1998 International Computer Symposium
Workshop on Image Processing and Character Recognition
December 17-19, 1998, N.C.K.U., Tainan, Taiwan, R.O.C.

A FAST APPROXIMATION ALGORITHM FOR 3D IMAGE RECONSTRUCTION

Tung-Kuang Wu *and* *Martin L. Brady*

Department of Information Management
Minghsin Institute of Technology
HsinFong, HsinChu, Taiwan, R.O.C.
Email: tkwu@mis1.mis.mhit.edu.tw

ABSTRACT

In this paper we present a fast algorithm for the computation of planar projections of a 3D image (i.e., the three-dimensional Radon Transform, or 3DRT) and the inverse problem of reconstructing a 3D image from its projections. For an $N \times N \times N$ image $I(x,y,z)$, it would generally require $O(N^5)$ time to compute the projections of N^3 different planes, since each plane contains about N^2 sample points from $I(x,y,z)$. Reconstruction of an $N \times N \times N$ image from N^3 projections also would usually take $O(N^5)$ time. Our fast approximate algorithmth performs planar projection and backprojection in only $O(N^3 \log N)$ time.

1. Introduction

The reconstruction of 3D volume data by 2D sectional imaging (a slice at a time) is commonly used in X-ray computed tomography. Consecutive 2D sections are stacked to form a 3D image, with the data for each section being acquired and reconstructed independently of any other section ([10], [12]). But this approach makes poor use of the available imaging photons in the case of nuclear medicine (in particular, Positron Emission Tomography) by rejecting the direction of the photons outside a single section. One of the reasons for this waste of activity was that no practical computer algorithm had been developed for reconstructing 3D images from all the data which could be acquired if the interslice collimators were omitted [15]. But the demand to increase the sensitivity of reconstructed images by making better use of the oblique rays is driving the development of 3D reconstruction of digital images.

For an $N \times N \times N$ image $I(x,y,z)$, it would generally require $O(N^5)$ time to compute the projections of N^3 different planes, since each plane contains about N^2 sample points from $I(x,y,z)$. Reconstruction of an $N \times N \times N$ image from N^3 projections also would usually take $O(N^5)$ time. A numerical algorithm for the approximate reconstruction from a finite set of plane integrals is reported in [4] and [17] with second order difference as the filter. [14] and [16] developed a two-stage reconstruction algorithm using two successive 2D filtered backprojection processes. The two-stage method reduces the computational steps from $O(N^5)$ to $O(N^4)$. An novel algorithm requiring only $O(N^3 \log N)$ computation steps is reported in [11] for 3D reconstruction (using the Fourier method) from data collected in Fourier space at points arranged on a grid of concentric cubes. Although fast in computation time, the special data collection method is limited to applications in MRI tomography. In the case of PET, to acquire a full set of planar projections is not possible due to the geometry of the detectors. To solve the problem, some researchers ([5], [6], [15] and [18]) propose a two-pass 3D reconstruction procedure by first estimating (reconstructing) the voxel value in each grid point using the conventional 2D backprojection method, followed by a planar forward projection process to approximately compute the planar projections that are not collected by the detectors. After the full set of projections are available, a 3D reconstruction is applied. The complete procedure is shown in Algorithm 1.

Algortihm 1. 3D Image Reconstruction Procedure

```
Step 1: Perform a 2D image reconstruction and
        obtain an estimation of the 3D voxel data set.
Step 2: Perform 2D forward projection to the 3D
        data to obtain a set of planar projections that
        are not collected by the detector.
Step 3: Perform a 3D image reconstruction using
        both the detector-collected projections and
        forward-projected planar projections
        (computed in Step 2).
```

Studies shown in [6], [13] and [19] indicate that it takes hours to compute the above 3D image reconstruction and between 78 to 86% of the total computation time is devoted to the forward projection and backprojection phases. As a result, a fast forward/backward projection algorithm has important implications to 3D imaging, which can potentially speed up the reconstruction process.

In this paper we will present a fast algorithm for the computation of planar projections of a 3D image (i.e., the three-dimensional Radon Transform, or 3DRT) and the inverse problem of reconstructing a 3D image from its projections.

The remainder of this paper is organized as follows. A brief description of the *Approximate Discrete Radon Transform* (ADRT) algorithm, upon which our projection algorithms are based, is given in Section 2. The extension of the ADRT to compute approximate planar projections is described in Section 3. A truely 3D image reconstruction algorithm based on the planar projection algorithm is presented in Section 4. Finally, we give a summary of our work and further research orientation in Section 5.

2. Approximate Discrete Radon Transform

In the 2D Discrete Radon Transform, a set of summed projections is computed through a 2D image at various orientations. Consider an $N \times N$ image, $I(x,y)$. If the sampling is dense enough so that every pixel is used to compute at least one ray at any given projection angle, then the number of sequential operations needed to compute a single 2D projection will be $\Omega(N^2)$, and computing projections at N different angles (independently) will require $\Omega(N^3)$. However, for discrete non-interpolated line sampling algorithms, different orientations do not necessarily have to be computed independently. There can be a great deal of intersection between the sample points of lines at neighboring angles. For example, Figure 1 shows lines at two orientations that share half of their data points. One can potentially save time by computing such shared partial sums only once for use in two or more lines. Unfortunately, it is generally difficult to determine the proper subsets and order the computations accordingly, and it may be easier to simply calculate the sums independently. The Approximate Discrete Radon Transform defines a new line sampling algorithm that sacrifices a little accuracy and generality in order to generate line rasterizations that allow maximum sharing of intermediate terms. As a result, the ADRT is able to compute a specific set of N projections over an $N \times N$ image in only $O(N^2 \log N)$ steps. We briefly describe the ADRT algorithm below (see [2],[3] for a detailed description).

