EM+TV for 3D image reconstruction

Ming Yan
University of California, Los Angeles

August 11, 2010

1 Radon Transform

In order to use algebraic reconstruction methods, we have the following system of linear equations,

$$Ax = b, \tag{1}$$

here, A is the matrix defining the discrete radon transform, x is the image, and b is the measurements. The problem is to reconstruct x from b. The matrix A can be derived by Siddon's algorithm. Evey row of A defines a line integral along the line segment from source to detector. If we know all the source-detector pairs, we can construction the matrix. However, for 3D image reconstruction, the number of voxels is too large and to store the matrix is impossible. Another way is to use a function to calculate the projection Ax for given image x and another function to calculate the back-projection $A^t y$ for given sinogram data y.

2 EM+TV

For EM(Expectation Maximization)+TV(Total Variation), we assume that the noise in the measurements are poisson noise and independent. Then

$$P(b|Ax) = \prod_{i=1}^{M} e^{-a_i x} \frac{(a_i x)^{b_i}}{b_i!}, \tag{2}$$

where a_i is the i^{th} row of A.

The optimization problem is

$$\begin{aligned} &\underset{x}{\text{minimize}} \int |\nabla x| - \alpha \log P \\ &\text{subject to } x \geq 0, \end{aligned} \tag{3}$$

here ∇x is a vector, and each component is a linear operation on x. When we use ∇, we consider x to be 2D or 3D image. Otherwise, x is considered to be a vector, except stated.

Plugging into the probability gives us

$$\begin{aligned} &\underset{x}{\text{minimize}} \int |\nabla x| + \alpha \sum_{i=1}^{M} (a_i x - b_i \log(a_i x)) \\ &\text{subject to } x \geq 0. \end{aligned} \tag{4}$$

The KKT condition of this is

$$-\text{div}(\frac{\nabla x}{|\nabla x|})_j + \alpha \sum_{i=1}^{M} (a_{ji}(1 - \frac{b_i}{a_i x})) - y_j = 0 \qquad j = 1, \cdots, N \tag{5}$$

$$y \geq 0 \tag{6}$$

$$x \geq 0 \tag{7}$$

$$y^T x = 0. \tag{8}$$

It becomes

$$-\frac{x_j}{a_j^T \overrightarrow{1}} \text{div}(\frac{\nabla x}{|\nabla x|})_j + \alpha \frac{\sum_{i=1}^{M} (a_{ij}(1 - \frac{b_i}{a_i x}))}{a_j^T \overrightarrow{1}} x_j = 0 \qquad j = 1, \cdots, N. \tag{9}$$

or

$$-\frac{x_j}{a_j^T \overrightarrow{1}} \text{div}(\frac{\nabla x}{|\nabla x|})_j + \alpha x_j - \alpha \frac{\sum_{i=1}^{M} (a_{ij}(\frac{b_i}{a_i x}))}{a_j^T \overrightarrow{1}} x_j = 0 \qquad j = 1, \cdots, N. \tag{10}$$

Here a_j is the j^{th} column.

Denote

$$x_j^{EM} = \frac{\sum_{i=1}^{M} (a_{ij}(\frac{b_i}{a_i x}))}{a_j^T \overrightarrow{1}} x_j \tag{11}$$

and this is the EM update.

The KKT condition becomes

$$-\frac{x_j}{a_j^T \overrightarrow{1}} \text{div}(\frac{\nabla x}{|\nabla x|})_j + \alpha x_j - \alpha x_j^{EM} = 0 \qquad j = 1, \cdots, N, \tag{12}$$

and this is the optimality for a TV minimization problem. We can solve it by iterations. For this step, we consider this to be 2D or 3D image. For 2D, we have the new notations $u_{i,j}$.

