COMP 478 - Image Processing

Assignment 03

Name	ID
Vaansh Vikas Lakhwara	40114764

Part I: Theoretical questions

1. Let center of the resulting 3×3 image in the spatial domain be represented by $f(x, y)$.

$\therefore$ The four closest neighbours are $f(x+1, y)$, $f(x-1, y)$, $f(x, y+1)$ & $f(x, y-1)$.

Filter in spatial domain:

$$h(x, y) = \frac{1}{4}\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

The resulting image in the spatial domain:

$$\begin{aligned} g(x, y) &= f(x, y) * h(x, y) \\ &= \frac{1}{4}(f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1)) \end{aligned}$$

Using the translation property:

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

and by Euler's Equation:

$$e^{ix} = cos(x) + i \cdot sin(x)$$

The resulting image in the frequency domain:

$$\begin{aligned} G(u, v) &= F(u, v) \cdot \frac{1}{4}(e^{\frac{j2\pi u}{M}} + e^{\frac{-j2\pi u}{M}} + e^{\frac{-j2\pi v}{N}} + e^{\frac{j2\pi v}{N}}) \\ &= F(u, v) \cdot \frac{1}{4}(2 \cdot cos(\frac{2\pi u}{M}) + \cancel{j \cdot sin(\frac{2\pi u}{M})} - \cancel{j \cdot sin(\frac{2\pi u}{M})} + 2 \cdot cos(\frac{2\pi v}{N}) + \cancel{j \cdot sin(\frac{2\pi v}{N})} - \cancel{j \cdot sin(\frac{2\pi v}{N})}) \\ &= F(u, v) \cdot \frac{1}{2}(cos(\frac{2\pi u}{M}) + cos(\frac{2\pi v}{N})) \\ &= F(u, v) \cdot H(u, v) \end{aligned}$$

Which means our filter in frequency domain is given by:

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

2. Radon transformation:

$$\mathcal{R}f \equiv g(\rho,\theta) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\delta(x\cos\theta + y\sin\theta - \rho)dxdy \tag{1}$$

a) To prove its linearity it is sufficient to prove that

$$\mathcal{R}(af_1 + bf_2) = a\mathcal{R}f_1 + b\mathcal{R}f_2$$

Plugging into Equation (1)

$$\begin{aligned}\mathcal{R}(af_1 + bf_2) &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} (af_1(x,y) + bf_2(x,y))\delta(x\cos\theta + y\sin\theta - \rho)dxdy \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} af_1(x,y)\delta(x\cos\theta + y\sin\theta - \rho)dxdy + \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} bf_2(x,y)\delta(x\cos\theta + y\sin\theta - \rho)dxdy \\ &= ag_1(\rho,\theta) + bg_2(\rho,\theta) \equiv a\mathcal{R}f_1 + b\mathcal{R}f_2\end{aligned}$$

$\square$

b) To prove its translation property it is sufficient to prove that

$$\mathcal{R}f(x - x_0, y - y_0) \equiv g(\rho - x_0\cos\theta - y_0\sin\theta, \theta)$$

Plugging into Equation (1)

$$\begin{aligned}\mathcal{R}f(x - x_0, y - y_0) &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\delta((x - x_0)\cdot\cos\theta + (y - y_0)\cdot\sin\theta - \rho)dxdy \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\delta(x\cos\theta + y\sin\theta - \rho - x_0\cos\theta - y_0\sin\theta)dxdy \\ &\equiv g(\rho - x_0\cos\theta - y_0\sin\theta, \theta)\end{aligned}$$