The ADRT algorithm computes projections at N different angles in the range $\theta = 0^\circ$ to 45° concurrently. Projections in the ranges 45° - 90°, 90° - 135°, and 135° - 180° are obtained from symmetric variations of the basic 0° - 45° computation that we discuss here. Rays are projected from integral points on the base line, $y = 0$, to integral points on the line $y = N-1$. In particular, rays from pixels $(x,0)$ to the pixels $(x+a, N-1)$ are projected, where $a = 0, 1, 2, \ldots, N-1$, to obtain projections at N different angles, $\theta = \tan^{-1}(a/(N-1))$. Additional samples between $(x,0)$ and $(x+a, N-1)$ are taken from pixels near the ray that passes through these end points.

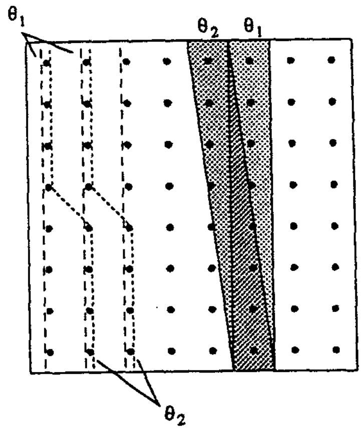

Figure 1. Overlap between neighboring lines.

In order to simplify the discussion, the image is right-padded with zeroes at coordinates (x,y), $N \leq x < 2N$, $0 \leq y < N$. $2N^2$ rays then projected from pixels $(x,0)$ to $(x+a \text{ [mod } 2N], N-1)$, where $x = 0, 1, 2, \ldots, 2N-1$. The cyclic definition of x-coordinates means that rays that "fall off" one of the sides of the image continue by taking samples from the zero-padded region. Each ray will draw exactly one sample from each row, and thus consist of the sum of N points. (In practice, unnecessary computations within the zero-padded region can be avoided by checking boundary conditions.)

Consider the N sample points of a ray projected using the ADRT algorithm. A set of p consecutive sample points are defined along the ray, from lower end point (x,y) to upper end point $(x+a, y+p-1)$, as a p-point *segment* of the ray. The *x-displacement* of a segment is the difference in the x-coordinates of its upper and lower end points. (The *y-displacement* of a p-point segment is $p-1$.) The *position* of a segment is described by the coordinates of its lower end point. The ADRT algorithm proceeds recursively, first computing the sums of a set of 2-point segments that are contained in many rays. Pairs of 2-point segments are then added to form sums along 4-point segments, and so forth.

The first three steps of the ADRT are illustrated in Figure 2 for an 8x8 portion of an image. It begins by computing the sums of pairs of pixels, to form segments of two pixels each, with orientations of 0° and 45°. Specifically, $2N \times N/2$ sums of vertical 2-point segments are obtained by adding the pixels $I(x,y)$ and $I(x,y+1)$ for $x = 0, 1, 2, \ldots, 2N-1$ and $y = 0, 2, 4, \ldots, N-2$. Similarly, $2N \times N/2$ diagonal 2-point segment sums are obtained by adding the pixels $I(x,y)$ and $I(x+1, y+1)$ in the same range (recall that the x-coordinates are taken modulo $2N$).

In the second pass, pairs of 2-point segments are added to obtain 4-point segments. The vertical 2-point segments are used to compute 4-point segments with x-displacements of 0 and 1, and the diagonal segments are used to compute 4-point segments with x-displacements of 2 and 3 (see Figure 2). In the third pass, pairs 4-point segments are summed to construct 8-point segments at eight different angles, with x-displacements from 0 to 7. In pass i, 2^i-point segments with x-displacements of 0, 1, 2,..., 2^i-1 are computed, where a segment of x-displacement a is the sum of two 2^{i-1}-point segments of x-displacement $\lfloor a/2 \rfloor$. The algorithm completes in $\log N$ passes, computing N-point segments with x-displacements a from 0 to $N-1$, from positions $(x,0)$, x = 0, 1, 2,..., $2N-1$.

A pseudo-code description of the ADRT algorithm for angles in the range 0–45° is shown below (Algorithm 2). $R_i(x,y,a)$ denotes a segment computed in pass i with lower end point (x,y) and x-displacement a. The initial image $I(x,y)$ is represented as $R_0(x,y,0)$, and the image is padded with an additional N^2 zeroes on the right side. In each pass, the number of different angles (x-displacements) doubles, but the number of y-coordinates that contain lower end points is halved, so the total amount of data remains constant. After $\log N$ passes, the projection data is obtained from $R_{\log N}(x,0,a)$, where the angle of a line (x,a) is $\theta = \tan^{-1}(a/(N-1))$ with respect to the positive y-axis and its distance from the origin is $d = x\cos\theta$.

