Rajat Sharma
40196467

Concordia University

COMP 478/6771
Image Processing

Assignment 3

Due Date: November 14, 2023

1. (**8 points**) Prove the validity of the following properties of the Radon transform:

$$g(\rho,\theta)=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\delta(x\cos\theta+y\sin\theta-\rho)\,dx\,dy$$

Show that the Radon transform of the Gaussian shape $f(x,y)=Ae^{(-x^2-y^2)}$ is given by $g(\rho,\theta)=A\sqrt{\pi}e^{-\rho^2}$. (**Hint**: Refer to Example 5.15 in the textbook [see the last page of this document], where we used symmetry to simplify integration, and remember that the integration of a Gaussian distribution function is 1.)

$g(\rho,\theta) = A\sqrt{\pi}\, e^{-\rho^2}$

$f(x,y) = A e^{(-x^2-y^2)}$ given by $g(\rho,\theta) = A\sqrt{\pi}\, e^{-\rho^2}$

$f(x,y)$ is rotationally symmetric i.e. from different angles, projections are same

$\therefore$ projection at $\theta = 0°$

using -(1) at $\theta = 0°$

$$g(\rho,\theta) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\,\delta(x\cos\theta + y\sin\theta - \rho)\,dx\,dy$$

$$= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\,\delta(x-\rho)\,dx\,dy$$

$$= \int_{-\infty}^{\infty} f(\rho,y)\,dy$$

$$= A\int_{-\infty}^{\infty} e^{-(-\rho^2-y^2)}\,dy$$

$$= A\,e^{-\rho^2}\int_{-\infty}^{\infty} e^{-y^2}\,dy \quad -(2)$$

Now we know that

$$\frac{1}{\sqrt{2\pi\sigma^2}}\int_{-\infty}^{\infty} e^{-z^2/2\sigma^2}\,dz = 1$$

$\therefore$ putting $\sigma^2 = 1/2$ then $1/\sqrt{\pi}\int_{-\infty}^{\infty} e^{-z^2}\,dz = 1$

$$\int_{-\infty}^{\infty} e^{-y^2}\,dy = \sqrt{\pi} \quad -(3)$$

putting eqn (3) in eqn (2) $\therefore$ $g(\rho,\theta) = A e^{-\rho^2}\sqrt{\pi}$

$\Rightarrow A\, e^{-\rho^2}\sqrt{\pi}$

2. (**10 points**) Given a 3x3 spatial mask that performs image blurring as below:

$$h = \frac{1}{6}\begin{bmatrix} 0 & 1 & 0 \\ 2 & 0 & 2 \\ 0 & 1 & 0 \end{bmatrix}$$

Find the equivalent filter *H(u,v)* in the frequency domain (use the coordinate system defined in the lecture). For this question, please use Property 3 of Table 4.4 in the textbook, which summarizes the property of spatial translation for an image:

3)	Translation (general)	$f(x,y)e^{j2\pi(u_0x/M + v_0y/N)} \Leftrightarrow F(u-u_0, v-v_0)$ $f(x-x_0, y-y_0) \Leftrightarrow F(u,v)e^{-j2\pi(ux_0/M + vy_0/N)}$

Denoting spatial mask as $g(x,y)$ with fourier transform $H(u,v)$.

Mask representation :-

$$g(x,y) = \frac{1}{4}[f(x,y+1) + f(x+1,y) + f(x-1,y) + f(x,y-1)]$$

From translation property given :-

$$f(x-x_0, y-y_0) = F(u,v)\, e^{-j2\pi(ux_0/M + vy_0/N)}$$

$$H(u,v) = \frac{1}{4} F(u,v)\left[e^{j2\pi v/N} + e^{j2\pi u/M} + e^{-j2\pi u/M} + e^{-j2\pi v/N}\right]$$

$$= \frac{1}{4} F(u,v) * 4\, H(u,v)$$

$$G(u,v) = F(u,v) \cdot H(u,v)$$

$$H(u,v) = \frac{1}{2}\left[\cos\left(\frac{2\pi u}{M}\right) + \cos\left(\frac{2\pi v}{N}\right)\right]$$

