Investigate the behaviour of backprojection reconstruction with and without ramp-filtering, and with Hamming-windowed ramp-filtering

Authors:

```
Oisin Watkins - 15156176
Rezvee Sikder - 15140997
Patrick Lu - 15154149
```

```
import numpy as np
In [23]:
         import matplotlib.pyplot as plt
         import cv2
         import scipy
         from skimage.transform import rotate
         import scipy.fftpack as fft
         from PIL import Image
         from IPython import display
         from scipy.misc import imsave
 In [7]: | file_path = './sinogram.png'
         image = cv2.imread(file path)
         b, g, r = cv2.split(image)
In [8]: def back project(channel, colourChannel):
             print("Back project start")
             RGBChannel = ['Reds', 'Greens', 'Blues']
             laminogram = np.zeros((channel.shape[1],channel.shape[1]))
             # changed from 960 to 1440 (shape 0 -> 1)
             theta = 180.0
             dTheta = theta / channel.shape[0]
             for i in range(channel.shape[0]):
                 temp = np.tile(channel[i],(channel.shape[1],1))
                 temp = rotate(temp, dTheta*i)
                 #plt.imshow(laminogram,cmap=RGBChannel[colourChannel], inte
         rpolation='none')
                 #display.clear output(wait=True)
                 #display.display(plt.show())
                 laminogram += temp
             print("Back project finished")
             return laminogram
```

```
In [9]: def filter_transform(channel):
    # fft translate
    channel = fft.rfft(channel, axis=1)
    # ramp filter
    ramp = np.floor(np.arange(0.5, channel.shape[1]//2 + 0.1, 0.5))
    channel = channel * ramp
    # inverst fft
    channel = fft.irfft(channel, axis=1)
    return channel
In [22]: def filter_transform_hamming(channel):
    #dont know why it's 566
    hamming = np.hamming(len(channel))
    channel = hamming*channel
```

Image cropping

The image dimensions are recovered from the sinogram by taking the number of non zero pixels at 0 degrees and 90 degrees

```
In [11]: # Variables to be used for cropping the image later on
    image_width = int(np.count_nonzero(image[0])/3)
    image_halfway = int((image.shape[0]/2))
    image_height = int(np.count_nonzero(image[image_halfway])/3)
In [12]: def image_crop(image):
    crop_width_start = int((image.shape[0]-1-image_width)/2)
    crop_height_start = int((image.shape[1]-1-image_height)/2)
    return image[crop_height_start:crop_height_start+image_height,
    crop_width_start:crop_width_start+image_width]
```

(a) Reconstruction without ramp-filtering

return channel

As expected, the output image is blurry due to the amount of backprojections



(b) Reconstruction with ramp-filtering

```
In [17]: bFilteredRamp = filter_transform(b)
    gFilteredRamp = filter_transform(g)
    rFilteredRamp = filter_transform(r)

    bp_b = back_project(bFilteredRamp, 2)
    bp_g = back_project(gFilteredRamp, 1)
    bp_r = back_project(rFilteredRamp, 0)

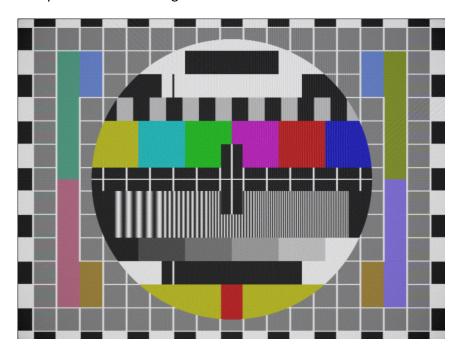
Back project start
    Back project finished
    Back project finished
    Back project start
    Back project start
    Back project finished
    Back project finished
    Back project finished
```

```
In [18]: ramp_filtered = np.dstack((bp_r, bp_g, bp_b))
    ramp_filtered_crop = image_crop(ramp_filtered)
    scipy.misc.imsave("2rampfiltered.png", ramp_filtered_crop)

/home/patrick/.local/lib/python3.6/site-packages/ipykernel_launche
    r.py:3: DeprecationWarning: `imsave` is deprecated!
    `imsave` is deprecated in SciPy 1.0.0, and will be removed in 1.2.
    0.
    Use ``imageio.imwrite`` instead.
    This is separate from the ipykernel package so we can avoid doin
```

This image is much clearer compared to the non filtered image. Very little signs of blurring remain in this image.

Some reconstruction artifacts can be seen however, due to the ramp filter emphasizing high frequency components in the image.



g imports until

(c) Reconstruction using a Hamming-windowed rampfilter

```
In [19]: bp_bHamming = filter_transform_hamming(bp_b)
    bp_gHamming = filter_transform_hamming(bp_g)
    bp_rHamming = filter_transform_hamming(bp_r)
```

```
In [20]: hamming = np.dstack((bp_rHamming, bp_gHamming, bp_bHamming))
    hamming_crop = image_crop(hamming)
    scipy.misc.imsave("3hamming.png", hamming_crop)
```

/home/patrick/.local/lib/python3.6/site-packages/ipykernel_launche r.py:3: DeprecationWarning: `imsave` is deprecated!
 `imsave` is deprecated in SciPy 1.0.0, and will be removed in 1.2.
 0.
 Use `imageio.imwrite` instead.

This is separate from the ipykernel package so we can avoid doin

This is separate from the ipykernel package so we can avoid doin g imports until

Clearest of the three results. Artifacts are reduced by the Hamming filter