Algorithm 2. ADRT computation for 0° – 45°.

```
Initialize R_0:
for y = 0 to N-1
{   for x = 0 to N-1      $R_0(x,y,0) = I(x,y)$
    for x = N to 2N-1     $R_0(x,y,0) = 0$
}
Compute the RT:
for i = 1 to log N
    for a = 0 to $2^i-1$
        for y = 0 to $N-2^i$ step $2^i$
            for x = 0 to 2N-1
                $R_i(x,y,a) = R_{i-1}(x, y, \lfloor a/2 \rfloor) +$
                $R_{i-1}(x+\lceil a/2 \rceil, y+2^{i-1}, \lfloor a/2 \rfloor)$
```

Exactly $2N^2$ segment sums are performed in each pass i of Algorithm 2, and thus only $O(N^2 \log N)$ operations are required to complete the algorithm, an $O(N/\log N)$ speedup over the time to compute the projections independently. Therefore, N approximate projections in only slightly more time than is required to compute a single projection are obtained. Furthermore, the ADRT requires only $\log N$ simple parallel steps, and can use up to $2N^2$ processors.

The ADRT casts a total of $2N^2$ rays (d,θ) (although $N/2$ of them pass entirely through the padded zeroes). The specific set of values (d,θ) are determined by the image size, N. Note that the angles and spacings are not uniformly distributed ($\Delta\theta$ and Δd are not constant). However, the set of rays are quite dense, so a desired sample (d,θ) can be obtained from the computed set by interpolation. The sample points along a ray are not interpolated, and may even be greater than 1/2-unit from the intended ray. However, the maximum distance of any sample from its ray has been proven to be less than $(1/2)(\log N-1)$, and is in practice even smaller [2]. In the first part of our work, we extend this technique to generate a variety of fast approximate 3D projection algorithms. We then show how to apply these algorithms to 3D image reconstruction.

3. The Approximate Dsicrete Planar Projection (ADPP) Algorithm

Using the fast 2D Approximate Discrete Radon Transform (2D ADRT) algorithm designed for the two-dimensional Radon Transform presented in section 2, we construct an algorithm to compute approximate planar projections at N^2 different orientations, in only $O(N^3\log N)$ time. A total of N planes are projected at each angle, and most pass

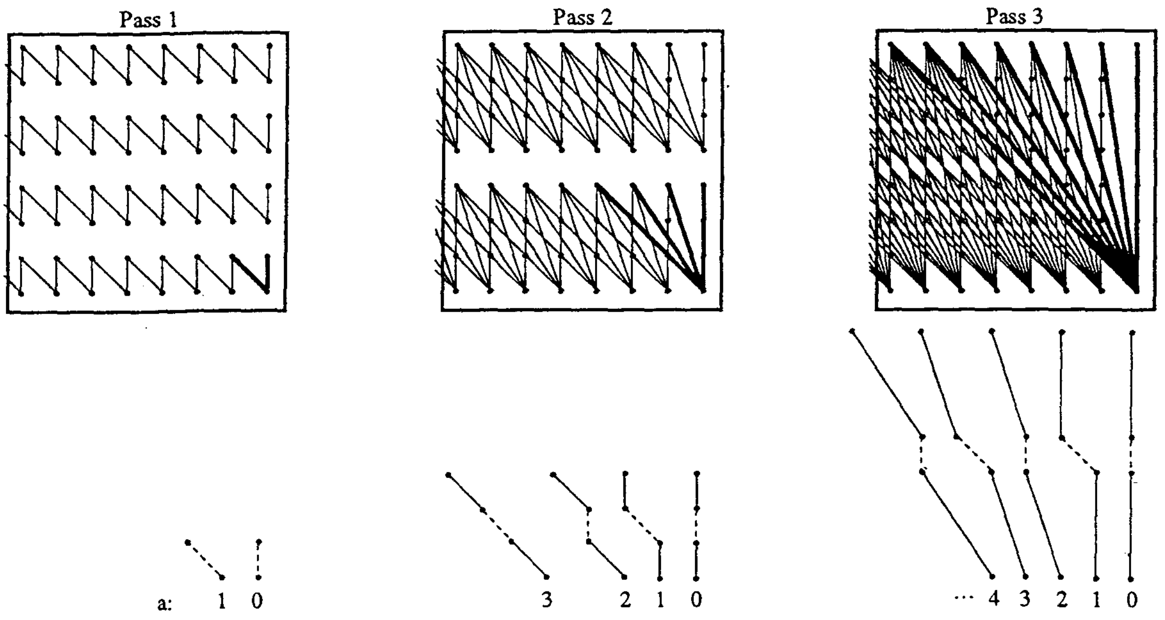

Figure 2. Segments computed in the first three passes of the ADRT. The upper half shows the set of segments computed. For each pass, one complete set of angles is highlighted, beginning in the lower right corner of the $N\times N$ image. Below, the construction of the segments (using the shorter segments from the previous pass) is illustrated. Solid lines represent segments computed in the previous pass, and a dashed line joins two segments to be added.

through $\Omega(N^2)$ data points. Our algorithm is thus $\Theta(N^2/\log N)$ faster than computing them independently.