⇒ filter transfer function in frequency domain

3. (**6 points**) Simple linear motion blur of an image $f(x,y)$ can be modelled as integration of linear shifts over time T: $g(x,y) = \int_0^T f[x - x_0(t), y - y_0(t)]dt$, where $g(x,y)$ is the blurred image. Thus, the Fourier transform of the filter function is defined as

$$H(u,v) = \int_0^T e^{-j2\pi[ux_0(t) + vy_0(t)]}\, dt \qquad (5\text{-}74)$$

For a particular motion blur, we have:

$$x_0(t) = \begin{cases} \frac{at}{T_1} & 0 \le t \le T_1 \\ a & T_1 < t \le T_1 + T_2 \end{cases} \quad \text{and} \quad y_0(t) = \begin{cases} 0 & 0 \le t \le T_1 \\ \frac{b(t-T_1)}{T_1} & T_1 < t \le T_1 + T_2 \end{cases}$$

By using Equation 5-74, please derive the corresponding transfer function H(u,v) in the Fourier domain.

$g(x,y) = \int_0^T f[x - x_0(t), y - y_0(t)]\, dt$

Given,

$x_0(t) = \begin{cases} \frac{at}{T_1}, & 0 \le t \le T_1 \\ a, & T_1 < t \le T_1 + T_2 \end{cases}$ — ①

$y_0(t) = \begin{cases} 0 & 0 \le t \le T_1 \\ \frac{b(t-T_1)}{T_1} & T_1 < t \le T_1 + T_2 \end{cases}$ — ②

$H(u,v) = \int_0^T e^{-j2\pi[ux_0(t) + vy_0(t)]}\, dt$ — ③

Substituting values of ① & ② in ③

$H(u,v) = \int_0^{T_1} e^{-j2\pi[ua\, t/T_1]} dt + \int_{T_1}^{T_1+T_2} e^{-j2\pi[ua + vb(t-T_1)/T_2]} dt$

$H(u,v) = \frac{T_1}{\pi ua} \sin(\pi ua)\, e^{-j\pi ua} + e^{-j2\pi ua} \int_{T_1}^{T_1+T_2} e^{-j2\pi vb(t-T_1)/T_2} dt$

putting $t - T_1 = z$

$= \frac{T_1}{\pi ua} \sin(\pi ua)\, e^{-j\pi ua} + e^{-j2\pi ua} \int_{T_1}^{T_1+T_2} e^{-j2\pi vbz/T_2} dz$

