Part 1 Direct Fourier transformation, the slice theorem

Image is loaded using imread() function and then transformed to double values by im2double() (Figure 1a). The 30 degree projection is obtained by radon('image file',30) (Figure 1b). This gives the line integral of all t at theta 30. This projection is then Fourier transformed to frequency domain by fft() (Figure 1c). From the Fourier Slice theorem, the Fourier transform of the projection of all t at theta 30 is the slice of the Fourier transform of the original image at the line passing original having 30 degree. In order to extracting all the value at 30 degree from original, I convert a matrix containing cartesian coordinates having exact dimension as the Fourier transformed image to polar coordinates. The values that laying at 30 degree to origin are located (Figure 1e). This vector containing the value of 30 degree Fourier transformation is then inverse transformed to spatial domain (Figure 1f).

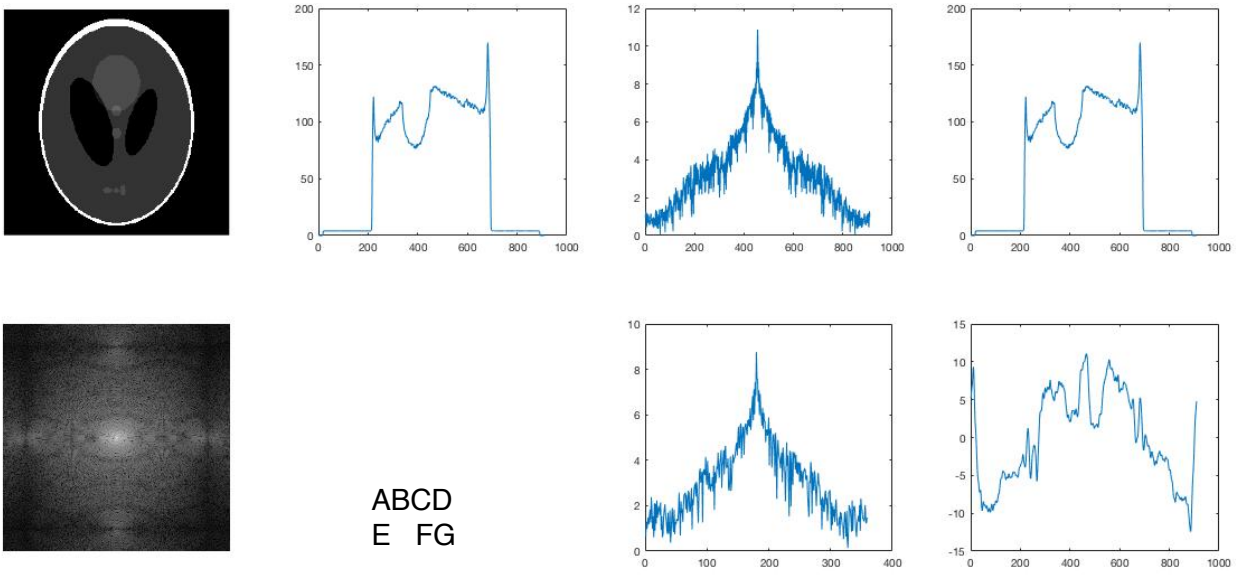
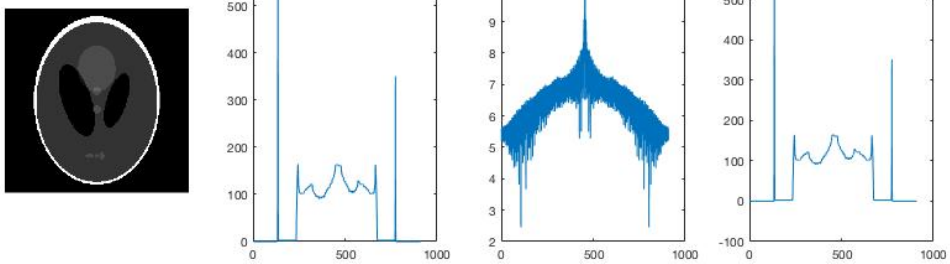


Figure 1: a) original image. b) Radon transform at theta 30. c) Fourier transform of b. d) inverse Fourier transform of c. d) Fourier transform of the original image a. e) the slice at 30 degree. f) inverse transform of e.

(Question 5, 6) Visually from the results, the slice at 30 degree is *almost* the same as the radon transform at 30 degree. However, the inverse transform is not. I think this is due to the lack of information stored in the discrete Fourier transform. In the cartesian coordinate matrix, value at 30 degree can only be estimated by trigonometric and many values are missing. This results in gaps between values and loss of details and precision in spatial domain. However, 0 and 90 degrees line contain all the

information needed without proximation. Figure 2, 0 degree line and figure 3 90 degree line. A perfect reverse transformation can therefore be obtained.



