Labsheet 8
Lab-8 Image Restoration

Ragja Palakkadavath
(SC18M003)

Sub: Image and Video Processing Lab
Department: Mathematics(MLC)
Date of Lab sheet: September 27, 2018
Date of Submission: October 3, 2018

1 Wiener Filtering

Perform Wiener Filtering on the motion blurred image of the image 'Book.tif' you got in labsheet 7.

Aim

To apply wiener filtering to a motion blurred image and obtain the restored image

Theory/Discussion

Wiener Filtering is a homogeneous linear image restoration technique.This tool solves an estimate(Least Mean Square Error estimation) for F according to the following equation

$$F(u,v) = \frac{|H(u,v)|^2 * G(u,v)}{|H(u,v)|^2.|H(u,v)| + K)}$$

Algorithm

1. Read The Image
2. Degrade the image using the motion blur filter formula
3. Take fourier transform of the degraded image
4. Apply wiener filtering using the given formula to get the fourier transform of the filtered image
5. Take inverse fourier transform to obtain the original image

Code

```
# -*- coding: utf-8 -*-
"""
Created on Wed Sep 26 14:49:39 2018

@author: IIST
"""
import numpy as np
import math
from PIL import Image
from scipy import fftpack as fftp
```

```
import cmath
from matplotlib import pyplot as plt
def return_degraded_image():
    img = Image.open('C:/Users/IIST/Downloads/Ragja/8_SC18M003_Ragja_IP2018/Book.tif').c
    img.load()
    data = np.asarray( img, dtype="float64" )

    image_fft=fftp.fft2(data)
    a=1
    b=1
    T=1
    H=np.zeros((688,688),dtype="complex")

    for u in range(1,689):
        for v in range(1,689):
            den=math.pi*(u*a+v*b)
            #print(den)
            temp=math.sin(den)
            complexno=1j
            exponent=cmath.exp((-complexno)*den)
            num=T*temp*exponent
            #num=complex(numerator)
            H[u-1,v-1]=num/den
    final_image2=image_fft*H
    image2=np.fft.ifft2(final_image2)
    plt.imsave('2_degraded_image.png',image2.real, cmap='gray')
    return(image2,H)

# -*- coding: utf-8 -*-
"""
Created on Wed Sep 26 18:43:20 2018

@author: IIST
"""

import numpy as np
from PIL import Image
from scipy import fftpack as fftp
from matplotlib import pyplot as plt
from degradation import return_degraded_image
g,H=return_degraded_image()
G=fftp.fft2(g)
img = Image.open('C:/Users/IIST/Downloads/Ragja/8_SC18M003_Ragja_IP2018/2_degraded_image
img.load()
data = np.asarray( img, dtype="float64" )
img0 = Image.open('C:/Users/IIST/Downloads/Ragja/8_SC18M003_Ragja_IP2018/Book.tif').conv
img0.load()

fig, ax = plt.subplots(nrows=1, ncols=3, figsize=(8, 5),
                       sharex=True, sharey=True)
F=np.zeros((688,688),dtype="complex")
K=0
for u in range(1,689):
    for v in range(1,689):
        den=1/H[u-1,v-1]
        temp=(abs(H[u-1,v-1])**2)/((abs(H[u-1,v-1])**2)+K)
        filt=den*temp
        F[u-1,v-1]=filt*G[u-1,v-1]
```

```
image2=np.fft.ifft2(F)
plt.gray()

ax[0].imshow(img0)
ax[0].axis('off')
ax[0].set_title('Original Image')

ax[1].imshow(img)
ax[1].axis('off')
ax[1].set_title('Degraded Image')

ax[2].imshow(image2.real)
ax[2].axis('off')
ax[2].set_title('After Wiener Filtering')


#plt.imshow(deconvolved_img.real, cmap='gray')
#plt.imsave('restored_image.png',deconvolved_img.real, cmap='gray')
```

Result

Figure 1: Image Restoration using Wiener Filtering

Inference

Wiener Filtering was applied on the motion blurred Image. By varying values of K, we were able to obtain different restored images to get the one closest to the original image

2 Radon Transform

Please use the image 'phantom.png'.