∴ the blurred image $= g(x,y) = [H(u,v) \cdot F(u,v)]^{-1}$

Part II: Programming questions

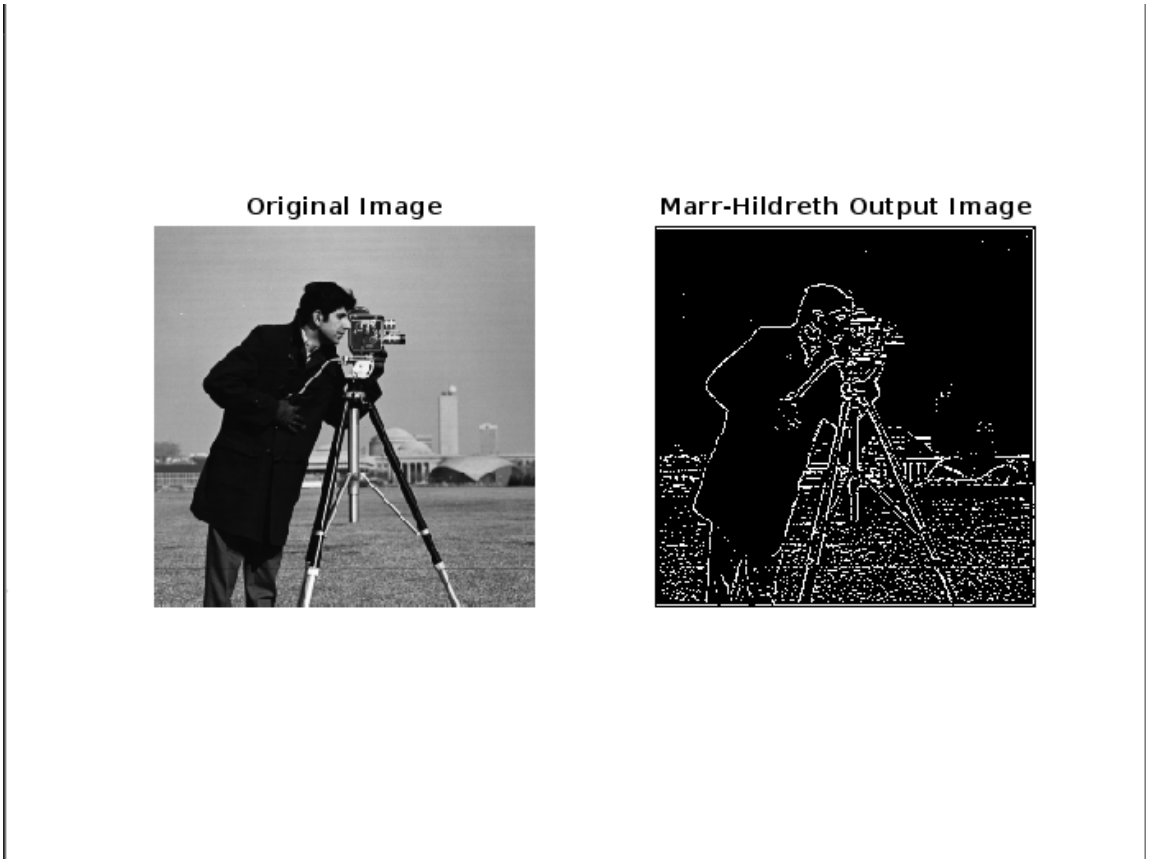
1. (**18 points**) Download the image "*cameraman.tif*" from the assignment package then perform edge detection using existing MATLAB functions (with the parameter choices of your own) for:
   a) Laplacian of Gaussian (Marr-Hildreth) edge detector
   b) Canny edge detector

   1) (**4 points**) Briefly list the steps involved in implementing the edge detectors.
   2) (**4 points**) Explain how edge linking (the final step of the **Canny algorithm**) was implemented. Does the first method need this step?
   3) (**4 points**) List the parameters that determine the performance of the algorithms. What parameter values did you use and why?
   4) (**6 points**) Show and compare the results obtained by the two methods (give some comments).

mhdetector.h

```
% Clear workspace
clear all
% Read the original image
im = imread('cameraman.tif');
im = im2double(im);
% Smoothen the image with a filter (Marr-Hildreth Edge Detector)
gfilter = [0 0 1 0 0;
           0 1 2 1 0;
           1 2 -16 2 1;
           0 1 2 1 0;
           0 0 1 0 0];
smim = conv2(im, gfilter);
% Finding zero crossings
[rr, cc] = size(smim);
zc = zeros([rr, cc]);
for i = 2:rr-1
   for j = 2:cc-1
       if (smim(i,j) > 0)
           if (smim(i,j+1) >= 0 && smim(i,j-1) < 0) || (smim(i,j+1) < 0 &&
smim(i,j-1) >= 0)
               zc(i,j) = smim(i,j+1);
           elseif (smim(i+1,j) >= 0 && smim(i-1,j) < 0) || (smim(i+1,j) < 0
&& smim(i-1,j) >= 0)
               zc(i,j) = smim(i,j+1);
           elseif (smim(i+1,j+1) >= 0 && smim(i-1,j-1) < 0) ||
(smim(i+1,j+1) < 0 && smim(i-1,j-1) >= 0)
               zc(i,j) = smim(i,j+1);
           elseif (smim(i-1,j+1) >= 0 && smim(i+1,j-1) < 0) ||
(smim(i-1,j+1) < 0 && smim(i+1,j-1) >= 0)
```