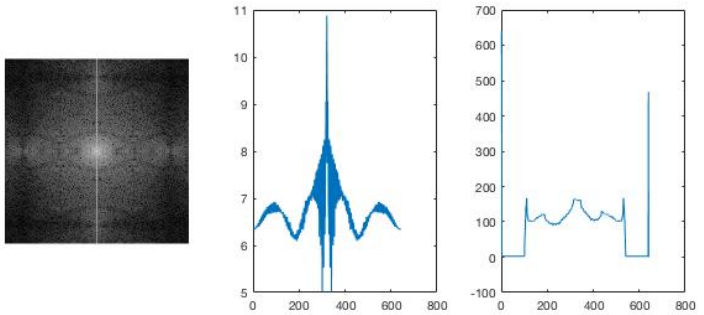
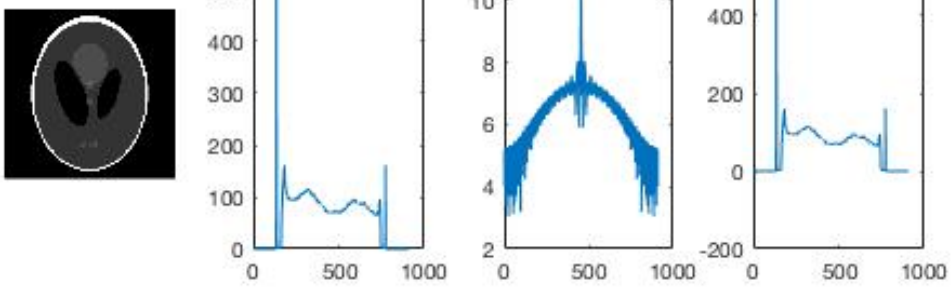


Figure 2: a) original image. b) Radon transform at theta 30. c) Fourier transform of b. d) inverse Fourier transform of c. d) Fourier transform of the original image a. e) the slice at 30 degree. f) inverse transform of e.



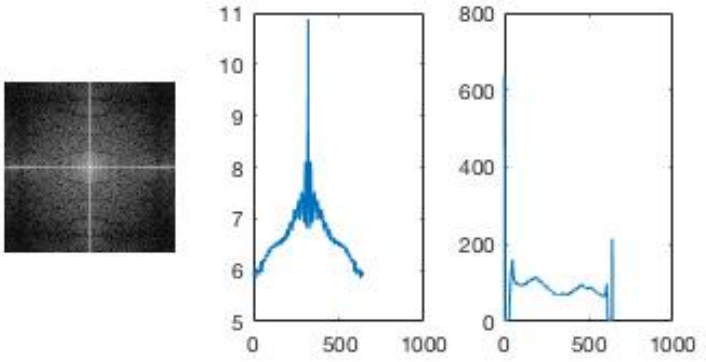
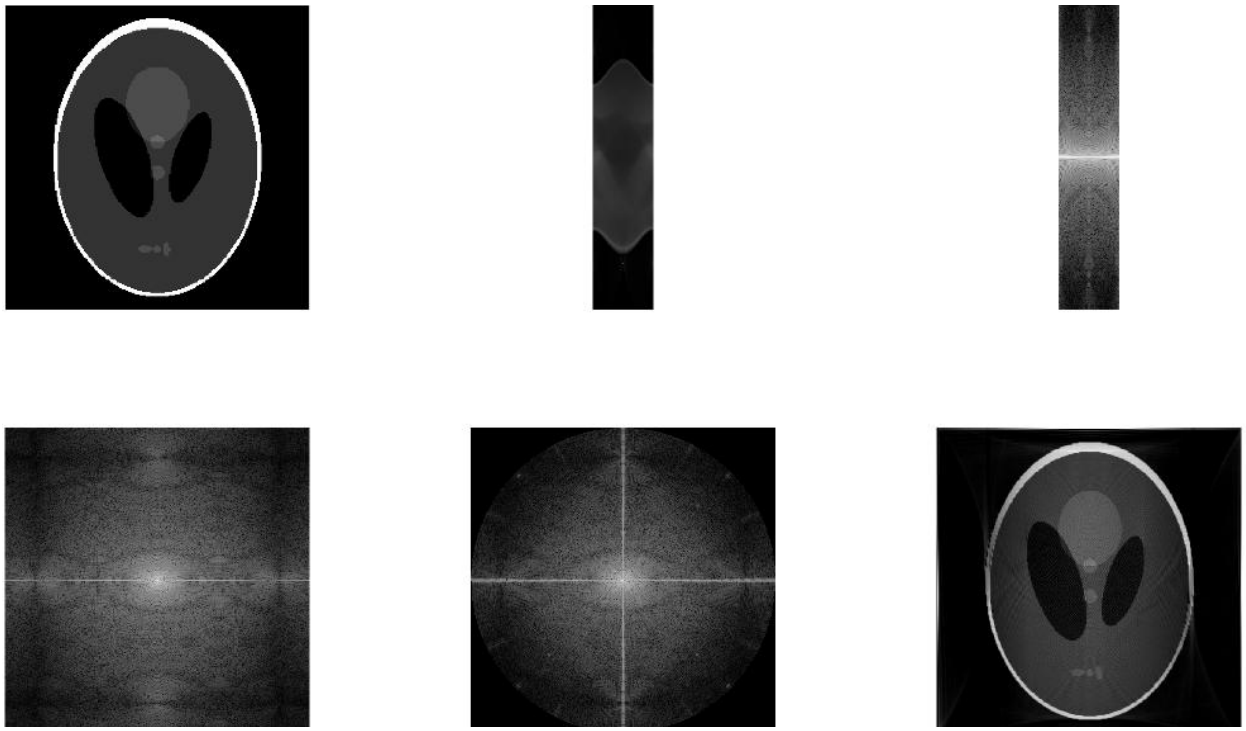


