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. Implement 2D DFT for all provided images and analyze their transformed images in the frequency domain
2. Reconstruct lena.pgm only using the phase angle from 2D DFT
3. Reconstruct lena.pgm only using the magnitude from 2D DFT

Algorithm:

1. Discrete Fourier Transform

$$F(u,v) = \frac{1}{MN}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f(m,n)\, e^{\frac{-j2\pi um}{M}} e^{\frac{-j2\pi uv}{N}}$$

Where F the image after taking Fourier Transform, f is the original image

Input: Input: an unsigned char array that stores the source image data

Output: an unsigned char array that stores the output image data

Shift image f, $f(m,n) = (-1)^{m+n} f(m,n)$

Create two zero image t and g, where Rt is the real part of t, It is the imaginary part of t, and the same as g.

```
for (i = 0; i < image. Height; i++) {
    for (j = 0; j < image. Width; j++) {
        for (m = 0; m < image. Height; m++) {
            theta = (-2) * PI * (i * m) / image.Height;
            Rt[i][j] += (int)tempin[m][j] * cos(theta);
            It[i][j] += (int)tempin[m][ j] * sin(theta);
        }
    }
}

for (i = 0; i < image->Height; i++) {
    for (j = 0; j < image->Width; j++) {
        for (n = 0; n < image->Width; n++) {
            theta = (-2) * PI * (j * n) / image->Width;
            Rg[i] [ j] += Rt[i][n] * cos(theta) - It[i][n] * sin(theta);
            Ig[i] [ j] += Rt[i][n] * sin(theta) + It[i][n] * cos(theta);
        }
        res = sqrt(pow(Rg[i][j], 2) + pow(Ig[i][j], 2))
        tempout[i][j] = res;
    }
}
```

2. IDFT Using Phase Angle

Input: Result of DFT including two array that stores the real and imaginary part of the image

Output: an unsigned char array that stores the output image data

$$f(m,n) = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} F(u,v)\, e^{\frac{j2\pi um}{M}} e^{\frac{j2\pi uv}{N}}$$

Calculate the phase spectrum $\varphi(u,v) = \arctan\left(\frac{I(u,v)}{R(u,v)}\right)$

Shift the image $F(u,v) = (-1)^{u+v}F(u,v)$

```
for (i = 0; i < inimage.Height; i++) {
    for (j = 0; j < inimage.Width; j++) {
        Ra[i ][ j] = cos(angle[i ][ j]);
        Ia[i ] + j] = sin(angle[i ][ j]);
    }
}
for (i = 0; i < inimage.Height; i++) {
    for (j = 0; j < inimage.Width; j++) {
        for (m = 0; m < image.Height; m++) {
            theta = 2 * PI * (i * m / image->Height);
            Rt[i ][ j] += (int)Ra[m][j] * cos(theta);
            It[i ][ j] += (int)Ia[m][j] * sin(theta);
        }
    }
}

for (i = 0; i < inimage.Height; i++) {
    for (j = 0; j < inimage.Width; j++) {
        real = 0;
        imaginary = 0;
        for (n = 0; n < image->Width; n++) {
            theta = 2 * PI * (j * n) / image.Width;
            real+= Rt[i ][n] * cos(theta) - It[i ][n] * sin(theta);
            imaginary += Rt[i ][n] * sin(theta) + It[i ][n] * cos(theta);
        }
        res = imaginary;
        if (res > 255) res = 255;
        if (res < 0) res = 0;
        outimage [i ][ j] = res;
return outimage
```

3. IDFT using magnitude spectrum

   Input: Result of DFT including two array that stores the real and imaginary part of the image

   Output: an unsigned char array that stores the output image data

$$f(m,n) = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} F(u,v)\, e^{\frac{j2\pi um}{M}} e^{\frac{j2\pi uv}{N}}$$

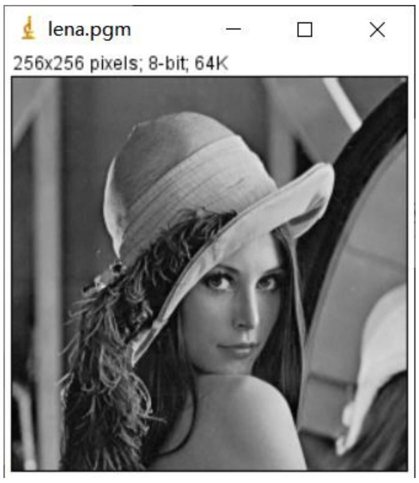
Calculate the phase spectrum $F(u,v) = (I(u,v)^2 + R(u,v)^2)^{-1}$