1. Find the Radon Transform of the image using inbuilt function.
2. Reconstruct the initial image using the inbuilt function.
3. Explain about projections and Radon Transform

Aim

To obtain the radon transform of an image at different angles and also perform inverse radon transform to obtain the restored image.

Theory/Discussion

The radon function computes projections of an image matrix along specified directions.

A projection of a two-dimensional function f(x,y) is a set of line integrals. The radon function computes the line integrals from multiple sources along parallel paths, or beams, in a certain direction. The beams are spaced 1 pixel unit apart. To represent an image, the radon function takes multiple, parallel-beam projections of the image from different angles by rotating the source around the center of the image. The following figure shows a single projection at a specified rotation angle.

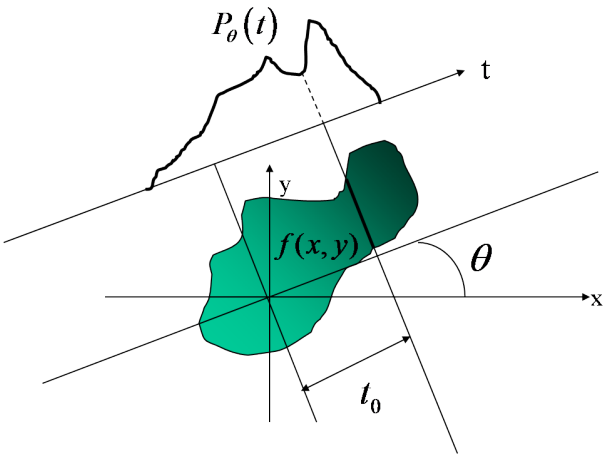

Figure 2: Projection of of an Image to a line

Algorithm

1. Find the Radon Transform of the image using radon function from skimage library
2. Reconstruct the initial image using the inverse radon function.
3. Display both the transform and reconstructed image

Code

```
import matplotlib.pyplot as plt

from skimage.io import imread
from skimage import data_dir
from skimage.transform import radon, iradon
from scipy.ndimage import zoom

image = imread("phantom.png", as_grey=True)
image = zoom(image, 0.4)

plt.figure(figsize=(8, 8.5))

plt.subplot(221)
plt.title("Original");
plt.imshow(image, cmap=plt.cm.Greys_r)

plt.subplot(222)
projections = radon(image, theta=[0, 45, 90])
plt.plot(projections);
plt.title("Projections at\n0, 45 and 90 degrees")
```

```
plt.xlabel("Projection axis");
plt.ylabel("Intensity");

projections = radon(image)
plt.subplot(223)
plt.title("Radon transform\n(Sinogram)");
plt.xlabel("Projection axis");
plt.ylabel("Intensity");
plt.imshow(projections)

reconstruction = iradon(projections)
plt.subplot(224)
plt.title("Reconstruction\nfrom sinogram")
plt.imshow(reconstruction, cmap=plt.cm.Greys_r)

plt.subplots_adjust(hspace=0.4, wspace=0.5)
plt.show()
```

Result

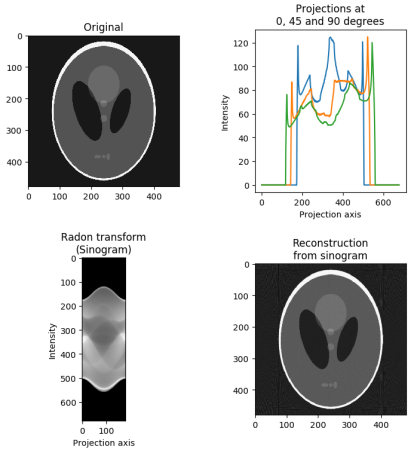

Figure 3: Radon Transform and Restoration

Inference

Radon transform of the phantom image was obtained for different values of angles of projection. The inverse radon transform was performed to restore the image from the obtained sinogram after radon transform

3 Fourier Slice Theorem

Please use the image 'phantom.png'.