Figure 1: a) original image. b) Radon transform at theta 30. c) Fourier transform of b. d) inverse Fourier transform of c. d) Fourier transform of the original image a. e) the slice at 30 degree. f) inverse transform of e.

Implementation of the Direct Fourier Reconstruction

The radon transform needs to performed for every degree from 0 to 179 degree, since 180 to 359 degree images are just the mirror images of 0 to 179. The image is shifted and 1D Fourier transformed, since each row stores the information of one line t at angle theta. These information is transformed into polar coordinates and interpolated to a 2D Fourier map. A mesh grid of polar coordinates is created ranging from the - radon transform size /2 to + radon transform size /2 and separated by the number of angles. Then, each radon data is fitted into this mesh grid by converting its cartesian coordinates to polar coordinates. As a result, all data will fit in with no overlap. The empty coordinates will be interpolated using interp2() with linear option (Linear is used because, empirically, it gives the best result). the empty space will be filled with 0. This Fourier map is finally inverse transformed to spatial domain.



ABC
DEF

Figure 4: a) Original image, b) radon transformation of a. c) Fourier transformation of b. d)Fourier transformation of the original image. e) Fourier map reconstructed from c. f) direct Fourier reconstruction from e.

Some remarks, first, the reconstructed image has lower sharpness comparing with the original image. This can be explained by comparing with the Fourier transformation of the original image. The corners are not available in the reconstructed Fourier map. Some high frequency information is thus lost. Also, the x and y axis (or 0,90,190 and 270 degrees) are different form the Fourier transform of the original image. I think this is caused by the cartesian to polar coordinates switch.

As show in the figure 5, when theta is multiple of 90 degree, either x or y is equal to 0, the result of atan2 is thus either positive or negative infinity. Some literature suggests to extend the range of radon transform space would solve this problem mathematically, but I still did not understand how this works exactly. Maybe by zero padding the radon transform result?