A planar surface integration (planar projection using the addition operator) can be computed by summing parallel line projections. For example, if each line in Figure 3 represents a line projection (with no pixels overlapping in each line projection) along a one-pixel wide strip specified by α, adding all of the parallel line projections along angle β gives the planar surface integration specified by α and β. This implies that planar integration can be computed using a two-phase procedure: (1) first compute line integrations (with orientation α) within slices of 3D data, (2) sum cross-slice line integrations (of angle α) along a specific angle (β).

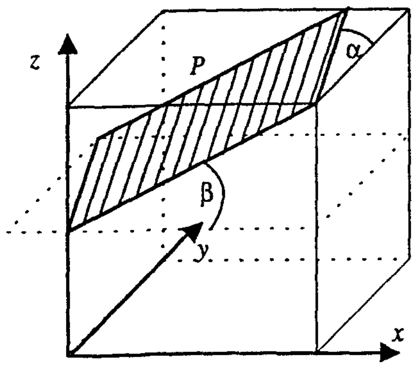

Figure 3. Planar surface projection computation by adding numerous parallel line projections.

The availability of the ADRT algorithm allows us to make use of the above two-phase procedure very efficiently. By proper arrangement of a 3D image data set, the two-phase *Approximate Discrete Planar Projection* (ADPP) (for $0 \le \alpha \le \pi/4$ and $0 \le \beta \le \pi/4$) can be computed with multiple applications of the 2D ADRT algorithm. The ADPP algorithm proceeds in two phases. In the first phase, we compute line projections within each 2D *slice* of the image, $I(x,y,z=c)$, where c is the set of integers between 0 and $N-1$ (data is arranged in a 3D array, with size $N \times N \times N$, as shown in Figure 4(a)). This is done by applying the (two-dimensional) ADRT algorithm to each of the N slices. This yields a set of line projections $P_{1D}(d,\alpha,z)$. In the second phase, we rearrange the computed line projections to a new 3D data set (as shown in Figure 4(b)) and again apply the 2D ADRT algorithm, this time within planes of constant α, i.e., slices $P_{1D}(d,\alpha=c,z)$. This is done so as to sum *coplanar* projections of a fixed α. We thus compute planar projections, $P_{2D}(d,\alpha,\beta)$, from line projections of a fixed angle β, where d denotes the distance of the plane from the origin, and (α,β) specifies its orientation. Note that in the ADPP algorithm, we represent the terms α and β with a and b, where $\alpha = \tan^{-1}(a/(N-1))$ and $\beta = \tan^{-1}(b/(N-1))$. Selecting any horizontal slice (say $P_{1D}(d,a=0,z)$, highlighted in Figure 4(b)) and integrating along some line on that slice as shown in Figure 4(c), a line integral specified by a and b is equivalent to a planar integration (also specified by a and b) in the original object space. For example, the line integrations $L1$ and $L2$ in Figure 4(c) are equivalent to $P1$ and $P2$ in Figure 4(d). We give more details of the ADPP algorithm as follows.

Assume the 3D volume is represented by a 3D array $I(x, y, z)$. The following pseudo code gives a high level description of the two-phase procedure.

Algorithm 3. ADPP computation for $0 \le \alpha \le \pi/4$ and $0 \le \beta \le \pi/4$

```
for range of angle [0,π/4] do
{
    for z=0 to N-1 do
    {
        apply 2D ADRT to slice I(x, y, z)
        store the output in 3D array
            P_1D(x, a, z), a=0,N-1
    }
    for range of angle [0,π/4] do
    {
        for a=0 to N-1 do
        {
            apply 2D ADRT to slice P_1D(x, a, z)
            store the output in 3D volume
                P_2D(x, a, b), a=0, N-1
        }
    }
```

The above algorithm computes N^2 sets of planar projections for $0 \le \alpha \le \pi/4$ and $0 \le \beta \le \pi/4$. The algorithm contains two phases, and each phase requires N applications of the 2D ADRT. Since the ADRT algorithm requires $O(N^2 \log N)$ time on an $N \times N$ 2D image, the ADPP algorithm takes only $O(N^3 \log N)$ time. By comparison, it takes $\Theta(N^2)$ time to compute a single planar projection and $\Theta(N^3)$ time for a set of N planar projections, provided that each planar projection is computed independently. A total of $\Theta(N^5)$ time is thus required to obtain all N^2 sets of planar projections independently. As a result, our algorithm achieves a speedup of $O(N^2/\log N)$.

4. Truly 3D Image Reconstruction

The planar projection algorithm presented in Section 3 can be applied to the image reconstruction problem in several ways. It can be used to compute forward projection, 2D image reconstruction, and truly 3D image reconstruction. In this section, we base our discussion on Algorithm 1 (the 3D image reconstruction procedure) and show when and how the planar projection algorithm can be of use.

