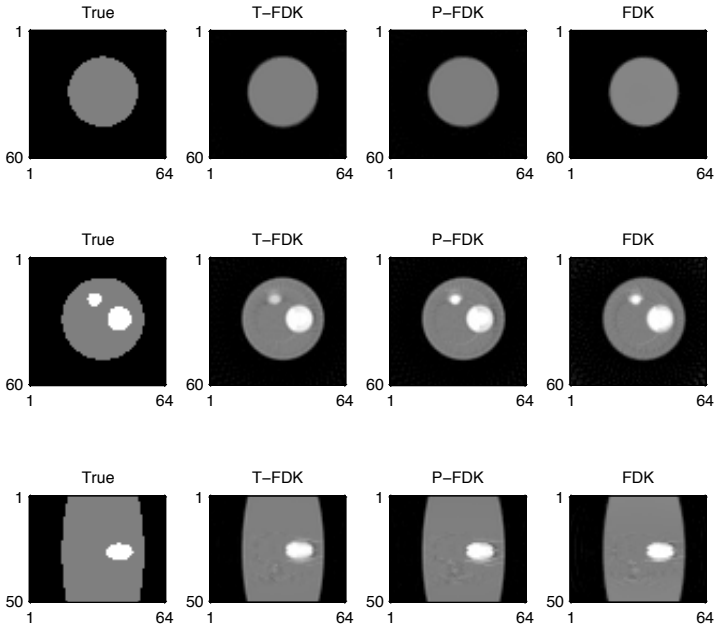

Figure 3: True and reconstruct images of a phantom for the first and middle slices, along with a side view of the phantom.

3. T-FDK

Instead of filtering and backprojecting after rebinning to the parallel geometry, the T-FDK method interpolates the projection data to form a Cartesian grid for the projection data on a virtual detector found at the axis of rotation. Before interpolation, the grid at this virtual detector can be seen in Figure 4. Instead of simply interpolating (in the vertical direction) a rectangle that is contained within this curved grid, we are interested in extrapolating in order to create a rectangle that contains the curved grid.

The function gridding.m performs this interpolation/extrapolation. It takes in the CT geometry and the projection matrix, and returns the new projection matrix, along with coordinates for the new grid. This function uses interp1 and currently performs linear interpolation, but this method can be changed. We will refer to the new projection after interpolation/extrapolation

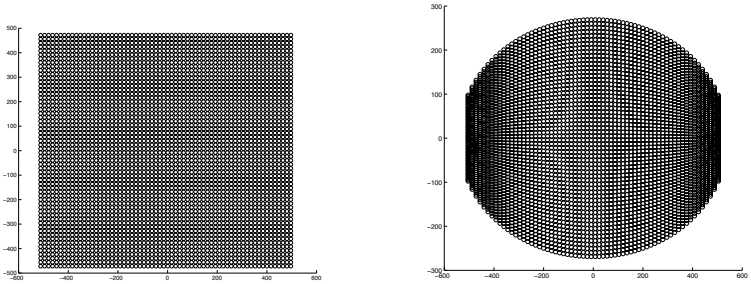

Figure 4: The left figure represents the projection grid on the detector, while the right figure represents the same grid on the virtual detector located at the center of rotation.

in the t-direction as $p^{pr}(r, t', \phi)$.

After performing the interpolation/extrapolation, the algorithm continues with pre-weighting and filtering, according to the equations

$$\begin{aligned} w^{pr}(r, t') &= \frac{\sqrt{D_{s0}^2 - r^2}}{\sqrt{D_{s0}^2 - r^2 + t'^2}} \\ \hat{p}^{pr}(r, t', \phi) &= \Big(w^{pr}(r, t') p^{pr}(r, t', \phi) \Big) * g^p(r). \end{aligned}$$

The new cosine weighting function, $w^{pr}(r, t')$, can be derived directly from Figure 2. Finally, the backprojection occurs with

$$f_{pr}(x, y, z) = \int_0^{2\pi} \hat{p}^{pr}(r(x, y, \phi), t'(x, y, z), \phi) d\phi,$$

where $r(x, y, \phi)$ is the same as before, and

$$t'(x, y, z, \phi) = \frac{z\sqrt{D_{s0}^2 - r(x, y, \phi)^2}}{\sqrt{D_{s0}^2 - r(x, y, \phi)^2} + v(x, y, \phi)}.$$

Again, this follows from the similar triangle identity

$$\frac{t'}{d'} = \frac{z}{d' + v}.$$

4. Conclusion/Future Work

Preliminary results show similar images for all three algorithms (FDK, T-FDK, and P-FDK). Future work should be done testing the methods on additional phantoms and real CT datasets. Additional work can be done on the interpolation/extrapolation step. Specifically, examining the effects of different interpolation methods, and adjusting the dt and nt values of the new grid. Currently, the new grid keeps nt constant. Similar to how our rebinning steps can be used to decrease ds, the interpolation step can be used to decrease dt. Lastly, parallel_feldkamp_example.m provides an example that runs the code found in the folder rebinnedFunctions, which can be placed in the folder irt/fbp.

5. References

[1] M. Grass, 3d cone-beam ct reconstruction for circular trajectories, Phys Med Biol. 45 (2000) 329347.

[2] J. Fessler, Book chapter section 3.9 (2012).

[3] H. Turbell, Cone-beam reconstruction using filtered backprojection, Dissertation no. 672, Linkoping Studies in Science and Technology (2001).