This is nonlinear equation, so we consider a linearized version

$$-\frac{u_{i,j}^n}{V_{i,j}}\frac{u_{i+1,j}^n-u_{i,j}^{n+1}}{\sqrt{\epsilon+(u_{i+1,j}^n-u_{i,j}^n)^2+(u_{i,j+1}^n-u_{i,j}^n)^2}} \tag{13}$$

$$+\frac{u_{i,j}^n}{V_{i,j}}\frac{u_{i,j}^{n+1}-u_{i-1,j}^n}{\sqrt{\epsilon+(u_{i,j}^n-u_{i-1,j}^n)^2+(u_{i-1,j+1}^n-u_{i-1,j}^n)^2}} \tag{14}$$

$$-\frac{u_{i,j}^n}{V_{i,j}}\frac{u_{i,j+1}^n-u_{i,j}^{n+1}}{\sqrt{\epsilon+(u_{i+1,j}^n-u_{i,j}^n)^2+(u_{i,j+1}^n-u_{i,j}^n)^2}} \tag{15}$$

$$+\frac{u_{i,j}^n}{V_{i,j}}\frac{u_{i,j}^{n+1}-u_{i,j-1}^n}{\sqrt{\epsilon+(u_{i+1,j-1}^n-u_{i,j-1}^n)^2+(u_{i,j}^n-u_{i,j-1}^n)^2}}+\alpha u_{i,j}^{n+1}-\alpha u_{i,j}^{EM}=0 \tag{16}$$

```
void Forward_Projection(DoubleArray3D &image, double D,
 int q, double ss, double d, double dtheta, double ssz, int qz, DoubleArray3D &sino)
```

The function used to do forward-projection from image to sino. D,q,ss,d,dtheta,ssz,qz are parameters used to define the geometry of the machine.

```
void Backward_Projection(DoubleArray3D &image, DoubleArray3D &image_denote,
double D, int q, double ss, double d, double dtheta, double ssz, int qz,
DoubleArray3D &sino);
```

The function used to do backward-projection from sino to image. Here we have another image_denote is used to denote which voxel is need to update. For some voxels, we can assign 0 for it, and do not have to be updated.

```
void EMupdate(DoubleArray3D &sino, DoubleArray3D &image, DoubleArray3D
&image_denote, DoubleArray3D &sumA, int max_iter, double D, int q, double ss,
double d, double dtheta, double ssz, int qz);
```

The function used to do EMupdating.

```
void TVupdate(DoubleArray3D &image, DoubleArray3D &sumA, double alpha, int max_iter);
```

The function used to do TVupdating (no sinogram data is needed).
The main code

```
void main()
{
    int N_x = 128, N_y = 128, N_z = 128;   // the size of the image, N_x * N_y * N_z
    DoubleArray3D image(N_x,N_y,N_z);      // used to store the reconstructed image
    DoubleArray3D sumA(N_x,N_y,N_z);    // used to store the summation of columns
    DoubleArray3D image_denote(1.,N_x,N_y,N_z);
    double D = 125, d =64;                 // the parameters used to define the geometry
    double theta = 10.0;
    int q = 150;
```

```
    int qz = 128;
    double ss = 0.5;
    double ssz = 1.0;
    int clk1, clk2;
    int max_iter_EM = 3;
    int max_iter_TV = 10;
    int alpha = 5;

    readImgArray3D("phantom_data128", image);
// read a image, this is used just for test, in practice, we donot know the
// image. We can have to just read sinogram data.

    clk2 = clock();
    DoubleArray3D sino(1,2*q + 1,2*qz + 1,(int)floor(359./theta)+1);
// used to store the sinogram data

    Backward_Projection(sumA, image_denote, D, q, ss, d, theta, ssz, qz, sino);
// find summation of columns


    Forward_Projection(image, D, q, ss, d, theta, ssz, qz, sino);
// creat the sinogram data, for final use, load the sinogram data

    Initialization(image_denote, D, q, ss, d, theta, ssz, qz, sino);
// initialize the image, make sure some pixels are zeros;
    image = image_denote;

    for (int iter = 0; iter < 10; iter++)
    {
        printf("%d",iter);
        EMupdate(sino, image, image_denote, sumA, max_iter_EM, D, q, ss, d, theta, ssz, qz
// EMupdating
        TVupdate(image, sumA, alpha, max_iter_TV);

    }

    clk1 = clock() - clk2;

    printf("%6.5f sec\n", ( float )( clk1 )/ ( float ) (CLOCKS_PER_SEC));

    writeImgArray3D("image_data",image);

}
```