4.1 Application of the ADPP in 3D Image Reconstruction

Obviously, the ADPP algorithm can be used in the forward projection process (Step 2 of Algorithm 1). It happens that the same algorithm can also be used in the backward projection process (Step 3, as will be discussed in later section). Note that if Step

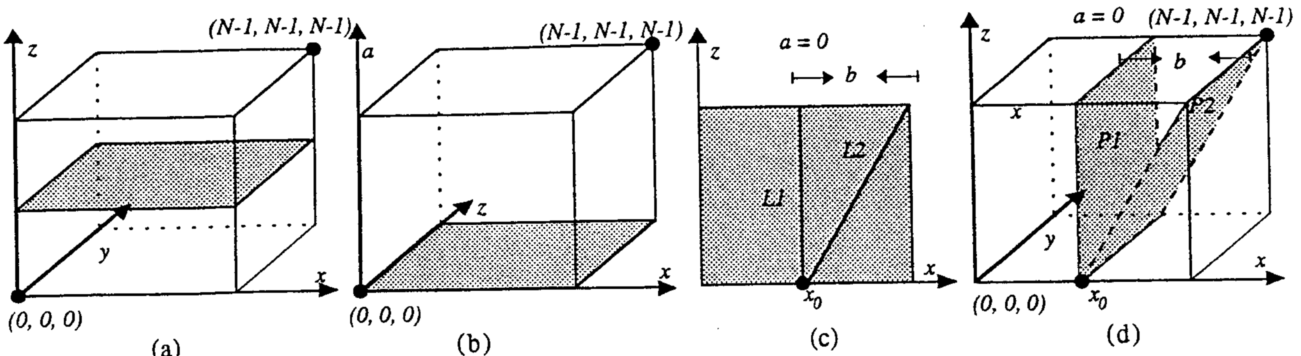

Figure 4. Planar integration procedure. (a) a 3D data set. (b) a reorganized 3D data set after the first application of ADRT. (c) A slice of 2D array extracted from (b), the two line integrations ($L1$ and $L2$) are equivalent to the two plane integrations ($P1$ and $P2$) shown in (d). Note the correspondence of b between (c) and (d).

1, the 2D image reconstruction, is computed using the 2D ADRT, the whole 3D image reconstruction procedure then contains repeated applications of the 2D ADRT, plus some filtering steps. This greatly simplifies the implementation of the 3D reconstruction process.

4.2 Linogram Property of the ADPP

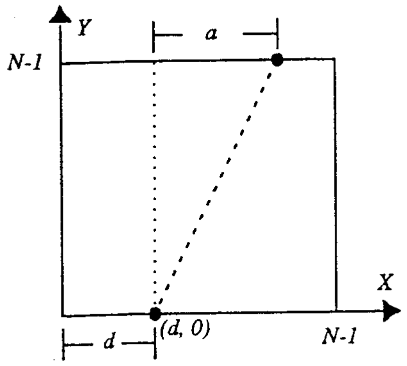

Figure 5. A plane (P) specified by (d, a, b) in the object space. The plane can also be regarded as one that is spanned by the two lines $L1$ and $L2$.

A line in 2D is specified by two variables d and θ, and the line can be represented by the equation $d = x\cos\theta + y\sin\theta$. A line projection process based on this line equation maps a fixed point in the (x, y)-coordinate system to a sinusoidal curve in (d, θ) space (which is why the mapping is called a *sinogram* [9]). As a result, conventional methods to reconstruct a fixed point (x, y) (through backprojection) from projection space require summation of the projection data that are located on the sinusoidal curve. We can, however, arrange the projection data so that projections that pass through a fixed point in the object space correspond to a straight line in the projection space. A sampling of this kind is called a *linogram* [7]. A direct advantage of a linogram sampling is that no interpolation step is required to produce uniform samples when computing backprojection through the Fourier Transform method ([1], [8]). It also indicates that the backprojection process can benefit from a fast line rasterization algorithm (in particular, the 2D ADRT algorithm).

Recall that a projection is specified by (d, a) in the 2D ADRT, where d represents the distance along the x-axis between the origin and the line, and a represents the x-coordinate difference (x-displacement) between the starting and end points of the line. An example of a line constructed by the 2D ADRT (for an $N \times N$ image) is depicted in Figure 5. In general, a line that passes through points (d, 0) and ($d+a$, $N-1$) can be represented as:

$$(N-1)x - ay = (N-1)d \qquad (1)$$

By rearranging Equation (1) to the (d, a)-coordinate system, we get:

$$\frac{y}{(N-1)}a + d = x \qquad (2)$$

As we can see, a line in (x, y) object space (Equation (1)) is mapped to a line in (d, a) projection space (Equation (2)). This shows that the non-uniform sampling of the 2D ADRT algorithm meets the linogram property. It remains to be shown that the ADPP algorithm preserves the linogram property by repeated applications of the 2D ADRT algorithm.

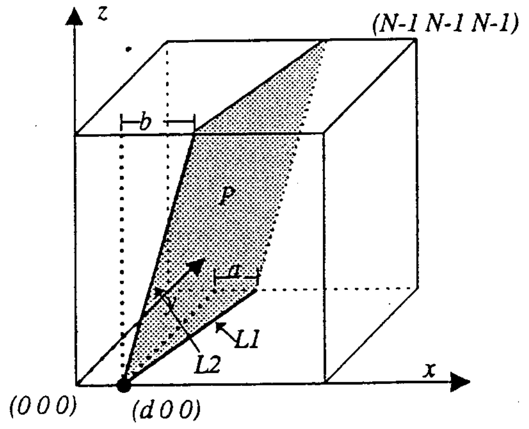

Figure 6. A plane (P) specified by (d, a, b) in the object space. The plane can also be regarded as one that is spanned by the two lines $L1$ and $L2$.

