

EE225b – Digital Image Processing
Lab Assignment #1 – Tomography

Overview:

In this assignment, you explore applications of the projection slice theorem to tomography.

Assignment specifics:

On the class website, there is a file `Pyramid.bmp`, 256 pixels wide and 384 pixels tall in 8-bit (256) gray. Using the `radon` function provided by Matlab, generate 180 equally spaced projections of the image. Take a moment to familiarize yourself with the capabilities of `radon` using the help files, and what it is computing for you. The `radon` function treats (192, 128) as the center of the image. For each projection, generate 465 sample points of $p_\theta(t)$ at equally spaced radius, roughly equivalent to one sample point per image unit.

Using the `iradon` function provided by Matlab, which implements the convolution (filtered) back-projection method, restore the original image. Save the reconstructed image as a bitmap file (`ConvBack.bmp`). Take a moment to familiarize yourself with the capabilities of `iradon` using the help files, and what it is computing for you. `iradon` includes a interpolation parameter (default linear), and a filter parameter (default Ram-Lak). How do these parameters affect the convolution back-projection algorithm presented in class and in your textbook (p. 42-45)? Are the defaults reasonable? What happens if you reduce the number of θ samples by a factor of 2? By a factor of 5? Save these reconstructed images as well in bitmap files (`ConvBack2.bmp` `ConvBack5.bmp`).

Implement the polar sampling method using nearest neighbor interpolation in the Fourier domain described in class, and compare it to the output of `iradon`. Save the reconstructed image as a bitmap (`Polar.bmp`).

Please submit a written lab writeup in class on the due date. Also please submit all your `.m` files and `.bmp` files via email to `ncorso@eecs.berkeley.edu`. Email submissions must be received before class on the due date. There should be an executable Matlab script `Lab3.m` which will generate all your results.

Here are some helpful Matlab commands:

<code>X = fft2(x)</code>	Computes the 2D-DFT of the matrix <code>x</code>
<code>x = ifft2(X)</code>	Computes the inverse 2D-DFT of the matrix <code>X</code>
<code>I = uint8(x)</code>	<code>I</code> is a matrix of integers ranging from 0..255
<code>imshow(I)</code>	Displays <code>I</code> as a grayscale image in the current figure
<code>I = imread('small.bmp', 'bmp')</code>	Reads the image file <code>small.bmp</code> and stores it in matrix <code>I</code>
<code>imwrite(I, 'result.bmp', 'bmp')</code>	Writes the matrix <code>I</code> to the image file <code>result.bmp</code>
<code>stuff = load('Phase.dat', '-mat')</code>	Loads the contents of <code>Phase.dat</code> stored in matrix format into the structure <code>stuff</code>
<code>stuff.ImagePhase</code>	Access the <code>ImagePhase</code> matrix stored in <code>stuff</code>