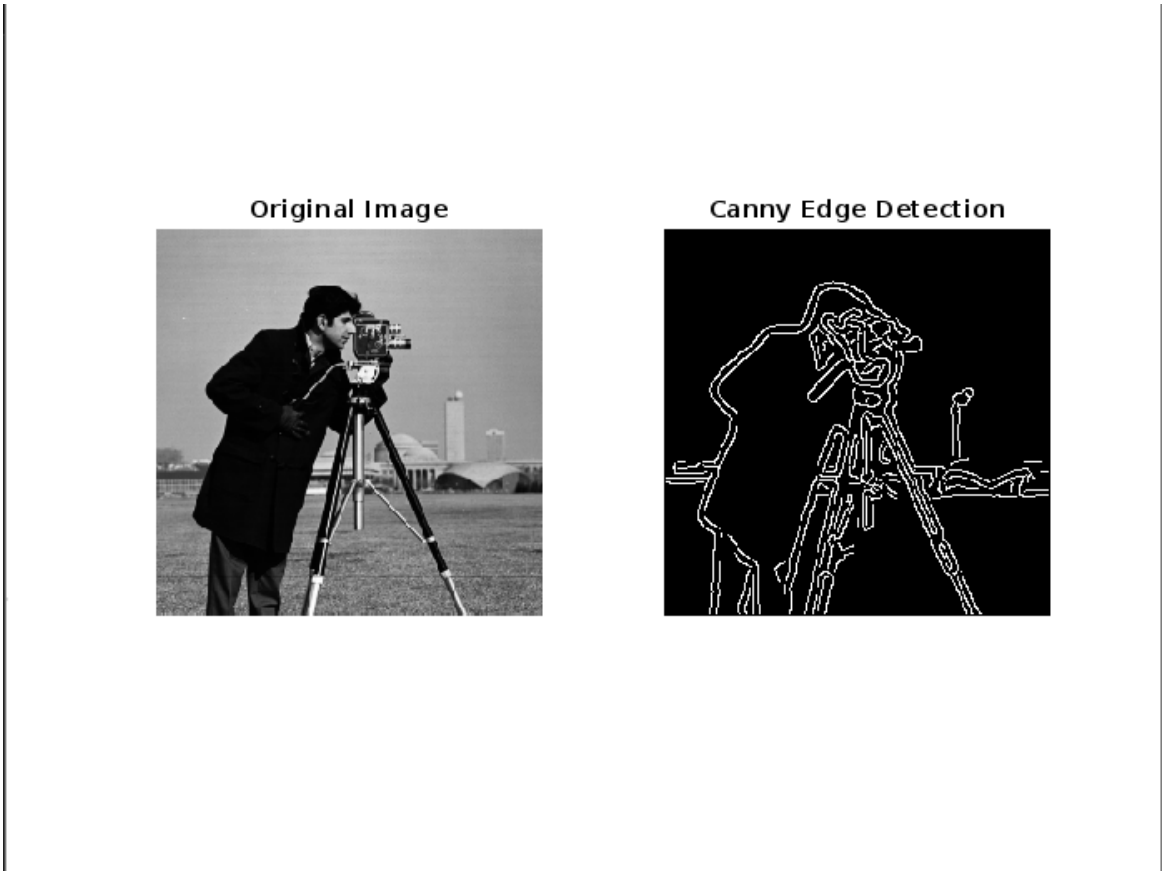
```
                zc(i,j) = smim(i,j+1);
            end
        end
    end
end
% Convert the result to uint8
otpt = im2uint8(zc);
% Thresholding
otptth = otpt > 105;
% Display only the original and final images
figure;
subplot(1,2,1); imshow(im); title('Original Image');
subplot(1,2,2); imshow(otptth); title('Marr-Hildreth Output Image');
```