To prove that the ADPP algorithm meets the linogram property, we need to show that planes that

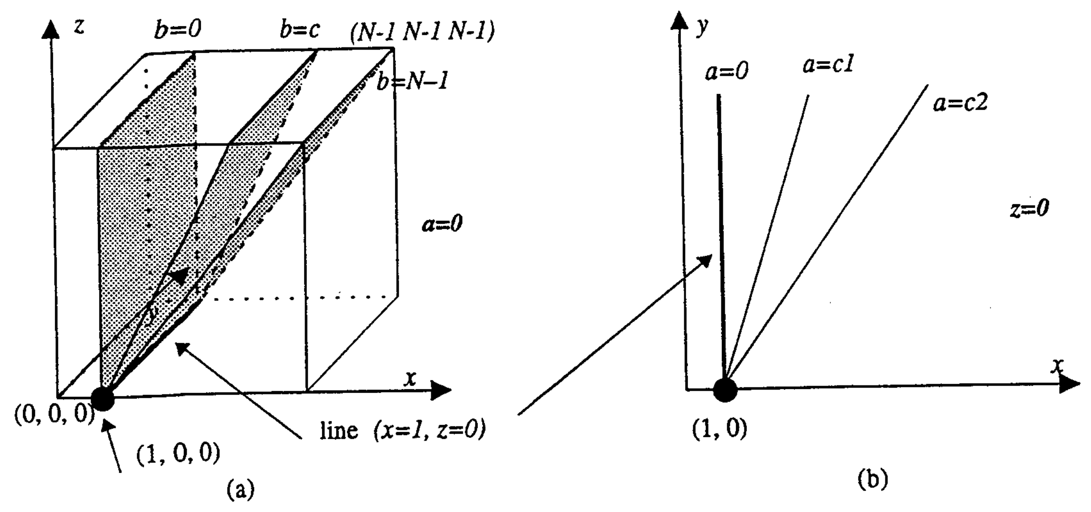

Figure 7. A two-phase backprojection example showing that a grid point can be reconstructed by summing planes that intersect at a common line, and then summing over such intersecting lines of various orientations within a slice of fixed z axis.

pass through a given point in the object space are mapped to points that are located on a plane of the projection space. Consider a plane P specified by (d, a, b) in object space, as shown in Figure 6. Note that the plane can also be considered to be defined by the two perpendicular lines, $L1$ and $L2$, located on the x-y and x-z planes respectively and sharing a common end point $(d, 0, 0)$. The two lines have end points $(d, 0, 0)$, $(d+a, N-1, 0)$ and $(d, 0, 0)$, $(d+b, 0, N-1)$ respectively and can be represented as follows.

$$L1: \quad \frac{y}{(N-1)}a + d = x \quad (3)$$

$$L2: \quad \frac{z}{(N-1)}b + d = x \quad (4)$$

Also note that a finite plane is formed by an aggregate of a finite number of parallel lines. For example, in Figure 6, lines that are parallel to $L2$ and originating from end points located on $L1$ can uniquely determine all the points on the plane P. As a result, combining the two equations by replacing the two end points of $L2$ with $(d_0, y, 0)$ and $(d_0+b, y, N-1)$, where $d \le d_0 \le d+a$ and is represented by Equation (3) as $d_0 = x - \frac{y}{(N-1)}a$, we obtain a number of parallel lines that uniquely specify the plane P as follows.

$$P: \frac{z}{(N-1)}b + \frac{y}{(N-1)}a + d = x \qquad (5)$$

Thus, a plane P is mapped to a point (d, a, b) in the projection space. Furthermore, assume that plane P passes through a grid point (x_0, y_0, z_0). The equation of P can be rearranged as follow:

$$P': \quad d + \frac{y_0}{(N-1)}a + \frac{z_0}{(N-1)}b = x_0 \qquad (6)$$

It can be easily seen that planes that pass through a fixed point in the object space indeed map to points that located on a plane in the projection space, and thus the ADPP algorithm does meet the linogram property.

Finally, we need to show that the planar projection specified by Equation (6) can be computed by application of the ADPP algorithm, by demonstrating the four end points of P' are located at integer grid points. This can be shown by substituting a and b with 0 or $N-1$ respectively, and checking the value of d. The results of these substitutions are shown below.

$a \backslash b$	0	$N-1$
0	$d = x_0$	$d = x_0 - z_0$
$N-1$	$d = x_0 - y_0$	$d = x_0 - y_0 - z_0$

Since x_0, y_0, z_0 are integer, d is thus an integer that ranges from $-2(N-1)$ to $N-1$.

