Assignment 5
Signal & Image Processing

Casper B. Hansen
University of Copenhagen
Department of Computer Science
fvx507@alumni.ku.dk

October 2, 2014

Abstract

In this assignment we discuss some of the fundamental mathematical principles that lays the foundation of medical imaging techniques, such as CT scanning. Among these we will focus on the Radon transform, and the inversion thereof.

Contents

1 Sinogram

In medical imaging we do not have an image source per se. The way the input is produced is by shooting, or projecting, rays through an object, and then read the scattered rays as they exit the object. So, all we have is a numerical value for each projection that is read. If we think of these as the integral over the projected rays, our task is then to recover the integrated function.

We can express this idea by the Radon transform, given in equation (1).

$$p(\xi, \phi) = \int f(x, y)\delta(x \cos\phi + y \sin\phi - \xi)\ dx\ dy \tag{1}$$

Where $f(x, y)$ is the function we wish to recover, δ is the Dirac delta (or impulse) function, ϕ is the angle of projection, and ξ is the offset of the projected line from the origin. Figuratively, at least in our case, we are trying to express, mathematically, all line projections (or exiting rays) covering the image plane for all angles $\phi \in M$.

In MatLab, we can simplify the calculation of this mathematical expression by simply rotating the source image, using `imrotate`, and integrate (`sum`) for each angle ϕ (see A.2.1 for implementation).

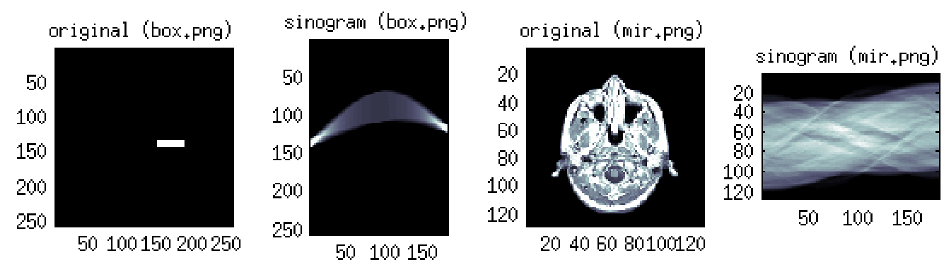

Figure 1: Sinograms of `box.png` and `mir.tif` using 180 projections

Figure (1) above shows a rendition of the corresponding sinogram for the `box.png` and `mir.tif` images, calculated using the algorithm discussed, and the MatLab program given in A.1.1. As can be seen, the image produced has M columns, each of which represent the integral for a given ϕ.

2 Backprojection

The reconstruction of the original image, or in the case of medical imaging, the actual image we wish to create, is called backprojection. It is expressed mathematically, as below

$$f_{\mathrm{BP}}(x, y) = \int_0^{\pi} p(x \cos \phi + y \sin \phi, \phi) \; d\phi \tag{2}$$

As we can see, we are integrating over a hemisphere ($\phi = [0; \pi]$), just as with the projections in producing the sinogram, we mustn't let the angle ϕ surpass 180° — just in radians.

2.1 Reconstruction

By integrating over $p(\xi, \phi)$ in the hemisphere interval we are essentially estimating a reconstruction of the original image from the sinogram, as the ones shown below in figure 2 that we will reconstruct (see figure 3).

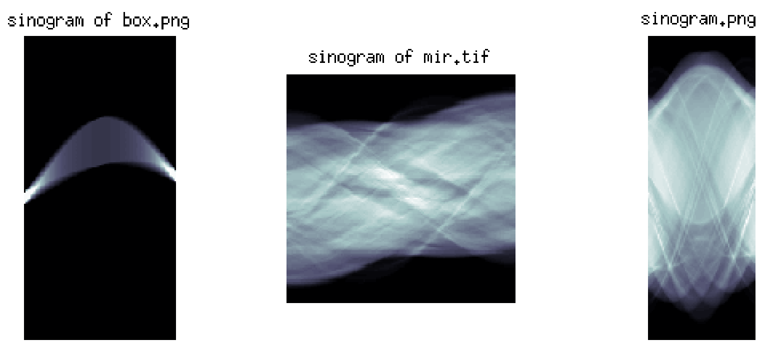

Figure 2: Sinograms to be reconstructed

The implementation of this algorithm can be reviewed in A.3. This algorithm uses a precalculated set of coordinates using `meshgrid` (line 5) in conjunction with a mid-point offset (line 4) to find the indices (lines 10–11) over which we want to integrate (line 17).

