Class Lab 5

Use two images for each operation to do the following operations and write down their advantages and disadvantages and explain your results:

1. 2D DFT (lena, bridge, rectangle):

Algorithm:

$$F(u,v) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)e^{-j2\pi(ux/M + vy/N)}$$

1. Image centralization
2. Use the Fourier transform
3. Calculate the frequency spectrum

$$|F(u,v)| = \left[R^2(u,v) + I^2(u,v)\right]^{1/2}$$

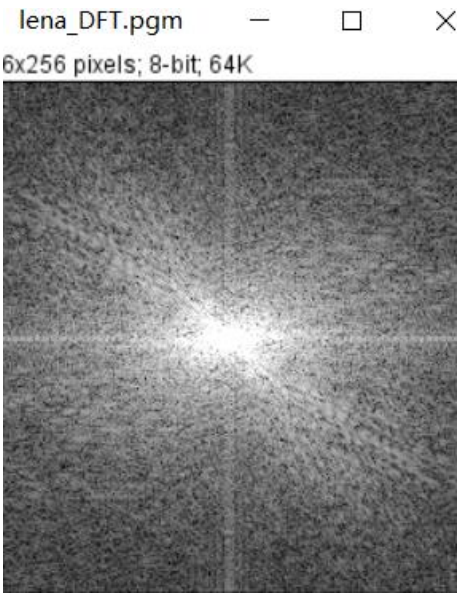
4. Normalized processing
5. Display using logarithmic transformation

Results (including pictures):

Result of processing "Lena.pgm":

Source Image:

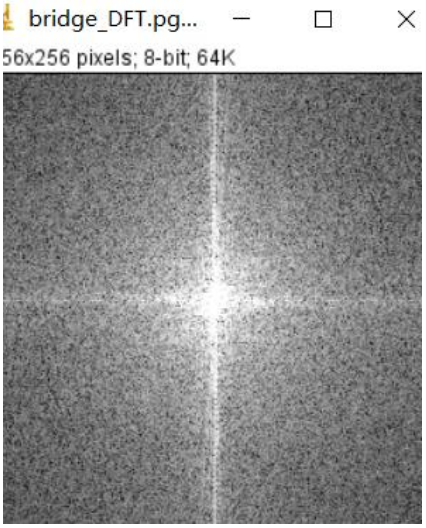
Result after DFT:



Result of processing “Bridge.pgm”:

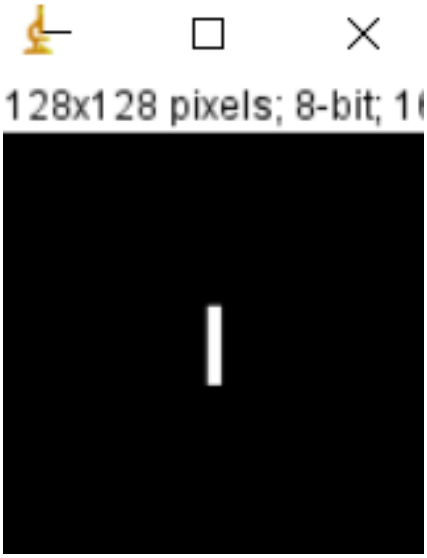
Source Image:

Result after DFT:

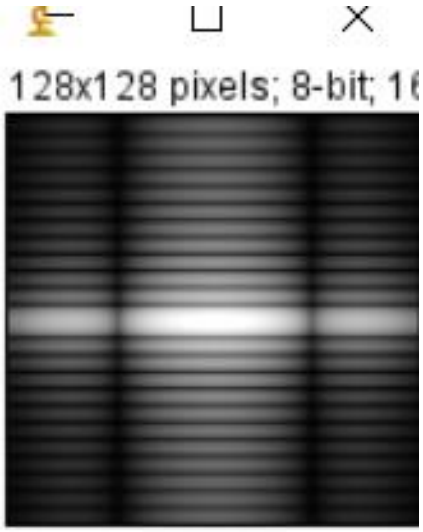


Result of processing “rectangle.pgm”:

Source Image:



Result after DFT:



Discussion:

The components of the spectrum of the DFT determine the amplitudes of the sinusoids that combine to form an image. At any given frequency in the DFT of an image, a large amplitude implies a greater prominence of a sinusoid of that frequency in the image. Conversely, a small amplitude implies that less of that sinusoid is present in the image.

Codes:

```
int size = matrixHeight * matrixWidth;
int matrix[matrixWidth][matrixHeight];
int c = matrixHeight;
memset(matrix, 0, sizeof(matrix));
for (int i = 0; i < matrixWidth; i++) {
    for (int j = 0; j < matrixHeight; j++, tempin++) {
        matrix[i][j] = *tempin * pow(-1, i + j);
    }
}

printf("Begin!...\n");

double re, im, temp;

for (int i = 0; i < matrixHeight; i++) {
    for (int j = 0; j < matrixWidth; j++, tempout++) {
        re = 0;
        im = 0;
        for (int x = 0; x < matrixHeight; x++) {
            for (int y = 0; y < matrixWidth; y++) {
                temp = (double)i * x / (double)matrixHeight + (double)j * y / (double)matrixWidth;
                re += matrix[x][y] * cos(-2 * PI * temp);
                im += matrix[x][y] * sin(-2 * PI * temp);
            }
        }
        re = c * re / size;
        im = c * im / size;
        int res = 50 * log(1 + sqrt(pow(re, 2) + pow(im, 2)));
        if (res < 0) {
            res = 0;
        } else if (res > 255) {
            res = 255;
        }
        *tempout = res;
    }
}
```