4.3 ADPP for 3D/2D Backprojection

The linogram feature of the ADPP algorithm ensures that the backprojection procedure can be taken along planes, and thus allowed an efficient implementation with repeated applications of the 2D ADRT algorithm. We give an intuitive example in Figure 7, which shows how a point can be backprojected from planar projections. Note that we use terms (a, b), the x- and y-displacement, instead of (θ, ϕ) to specify the orientation. Figure 7(a) shows planes (planar projections) with $d=1$, $a=0$ and varying b that pass through grid point $(1, 0, 0)$. As can be seen, the planes intersect at a common line $(x=1, z=0)$ on the y-z plane. As we change the value of angle a of the planar projections, the common line of intersection also changes its orientation, but still pass through a common grid point, $(1, 0, 0)$, as illustrated in Figure 7(b). An integration over all of the linear projections (represented by lines) thus reconstructs grid point $(1, 0, 0)$. By varying the values of d, a, and b, and summing the planar projections accordingly, we can reconstruct the value of different grid points. Note that each of the planes (planar projections) shown in Figure 7(a) is represented by a point in the projection space. More specifically, these points are located on the same plane $(a=0)$ of the (d, a, b)-volume. As a result, application of the ADRT algorithm to slices with varying b and constant a give a set of backprojected lines (linear projections), as shown in Figure 7(b). Summing all the backprojected linear projections along different planes of a then give the backprojected grid points. This is how our two-phase ADPP-based reconstructed algorithm operates.

Algorithm 4. Procedure for the computation of forward/backward projections and filtering.

```
Forward Projection:
Phase 1:
    for slice $I(x, y=c, z)$, $0 \le c < N$ do
    {
        for $0 \le i < 4$ do
            Compute line projections
            $P1_i(x, y=c, a)$ using the 2D ADRT
    }
Phase 2:
    for slice $P1_i(x, y, a=c)$,
    $0 \le c < N$ and $0 \le i < 4$ do
    {
        for $0 \le j < 4$ do
        Compute planar projections
        $P2_{ij}(x, b, a=c)$ using the 2D ADRT
    }
Backward Projection
Filtering:
    Compute second order difference
    $P''2_{ij}(x, b, a)$, $0 \le i, j < 4$
    from planar projections $P2_{ij}(x, b, a)$
Phase 3:
    for slice $P''2_{ij}(x, b, a=c)$,
    $0 \le c < N$, and $0 \le i, j < 4$ do
    {
        Compute projections
        $P3_{ij}(x, y, a=c)$ using the 2D ADRT
    }
    Sum partial projections
    $P3_{ij}(x, y, a=c)$, $0 \le j < 4 \Rightarrow P3_i(x, y, a=c)$
Phase 4:
    for slice $P3_i(x, y=c, a)$,
    $0 \le c < N$ and $0 \le i < 4$ do
    {
        Compute line projections
        $P4_i(x, y, z)$ using the 2D ADRT
    }
    Sum partial projections
    $P4_i(x, y, z)$, $0 \le i < 4 \Rightarrow \tilde{I}(x, y, z)$
```

We give a detailed description of the reconstruction in Algorithm 4. The algorithm presented in Section 3 computes forward planar projections at angles corresponding to $0 \le \alpha, \beta \le \pi/4$ ($0 \le a, b < N$) in the spherical coordinate system. Other angle ranges can be computed by slight modifications to the basic algorithm. A total of 16 applications of the ADPP algorithm (for $0 \le \alpha, \beta \le \pi$) are thus necessary to complete the full planar projection process. Since each application of the ADPP algorithm takes $O(N^3 \log N)$ time for an $N \times N \times N$ 3D image, the total computation time for N^3 planar projections remains $O(N^3 \log N)$. Note, however, that in an actual implementation, the ADPP algorithm can be broken down into two steps and some of the repeated line projection computations in the first phase can be avoided. We give the procedure for the computation of the full set of forward and backward projections (plus the filtering step) below. We also present the forward projection procedure, since it is needed to compute forward planar projections for use with our study. $I(x, y, z)$ and $\tilde{I}(x, y, z)$ represent the original and the reconstructed 3D image respectively.

Note that 20 (rather than 32) applications of the 2D ADRT (to each of N slices of size $N \times N$) are required in the forward or backward projection process.

The 3D image reconstruction procedure can be performed starting from the filtering step in Algorithm 4 if a full set of planar projections are available. However, the ADPP-based algorithm can also be used to compute the 3D reconstruction if only line projections are available. By starting from the second phase of Algorithm 4, we forward-project the line projections to 2D planar projections. Once the planar projections are computed, a 3D image reconstruction can be performed. The computational complexity remains $O(N^3 \log N)$, however the complicated filtering steps required in 2D image reconstruction are avoided. Note that the data must be collected in linogram fashion (see [1] for approaches in collecting Linogram projections) or interpolated into the proper sampling distribution.

5. Conclusion

In this paper we have presented an asymptotically fast approximate planar forward/backward projection algorithm which, when combined with a simple second order difference filter, can compute truly 3D image reconstruction (from projections) very efficiently. This algorithm is directly applicable to medical imaging techniques that collect planar projection data directly, such as in Magnetic Resonance Imaging (MRI) and Positron Emission Tomography (PET). In this case, the data must either be collected in Linogram fashion [1] or interpolated into the proper sampling distribution. The fast filtered backprojection can then be applied. However, the approach can also be applied to data collection techniques that produce line projection data. Often, such data are collected as independent 2D slices, and reconstructed independently using 2D reconstruction methods. Normally, this would require $O(N^4)$ time to reconstruct data with N angles, N projections per angle, and N slices. Furthermore, it requires a more complex (and in practice, time-consuming) filtering operation, a convolution with the function whose Fourier transform is $|\omega|$. Note that after the first phase of our *forward* projection algorithm (Algorithm 4), we obtain this same type of data. We can therefore apply our method to this data by starting from the second phase of the forward projection algorithm to compute planar projections from the line projections. We then compute a full 2-pass 3D reconstruction as before. The backprojection time is $O(N^3 \log N)$. Furthermore, the relatively complicated Fourier domaing filtering is replaced by a simple local second order difference filter, requiring only $O(N^3)$ time [17].