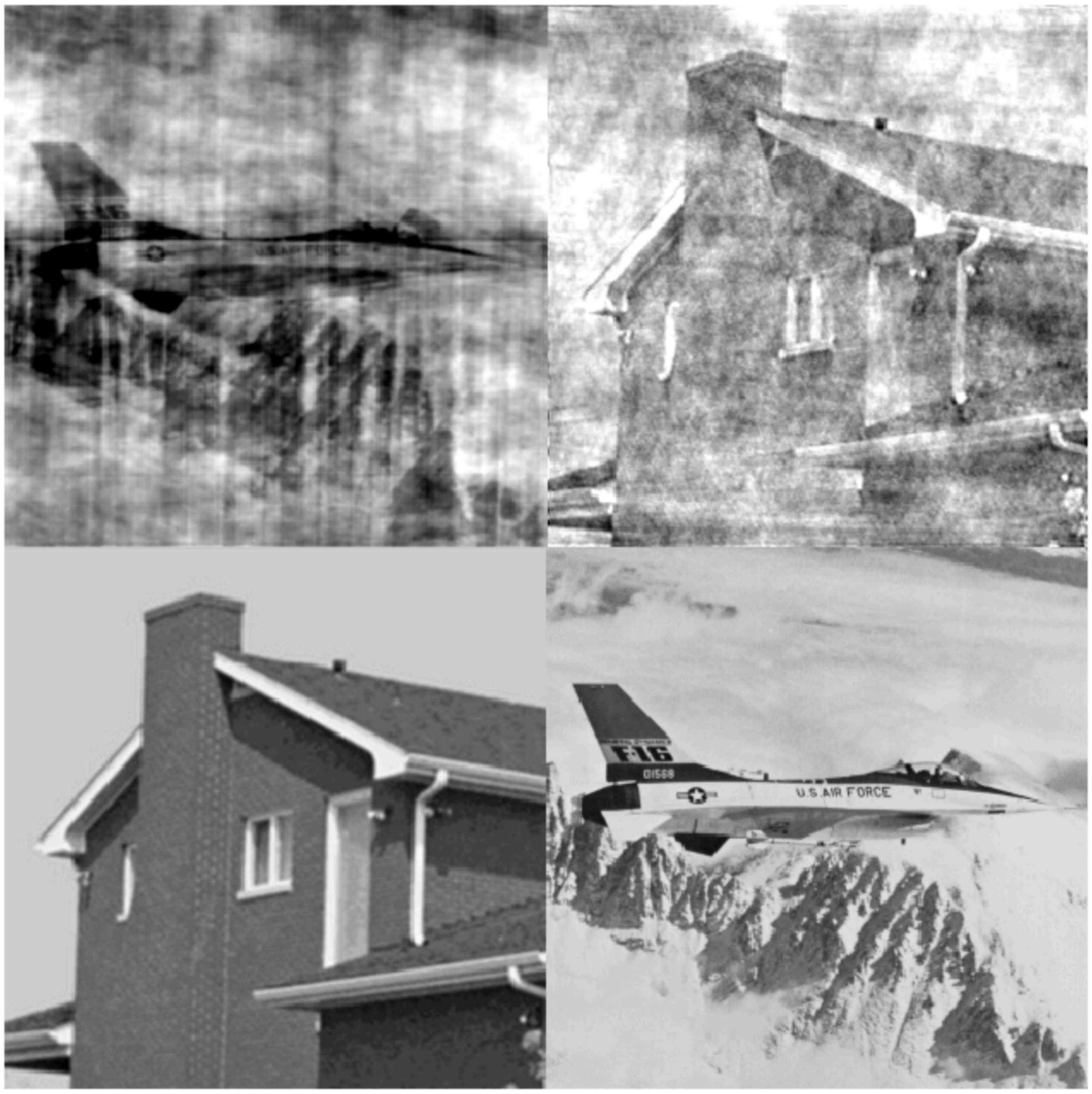
$\square$

Part II: Programming questions

1. Consider I_3 and I_4 – reconstructed with the correct corresponding magnitude $|F|$ and phase angles Ω.

Clearly I_2 is a better reconstruction of I_3 than I_1. This is because I_2 and I_3 have the same phase angle Ω_A which signifies the shape/structure/details of the image.

Top: I1 (Fa & Ob) vs I2 (Fb & Oa)
Bottom: I3 (Fa & Oa) vs I4 (Fb & Ob)



Code:

```
% Load and process images
H = imread("house.tif");
H = H(:, :, 1);
H = im2double(H);

J = imread("jet.tif");
J = J(:, :, 1);
J = im2double(J);

% Separate images into magnitude
% & phase angle using 2D fft
Hfft = fft2(H);
Fa = abs(Hfft);
Oa = angle(Hfft);

Jfft = fft2(J);
Fb = abs(Jfft);
Ob = angle(Jfft);

% Construct images as required
Ifd1 = Fa .* exp(j * Ob);
I1 = ifft2(Ifd1);

Ifd2 = Fb .* exp(j * Oa);
I2 = ifft2(Ifd2);

Ifd3 = Fa .* exp(j * Oa);
I3 = ifft2(Ifd3);

Ifd4 = Fb .* exp(j * Ob);
I4 = ifft2(Ifd4);

% Output images
montage({I1, I2, I3, I4});
title({"Top: I1 (Fa & Ob) vs I2 (Fb & Oa)",
    "Bottom: I3 (Fa & Oa) vs I4 (Fb & Ob)"})
```

2.

(1) Steps involed:

a. **Laplacian of Gaussian (Marr-Hildreth) edge detector**
    1. Smoothing the image with a Gaussian lowpass filter.
    2. Find Laplacian of the resulting image with a Laplacian mask.
    3. Finding zero crossings of the image.
b. **Canny edge detector**
    1. Smoothing the image with a Gaussian lowpass filter.
    2. Using Sobel operator to determine the edge magnitude.
    3. Using Sobel operator to determine the edge direction.
    4. Relate edge directions by grouping them into discrete bins.
    5. Suppress non-edge pixels i.e, non-maximum suppression.
    6. Link the edges together.