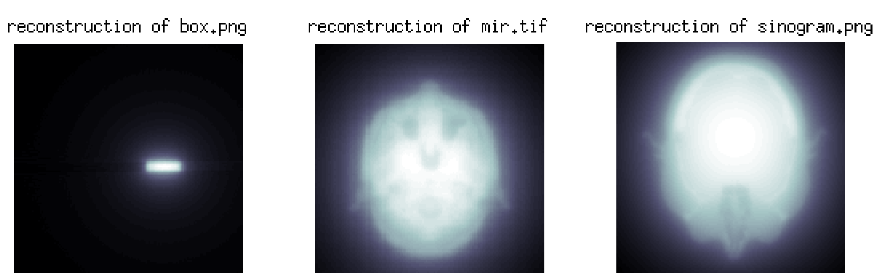

Figure 3: Reconstructed images from sinograms

2.2 Filtered

The algorithm for filtered backprojection remains largely the same as that of ordinary backprojection, although it involves convoluting with a ramp filter for each summation. This has been built into the previously mentioned MatLab implementation of backprojection (see A.3) using an optional filter argument. This makes it extensible, and the ramp filter is simply supplied as an argument in A.1.2.

Figure 4: Reconstructed images from sinograms using filtered backprojection

As we can see by comparing figure 3 with the results of the filtered backprojection, shown above in figure 4, there is a clear improvement in distinguishing the features of the image, as opposed to the latter attempt at reconstructing the images.

2.3 Projections

As we increase the number of projections M used to produce the sinograms, we see a steady increase in clarity of the reconstructed image from its sinogram. In figure 5 below I've chosen to show renditions for very small values of M, as this allows us to observe the individual projection lines very clearly.

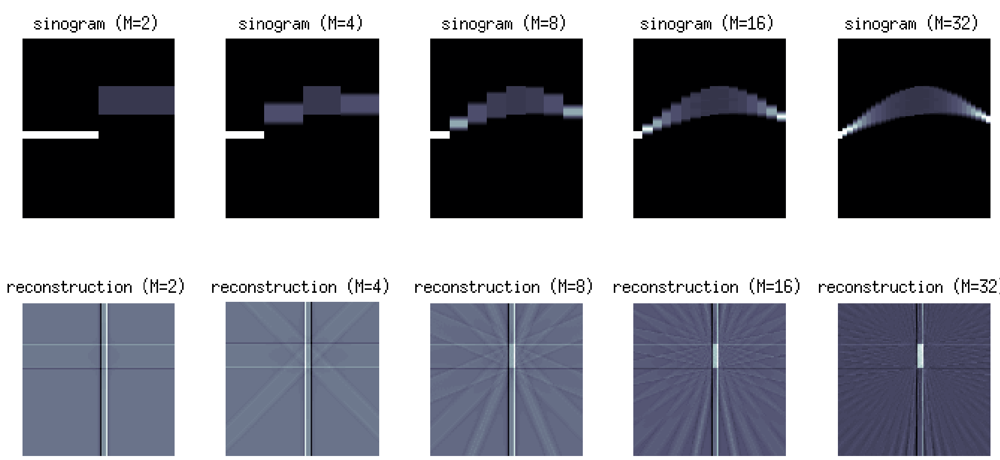

Figure 5: Sinograms and corresponding reconstructions as M increases

It should be noted that for $M > N$ I stop seeing any effect within the reconstructed image.

A Appendix

A.1 Assignments

A.1.1 Sinogram

```
%% assignment 1

I1 = imread('images/box.png');
I2 = imread('images/mir.tif');
R1 = sinogram(I1, 180);
R2 = sinogram(I2, 180);

figure (1), colormap(bone);
subplot(1,4,1), imagesc(I1), axis image, title('original (box.png)');
subplot(1,4,2), imagesc(R1), axis image, title('sinogram (box.png)');
subplot(1,4,3), imagesc(I2), axis image, title('original (mir.png)');
subplot(1,4,4), imagesc(R2), axis image, title('sinogram (mir.png)');
```

Figure 6: Solution for sinogram assignment. (../assignment.m)

A.1.2 Backprojection

```
%% assignment 2

M = 128;

% make sinograms
S1 = sinogram(imread('images/box.png'), M);
S2 = sinogram(imread('images/mir.tif'), M);
S3 = imrotate(imread('images/sinogram.png'), 90, 'bilinear');

% make reconstructions
R1 = imrotate(backprojection(double(S1), M), -90, 'bilinear');
R2 = imrotate(backprojection(double(S2), M), -90, 'bilinear');
R3 = backprojection(double(S3), size(S3,2));

figure (1), colormap(bone);
subplot(1,3,1), imagesc(S1), axis image, axis off, title('sinogram of box.png');
subplot(1,3,2), imagesc(S2), axis image, axis off, title('sinogram of mir.tif');
subplot(1,3,3), imagesc(S3), axis image, axis off, title('sinogram.png');

figure (2), colormap(bone);
subplot(1,3,1), imagesc(R1), axis image, axis off, title('reconstruction of box.png');
subplot(1,3,2), imagesc(R2), axis image, axis off, title('reconstruction of mir.tif');
subplot(1,3,3), imagesc(R3), axis image, axis off, title('reconstruction of sinogram.png');
```