A qualitative comparison of the reconstructed images between the ADPP-based 3D image reconstruction and ADRT-based reconstruction methods would be interesting. Also, there might exist other approximate line sampling techniques, which may be potentially faster and more efficient than the ADRT-based one. An approximate curvature (non-planar) sampling technique should have many potential applications in the fields of image processing and computer vision.

6. Reference

[1] L. Axel, G. T. Herman, and D. Roberts, "Linogram Reconstruction for Magnetic Resonance Imaging (MRI)", *IEEE Trans. on Medical Imaging*, **9** (4), pp. 447-449 (1990).

[2] M. L. Brady, "A Fast Discrete Approximation Algorithm for the Radon Transform", Tech. Report CSE-93-007, Dept. of Comp. Sci. and Eng., The Pennsylvania State University (Oct. 1993). (submitted for publication)

[3] M. L. Brady, W. Yong, "Fast Parallel Approximation Algorithms for the Radon Transform," *Proc. 4th ACM Symp. on Parallel Algorithms and Architectures*, pp. 91-99 (oct. 1992).

[4] M. Y. Chiu, H. H. Barrett, and R. G. Simpson, "Three-Dimensional Image Reconstruction from Planar Projections," *Journal of American Optical Society* **70** pp. 755-762 (1980).

[5] S. R. Cherry, M. Dahlbom, and E. J. Hoffman, "Evaluation of a 3D Reconstruction Algorithm for Multi-slice PET Scanners," *Physics in Medicine and Biology*, **37** (3), pp. 779-790 (1992).

[6] M. Defrise, D. Townsend, and A. Geissbuhler, "Implementation of Three-Dimensional Image Reconstruction for Multi-ring Positron Tomographs," *Physics in Medicine and Biology*, Vol. **35**, pp. 1361-1372 (1990).

[7] P. Edholm, and G. T. Herman, "Linograms in Image Reconstruction from Projections," *IEEE Trans. on Medical Imaging*, **MI-6** (4), pp. 301-307 (1987).

[8] P. Edholm, G. T. Herman, and D. A. Roberts, "Image Reconstruction from Linograms: Implementation and Evualation," *IEEE Trans. on Medical Imaging*, **7** (3), pp. 239-246 (1988).

[9] P. Edholm, and B. Jacobson, Poster, *1975 Meeting on Image Processing for 2-D and 3-D Reconstruction from Projections*, (1975).

[10] G. T. Herman, Image Reconstruction from Projections, (New York: Academic Press), 1980.

[11] G. T. Herman, D. Roberts, and L. Axel, "Fully Three-Dimensional Reconstruction from Data Collected on Concentric Cubes in Fourier Space: Implementation and a Sample Application to MRI," *Physics in Medicine and Biology*, Vol. **37**, pp. 673-687 (1992).

[12] A. C. Kak, "Image Reconstruction from Projections," *Digital Image Processing Techniques*, Academic Press (1984).

[13] P. E. Kinahan, and J. G. Rogers, "Analytic 3D Image Reconstruction Using All Detected Events," *IEEE Trans. on Nuclear Science*, **36** (1), pp. 964-968 (1989).

[14] P. C. Lauterbur, and C. M. Lai, "Zeugmatography by Reconstruction from Projections," *IEEE Trans. on Nuclear Science*, **NS-27** (3), pp. 1227-1231 (June 1980).

[15] J. G. Rogers, R. Harrop, and P. E. Kinahan, "The Theory of Three-Dimensional Image Reconstruction for PET," *IEEE Trans. on Medical Imaging*, Vol. **MI-6**, pp. 239-243 (1987).

[16] C. M. Lai, and P. C. Lauterbur, "A Gradient Control Device for Complete Three-Dimensional Nuclear Magnetic Resonance Zeugmatographic Imaging," *J. Phys. E: Sci. Instrum.*, **13**, pp. 747-750 (1980).

[17] L. A. Shepp, "Computerized Tomography and Nuclear Magnetic Resonance," *Journal of Computer Assisted Tomography*, **4** (1), pp. 94-107 (1980).

[18] M. W. Stazyk, J. G. Rogers, and R. Harrop, "Full Data Utilization in PVI Using the 3D Radon Transform," *Physics in Medicine and Biology*, Vol. **37** (3), pp. 689-704 (1992).

[19] D. W. Townsend, T. Spinks, T. Jones, A. Geissbuhler, and M. Defrise, "Three Dimensional Reconstruction of PET Data from a Multi-ring Camera," *IEEE trans. on Nuclear Science*, Vol. **36** (1), pp. 1056-1065 (1989).