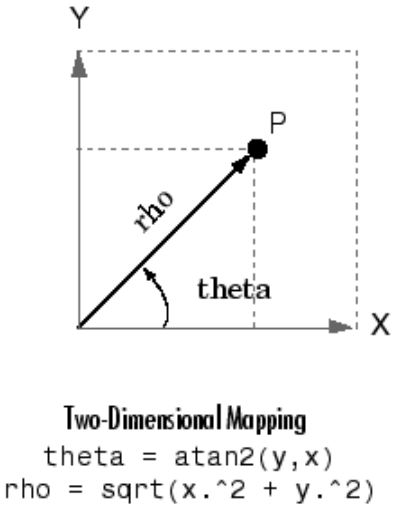


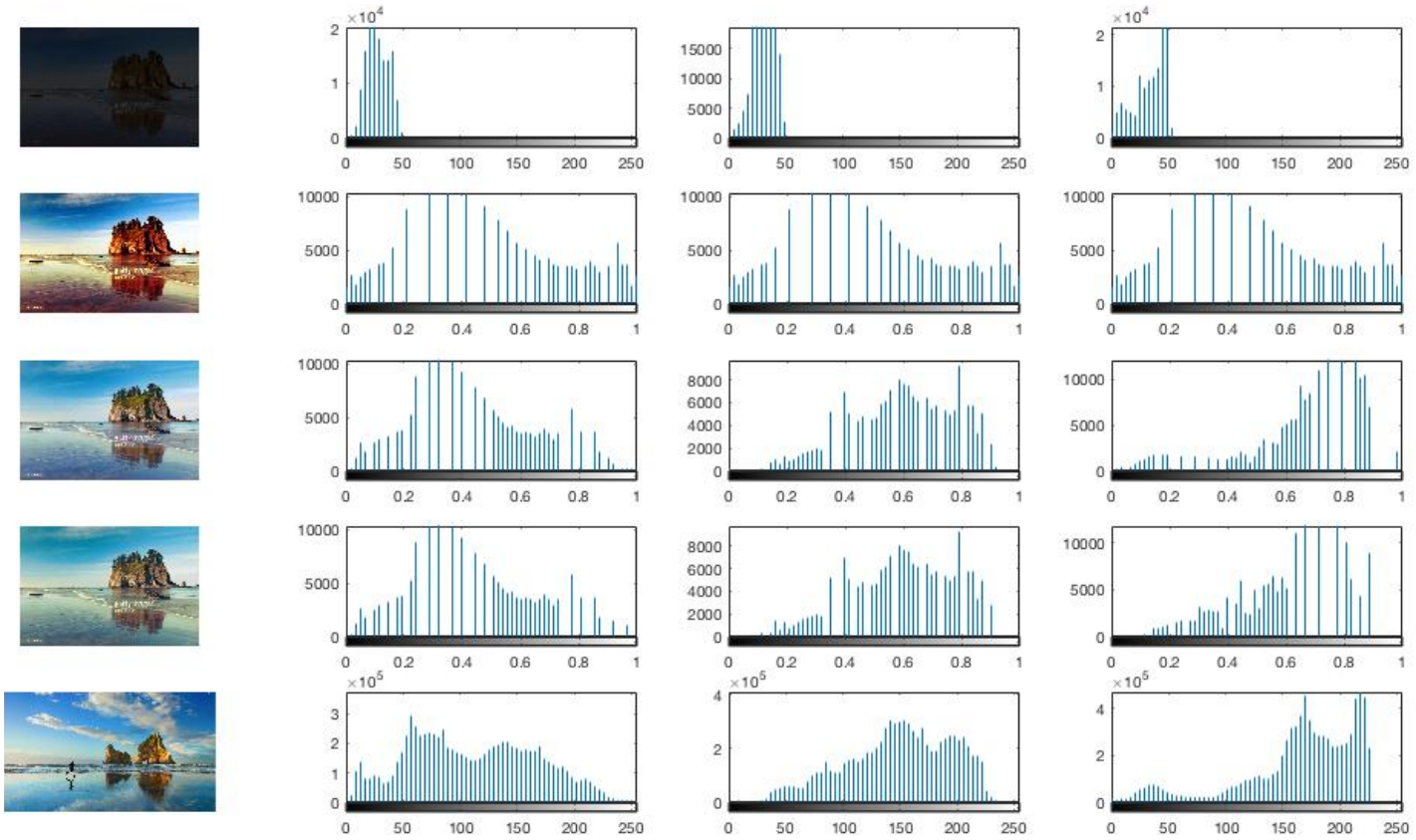
Figure 5, cartesian to polar conversion.(Figure source, matlab cart2pol description page:https://www.mathworks.com/help/matlab/ref/cart2pol.html)

```
function directFourierReconV2(imageFile,Ntheta)
I = im2double(imread(imageFile));%load image to double
% angles of radon to be calculated
theta = 0:Ntheta;
rt = radon(I,theta); % radon T using theta angles;
RTShift = ifftshift(fft(fftshift(rt))); %fourier transform

%define the polar coordinates, center at 0
RTsize = size(rt,1);
singleAngle = 2*pi/RTsize;
%the difference between each angle, this value  depends on the size of the image
t = (-RTsize)/2:(RTsize/2-1);  %coordinates span the size of polar coord
t = t*singleAngle;      %all coordinates in radian unit
%label elements in the matrix RTShift with the corresponding theta and t
[THETA, T]=meshgrid(theta,t); %cartesian coord that currently holding rt
[T_X,T_Y]=meshgrid(t,t); % polar coord that will be given to rt

%creating a meshgrid with all the radian coordinates, ranging from - size/2 to size/2
[THETAp, Tp] = cart2pol(T_X,T_Y);%transform cartesian coord to polar
Tp = Tp.*sign(THETAp); %make sure sign is correctly shown for each quadrant
THETAp = mod(THETAp*(180/pi),180); % get the angle range 0 to 180 with proper sign
% remove the sign confusion caused by atan2(). (or cart2pol())
% reference : Direct Fourier Tomographic Reconstruction Image-to-Image Filter
% Dominique Zosso, Meritxell Bach Cuadra, Jean-Philippe Thiran August 24, 2007

% interpolate
RT2D = interp2(THETA, T, RTShift,THETAp,Tp,'linear',0);
%all point unable to match are given value 0;
Final_image=flipud(ifftshift(ifft2(fftshift(RT2D))));
% image is fliped due to the matlab assign y axis pointing downward.
Final_image=Final_image(135:775,135:775);%crop the image
```