canny.m

```
% Read the image
originalImage = imread('cameraman.tif');
% Convert the image to double for processing
originalImage = double(originalImage);
% Step 1: Apply Gaussian smoothing
sigma = 1.5; % Standard deviation of the Gaussian filter
smoothedImage = imgaussfilt(originalImage, sigma);
% Step 2: Calculate gradient magnitude and direction
```

```
[gradientMagnitude, gradientDirection] = imgradient(smoothedImage,
'Sobel');
% Step 3: Non-Maximum Suppression
nmsImage = imhmax(gradientMagnitude, 20); % Using a hysteresis value of 20
% Step 4: Edge Tracking by Hysteresis
lowThreshold = 0.1; % Adjust based on the image characteristics
highThreshold = 0.2; % Adjust based on the image characteristics
edgeImage = edge(nmsImage, 'Canny', [lowThreshold highThreshold]);
% Display the results
figure;
subplot(1, 2, 1), imshow(originalImage, []), title('Original Image');
subplot(1, 2, 2), imshow(edgeImage), title('Canny Edge Detection');
```



1) Steps Involved in Implementing Edge Detectors:

a) Laplacian of Gaussian (Marr-Hildreth) Edge Detector:

Step 1: Read the image.
Step 2: Convert the image to double for processing.
Step 3: Apply Gaussian smoothing.
Step 4: Calculate the Laplacian of the smoothed image.
Step 5: Find zero-crossings in the Laplacian image.

Step 6: Thresholding.
b) Canny Edge Detector:

Step 1: Read the image.
Step 2: Convert the image to double for processing.
Step 3: Apply Gaussian smoothing.
Step 4: Calculate gradient magnitude and direction.
Step 5: Non-Maximum Suppression.
Step 6: Edge Tracking by Hysteresis.

2) Edge Linking in Canny Algorithm:

Edge linking is the final step in the Canny algorithm, where edges are connected to form continuous curves. This is achieved through hysteresis thresholding. Pixels above the high threshold are immediately considered as strong edges. Pixels below the low threshold are discarded. Pixels between the low and high thresholds are considered weak edges if they are connected to strong edges. This is done by tracing along the gradient direction.

The Marr-Hildreth method does not explicitly have an edge linking step. The zero-crossings in the Laplacian image serve as edges.

3) Parameters and Their Values:

Marr-Hildreth (Laplacian of Gaussian):

Sigma (Standard Deviation of Gaussian): Chosen based on the level of smoothing required. Higher values result in more smoothing.
Threshold for Zero-Crossings: Determined based on the characteristics of the image.

Canny Edge Detector:

Sigma (Standard Deviation of Gaussian): Chosen for Gaussian smoothing. Higher values result in more smoothing.
Low and High Thresholds: Adjusted based on the image characteristics. Low and high thresholds determine which edges are weak, strong, or discarded during hysteresis.

4) Results and Comparison:

The actual results will depend on the specific parameter values chosen, and the characteristics of the input image. Generally:

Canny tends to perform well due to its multi-stage approach and the ability to handle noise.
Marr-Hildreth might be sensitive to the choice of the threshold for zero-crossings and may require more tuning.

2. (**8 points**) Please continue to use the same image as Q1, "*cameraman.tif*". In class, we have learnt the filter function of motion blur in Fourier domain as (see Page 356 of the textbook):

$$H(u,v) = \frac{T}{\pi(ua + vb)} \sin[\pi(ua + vb)] e^{-j\pi(ua + vb)} \qquad (5\text{-}77)$$

To generate a discrete filter transfer function of size $M \times N$, we sample this equation for $u = 0, 1, 2, \ldots, M-1$ and $v = 0, 1, 2, \ldots, N-1$.

Simulate motion blur for the input image with the parameters of T=1, a=0.1, and b=0.1. Please showcase the images before and after the motion degradation, as well as the magnitude image of the filter function in the Fourier domain.

```
im = imread('cameraman.tif');
[m, n] = size(im);
[u, v] = meshgrid(1:m, 1:n);
T = 1;
a = 0.1;
b = 0.1;
X = pi * (u*a + v*b);
Hmb = (T./X) .* sin(X) .* exp(-1j * X);
MB = ifft2(fft2(im) .* Hmb);
% Display the results
figure;
subplot(1, 2, 1), imshow(im, []), title('Original Image');
subplot(1, 2, 2), imshow(log(abs(MB) + 1), []), title('Motion Blurred
Image');
```