1. Read in the image f(x,y)
2. Obtain the Radon Transform of the image p(r,θ)
3. For each θ , p(r) is the projection. Obtain each of its 1-D transform to get the matrix P(ρ, θ).
4. Fill up an empty matrix F(u,v) such that , P(ρ, θ) = F(ρcosθ, (ρsinθ)
5. Take the inverse FFT of F(u,v) to get f(x,y). (f) Display the reconstructed image f(x,y)
6. Find the 2-D FFT of the image and compare it with the F(u,v) you just got. What is the relation between the two?

7. Comment on the results obtained

8. Comment on other methods present for reconstruction.

Aim

To apply Fourier Slice Theorem to obtain the Fourier Transform of an image by finding the 1D Fourier Transform of its radon transforms in different angles

Theory/Discussion

In order to reconstruct the image, we use what is known as the Fourier Slice Theorem. The Slice Theorem tells us that the 1D Fourier Transform of the projection function(which is obtained from the radon transform of the image) P(ρ, θ) is equal to the 2D Fourier Transform of the image evaluated on the line that the projection was taken on (the line that P(ρ, θ) was calculated from). Now, we can simply take the 2D inverse Fourier Transform and have our original image.

Algorithm

1. Read The Image

2. Obtain Radon Transform of the image

3. Take 1D Fourier Transform of each column of the obtained Radon Transform matrix, which is called P(ρ,θ)

4. Fill up an empty matrix F(u,v) such that , P(ρ, θ) = F(ρcosθ, (ρsinθ)

5. Take inverse fourier transform of this result to obtain the restored image

Code

```
# -*- coding: utf-8 -*-
"""
Created on Thu Sep 27 15:07:10 2018

@author: Student
"""

import matplotlib.pyplot as plt
from scipy import fftpack as fftp
from skimage.io import imread
from skimage import data_dir
from skimage.transform import radon, iradon
from scipy.ndimage import zoom
import numpy as np
import math
from scipy.misc import toimage
from PIL import Image
image = Image.open("phantom.png").convert('L')
image = image.resize((180,180))
image.save("phantom_mod.png")
image = imread("phantom.png", as_grey=True)
data = np.matrix( image, dtype="uint8" )
pr = radon(data)
#toimage(pr).show()
F=np.zeros((180,180),dtype="complex")
P=np.zeros((pr.shape[0],pr.shape[1]),dtype="complex")
```

```
#plt.plot(projections);
for theta in range(pr.shape[1]):
    column=pr[:,theta]
    ffted=fftp.fftshift(fftp.fft((column)))
    P[:,theta]=ffted

for rho in range(180):
    for theta in range(180):
        f_cos=abs(round(rho*math.cos(math.radians(theta))))
        f_sin=abs(round(rho*math.sin(math.radians(theta))))
        #print("({},{})".format(f_rho,f_theta))
        F[f_cos,f_sin]=P[rho,theta]
imgreal=np.real(fftp.ifftshift((fftp.ifft2(F))))
maxim=max(imgreal.flatten())
minm=min(imgreal.flatten())
for u in range(180):
    for v in range(180):
        imgreal[u,v] = round(((imgreal[u,v] - minm) / (maxim - minm) ) * (255 - 0) + 0)
#toimage(imgreal.show())
plt.imsave('original_image.png',imgreal)
```

Result

Figure 4: Original Phantom Image

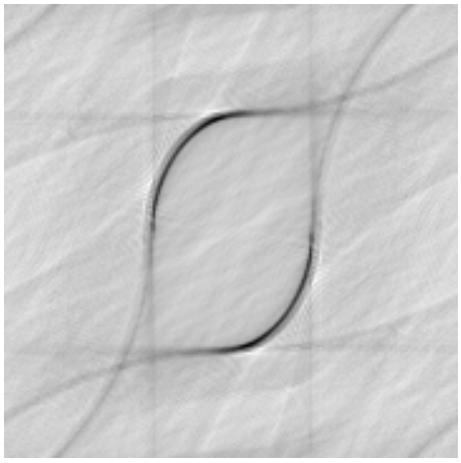

Figure 5: Restored Image with Radon Transform and Fourier Slice Theorem

Inference

The radon transform of the image was taken. 1D Fourier transform of these projections was used to obtain the 2D Fourier transform of the image. Other reconstruction methods are:
Constrained Matrix Inversion, Maximum a posteriori probability estimation, Geometric image restoration(degradation due to error in camera lens)