Part 2: Histogram matching

Histogram can be considered as probability density function (pdf) of an image, the cumulative density function (cdf) of this image is calculated from pdf. After calculating the cdf of the images, from cdf of a desired picture, the histogram of the picture of interest can be modified. The pixel intensities in the picture to be modified is matched to the lowest intensity that has the same or closest cdf in the desired histogram. This will give the image i in figure 6. The advantage of matching comparing to histogram equalization is that matching prevents colour alteration shown in Figure 6e.

Another remark is that similar to histogram equalization (Figure 6 f to h) , a perfectly matched histogram cannot be generated due to the fact that some intensity values will be matched to the same value in the desired histogram. This will cause gaps in the modified histogram as shown in Figure 6 j to l and n to p.



ABCD
EFGH
IJKL
MNOP
QRST

Figure 6: a)Original image. e)histogram equalized image a. i)histogram matched to image q without histogram equlaization. m) histogram matched to image q after histogram equalization. q)image whose histogram is to be matched by image a. b) to d), f) to h), j) to k), n) to p), r) to t), are the red/green/blue components of the corresponding image a),e),i),m),q) .

```
function [ImageOut,hist]=histogramMatching(ImageToMod,ImageTarget)
    %load images
    Is = imread(ImageToMod); It = imread(ImageTarget);
    %Cdf of all layers
    ItrCdf=HistCdfLayer(It,1);   ItgCdf=HistCdfLayer(It,2);  ItbCdf=HistCdfLayer(It,3);
    IsrCdf=HistCdfLayer(Is,1); IsgCdf=HistCdfLayer(Is,2); IsbCdf=HistCdfLayer(Is,3);

    Rmatched= histMatch(Is(:,:,1),IsrCdf,ItrCdf);
    Gmatched= histMatch(Is(:,:,2),IsgCdf,ItgCdf);
    Bmatched= histMatch(Is(:,:,3),IsbCdf,ItbCdf);
    Fmatched = im2double(cat(3,Rmatched,Gmatched,Bmatched));

    function [imageMod, newhist] = HistEqual(image,layer,cdf)
        histeq = zeros(1,256);
        for i=1:256
            histeq(i)=round(255*cdf(i));
        end
        imageMod = zeros(size(image,1),size(image,2),'uint8');
        for i=1:size(image,1)
            for j=1:size(image,2);
            imageMod(i,j)=histeq(image(i,j,layer)+1);
            end
        end
        newhist = HistCdfLayer(imageMod,1);
    end
    function histCdf = HistCdfLayer(image, layer)
        %get layer
        Layer = image(:,:,layer);
        %get histogram of each layer from target image
        Layerhist = imhist(Layer);
        %get total pixels
```

```
        total = sum(Layerhist);
        %get cdf of each layer
        histCdf = zeros(1,size(Layerhist,1));
        for i=1:size(Layerhist,1)
            if i==1
                histCdf(i)=Layerhist(i,1)/total;
            else
                histCdf(i)=Layerhist(i,1)/total+histCdf(i-1);
            end
        end
    end
    function [ImageFinalLayer] = histMatch(image,cdfimage,cdfDesired)
        ImageFinalLayer = zeros(size(image,1),size(image,2),'uint8');
        mapping = zeros(256,1);
        for i = 1:256
            for j = mapping(i)+1:256
                if (cdfDesired(j)-cdfimage(i))>=0
                    mapping(i)=j;
                    break;
                end
            end
        end
        for i = 1:size(image,1)
            for j= 1:size(image,2)
                ImageFinalLayer(i,j)=mapping(image(i,j)+1);
            end
        end
    end
end
```