(2) Final step of edge linking results in an image with strong edges and the connected weak edges. Implementation in Canny algorithm:

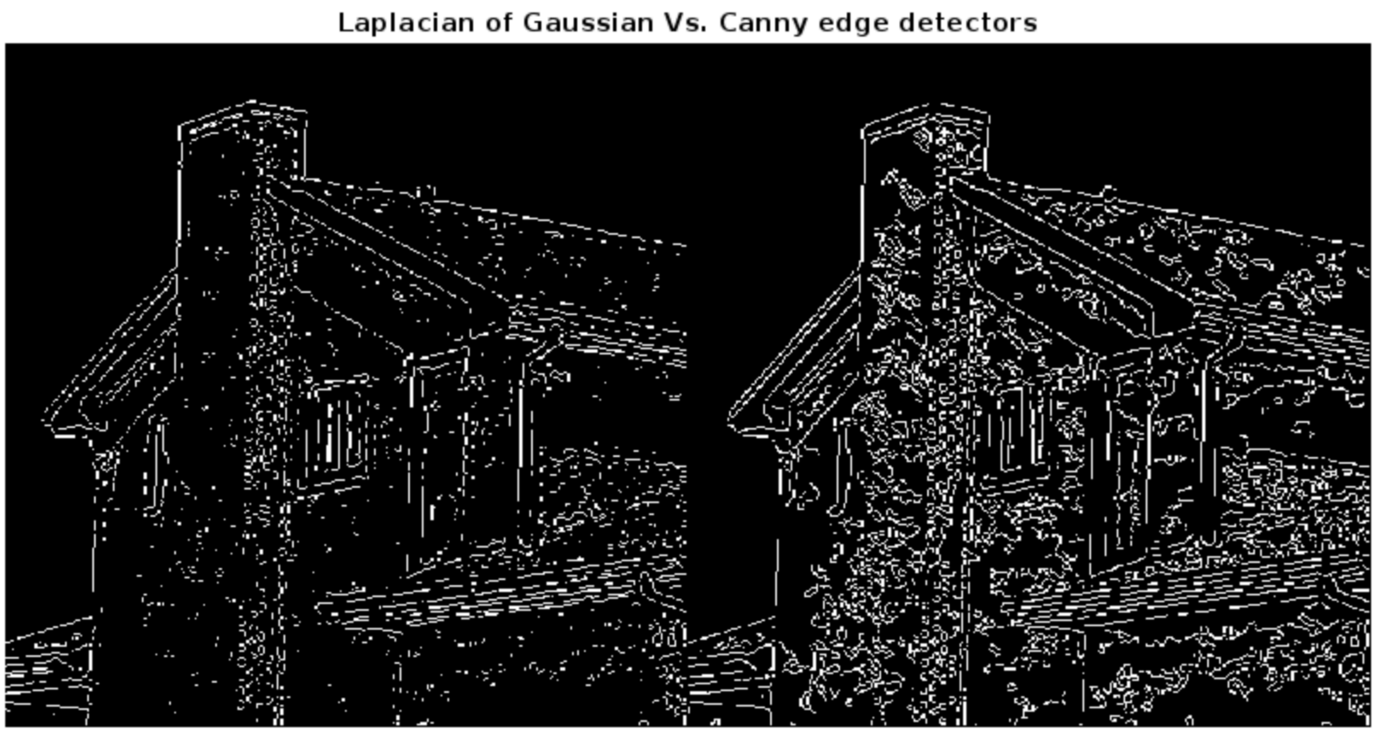
1. Define high (T_1) and low (T_2) thresholds.
2. If the value is greater than the high threshold or it is between the low and high thresholds ($T_1 \leq$ value i.e, start of an edge or $T_2 \leq$ value $< T_1$ i.e, continuation of the edge) it is an edge point, otherwise it is not an edge point. Use this to eventually connect the weak and strong edges.

No, the first method does not need this step as we do not find the direction of edges since a Laplacian filter is used.

(3) Parameters:

1. The Gaussian kernel spread/size represented by σ. For both LoG and Canny detectors I used the default `edge` method values for σ i.e, 2 for the standard deviation of the Laplacian of Gaussian filter and $\sqrt{2}$ for the standard deviation of the Gaussian filter respectively since the results below were satisfactory with these values. A small value detects the finer features whereas a large value detects larger scale edges.
2. The low and high threshold for the Canny. This determines the final step of edge linking as described in the answer to sub-question (2). I again went with the default `edge` method values for the threshold (representing the `[low, high]` thresholds that is $[0, 1]$ calling the `approxcanny` method) in Matlab since the results were satisfactory.

(4)



Note: `lhs` represents LoG edge detector/result and `rhs` represents Canny edge detector/result below.

Comments:

1. Overall rhs does a better job than lhs at detecting edges.
2. rhs detects the weak edges that are connected to the strong edges whereas lhs does not.
3. lhs is noise-sensitive while rhs is less affected by noise which can be attributed to point number 2.

Code:

```
I = imread("img/house.tif");
I = I(:, :, 1);

L = edge(I, "LoG");
C = edge(I, "Canny");

montage({L, C})
title("Laplacian of Gaussian Vs. Canny edge detectors")
```