2. Phase Angle Reconstruct Image(lena):

Algorithm:

1. Use DFT transform the image
2. Calculate the phase angle

$$\phi(u,v) = \arctan\left[\frac{I(u,v)}{R(u,v)}\right]$$

3. Use IDFT and phase angle to reconstruct the image

$$f(x,y) = \frac{1}{MN}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F(u,v)e^{j2\pi(ux/M + vy/N)}$$

Results (including pictures):

Result of processing “Lena.pgm”:

Source Image:

Result after Phase Angle Reconstruct:

Discussion:

Phase Angle has advantages in determining spatial features of images. The reconstructed image using only phase Angle will contain most gray information, but we can still see from the reconstructed image that its shape features are obviously from the original image.

Codes:

```
double PhaseAngle[matrixWidth][matrixHeight];

printf("Begin!...\n");

for (int i = 0; i < matrixHeight; i++) {
    for (int j = 0; j < matrixWidth; j++) {
        double re = 0;
        double im = 0;
        for (int x = 0; x < matrixHeight; x++) {
            for (int y = 0; y < matrixWidth; y++) {
                double temp = (double)i * x / (double)matrixHeight + (double)j * y / (double)matrixWidth;
                re += matrix[x][y] * cos(-2 * PI * temp);
                im += matrix[x][y] * sin(-2 * PI * temp);
            }
        }
        PhaseAngle[i][j] = atan2(im, re);
    }
}

for (int i = 0; i < matrixHeight; i++) {
    for (int j = 0; j < matrixWidth; j++, tempout++) {
        double re = 0;
        for (int x = 0; x < matrixHeight; x++) {
            for (int y = 0; y < matrixWidth; y++) {
                double temp = (double)i * x / (double)matrixHeight + (double)j * y / (double)matrixWidth;
                re += cos(PhaseAngle[x][y] + 2 * PI * temp);
            }
        }
        int res = re;
        if (res < 0) {
            res = 0;
        } else if (res > 255) {
            res = 255;
        }
        *tempout = res;
    }
```

3. Magnitude Reconstruct Image(lena):

Algorithm:

4. Use DFT transform the image
5. Calculate the magnitude

$$|F(u,v)| = \left[R^2(u,v) + I^2(u,v)\right]^{1/2}$$

6. Use IDFT and magnitude to reconstruct the image

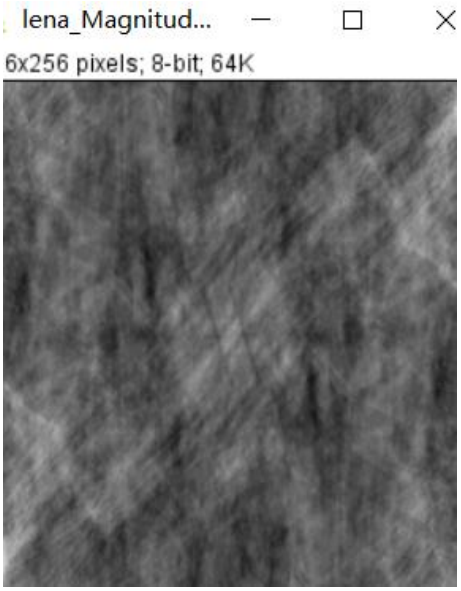
$$f(x,y) = \frac{1}{MN}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F(u,v)e^{j2\pi(ux/M + vy/N)}$$

Results (including pictures):

Result of processing “Lena.pgm”:

Source Image:

Result after Magnitude Reconstruct:



Discussion:

You just reconstruct the image using the spectrum, which means you set the exponent to one, which means you set the phase Angle to zero. Only grayscale information will be included in the result, not shape information, because the phase has been set to 0.

Codes:

```
for (int i = 0; i < matrixHeight; i++) {
    for (int j = 0; j < matrixWidth; j++) {
        re = 0;
        im = 0;
        for (int x = 0; x < matrixHeight; x++) {
            for (int y = 0; y < matrixWidth; y++) {
                temp = (double)i * x / (double)matrixHeight + (double)j * y / (double)matrixWidth;
                re += matrix[x][y] * cos(-2 * PI * temp);
                im += matrix[x][y] * sin(-2 * PI * temp);
            }
        }
        Magnitude[i][j] = sqrt(pow(re, 2) + pow(im, 2));
    }
}

for (int i = 0; i < matrixHeight; i++) {
    for (int j = 0; j < matrixWidth; j++, tempout++) {
        re = 0;
        for (int x = 0; x < matrixHeight; x++) {
            for (int y = 0; y < matrixWidth; y++) {
                temp = (double)i * x / (double)matrixHeight + (double)j * y / (double)matrixWidth;
                re += Magnitude[x][y] * cos(2 * PI * temp);
            }
        }
        int res = re / size;
        if (res < 0) {
            res = 0;
        } else if (res > 255) {
            res = 255;
        }
        *tempout = res;
    }
}
```