Figure 7: Solution for backprojection (../assignment.m)

A.1.3 Filtered Backprojection

```
%% assignment 3

M = 128;

% make sinograms
S1 = sinogram(imread('images/box.png'), M);
S2 = sinogram(imread('images/mir.tif'), M);
S3 = imrotate(imread('images/sinogram.png'), 90, 'bilinear');

% make reconstructions
F1 = backprojection(double(S1), size(S1,2), [floor(size(S1,1)/2):-1:0 1:ceil(size(S1,1)/2-1)]');
F2 = backprojection(double(S2), size(S2,2), [floor(size(S2,1)/2):-1:0 1:ceil(size(S2,1)/2-1)]');
F3 = backprojection(double(S3), size(S3,2), [floor(size(S3,1)/2):-1:0 1:ceil(size(S3,1)/2-1)]');

figure (1), colormap(bone);
subplot(1,3,1), imagesc(S1), axis image, axis off, title('sinogram of box.png');
subplot(1,3,2), imagesc(S2), axis image, axis off, title('sinogram of mir.tif');
subplot(1,3,3), imagesc(S3), axis image, axis off, title('sinogram.png');

figure (2), colormap(bone);
subplot(1,3,1), imagesc(F1), axis image, axis off, title('reconstruction of box.png');
subplot(1,3,2), imagesc(F2), axis image, axis off, title('reconstruction of mir.tif');
subplot(1,3,3), imagesc(F3), axis image, axis off, title('reconstruction of sinogram.png');
```

Figure 8: Solution for filtered backprojection (../assignment.m)

A.1.4 Projections

```
%% assignment 4

clc, clear all;

img = 'images/box.png';

% plot
S = cell(5);
R = cell(5);
for i = 1:5
    M = 2^i;
    S{i} = sinogram(imread(img), M);
    H = [floor(size(S{i},1)/2):-1:0 1:ceil(size(S{i},1)/2-1)]';
    R{i} = backprojection(double(S{i}), size(S{i},2), H);
end

% plot
colormap('bone');
for i = 1:5
    clear t;
    t = strcat('sinogram (M=', int2str(2^i), ')');
    subplot(2,5,i), imagesc(S{i}), axis off, title(t);
end

for i = 1:5
    clear t;
    t = strcat('reconstruction (M=', int2str(2^i), ')');
    subplot(2,5,i+5), imagesc(R{i}), axis image, axis off, title(t);
end
```

Figure 9: Solution for showing the effects of M, or number of projections (../assignment.m)

A.2 Functions

A.2.1 Sinogram

```
function [ out ] = sinogram( I, M )
    N = size(I, 1);
    out = zeros(N, M);
    for i = 1:M
        phi = 180/M * (i-1);
        tmp = imrotate(I, phi, 'bilinear', 'crop');
        out(:, i) = sum(tmp, 2); % integrate over the projection
    end
end
```

Figure 10: Implementation of the `sinogram` function (../sinogram.m)

A.3 Backprojection

```
function [ out ] = backprojection( I, M, H )
    N = size(I, 1);
    out = zeros(N, N);
    mid = floor(N/2)+1;
    [xs, ys] = meshgrid(-N/2:N/2-1);

    % integrate p(x cos(phi) + y sin(phi)) d(phi) from 0 to pi
    for i = 1:M
        phi = 180/M * (i-1) * (pi/180);
        xi = round(mid + xs * cos(phi) + ys * sin(phi));
        indices = find((xi > 0) & (xi <= N));
        if (exist('H','var'))
            F = real(ifft(ifftshift(H .* fftshift(fft(I(:,i))))));
            out(indices) = out(indices) + F(xi(indices)) ./ M;
        else
            H = true;
            out(indices) = out(indices) + I(xi(indices), i) ./ M;
        end
    end
end
```

Figure 11: Implementation of the `backprojection` function (../backprojection.m)

References

[1] Solomon & Breckon, Fundamentals of Digital Image Processing - A Practical Approach with Examples in Matlab, 2011

[2] Lecture 30 — The Fourier Transforms and its Applications, Stanford University (YouTube channel), `https://www.youtube.com/watch?v=xSUHVojA404`, 2008