Shift the image $F(u,v) = (-1)^{u+v}F(u,v)$

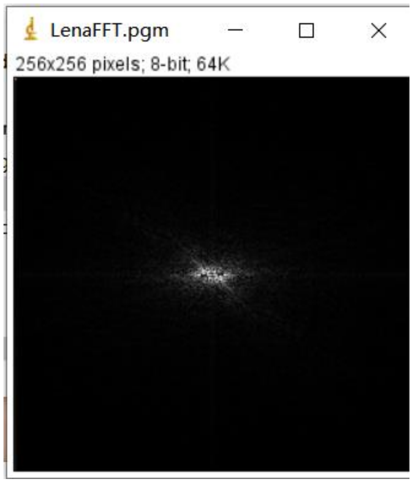
```
for (i = 0; i < inimage.Height; i++) {
    for (j = 0; j < inimage.Width; j++) {
        res=sqrt(pow(Rg[i ][ j], 2) + pow(Ig[i ][ j], 2));
        magnitude[i ][ j] = res;            }
}
for (i = 0; i < inimage.Height; i++) {
        for (j = 0; j < inimage.Width; j++) {
            for (m = 0; m < image.Height; m++) {
                theta = 2 * PI * (i * m / image->Height);
                Rt[i ][ j] += (int)Ra[m][j] * cos(theta);
                It[i ][ j] += (int)Ia[m][j] * sin(theta);
            }
        }
    }

    for (i = 0; i < inimage.Height; i++) {
        for (j = 0; j < inimage.Width; j++) {
            real = 0;
            imaginary = 0;
            for (n = 0; n < image->Width; n++) {
                theta = 2 * PI * (j * n) / image.Width;
                real+= Rt[i ][n] * cos(theta) - It[i ][n] * sin(theta);
                imaginary += Rt[i ][n] * sin(theta) + It[i ][n] * cos(theta);
            }
            res = imaginary;
            if (res > 255) res = 255;
            if (res < 0) res = 0;
            outimage [i ][ j] = res;
return outimage
```

Results (compare the results with the original image):

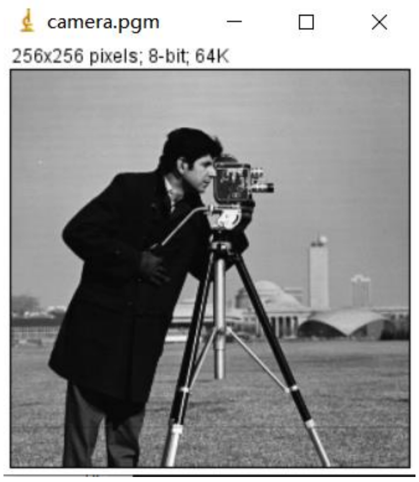
1. Fourier Transform

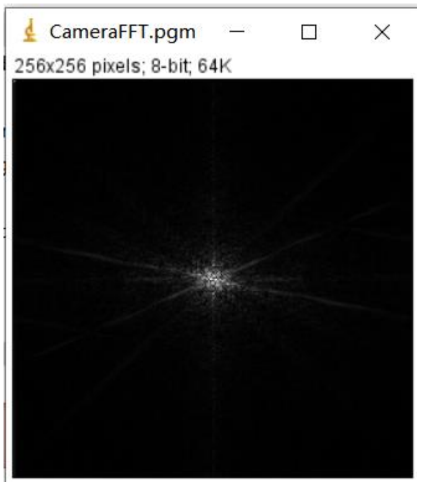


Source



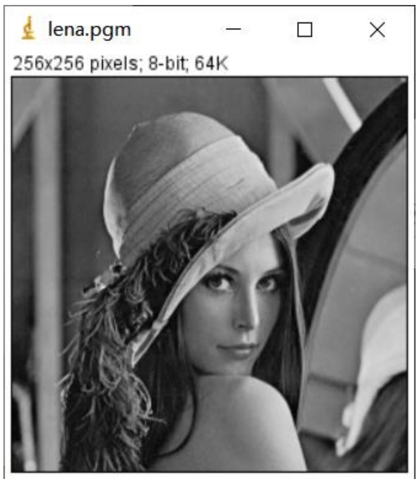
After

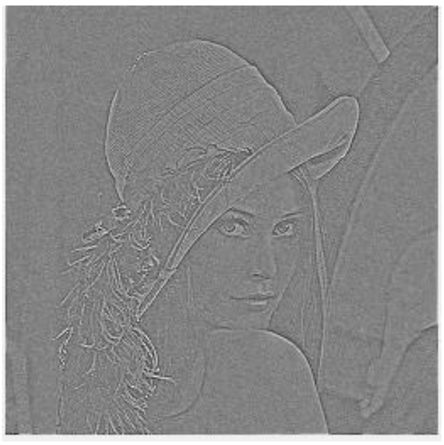




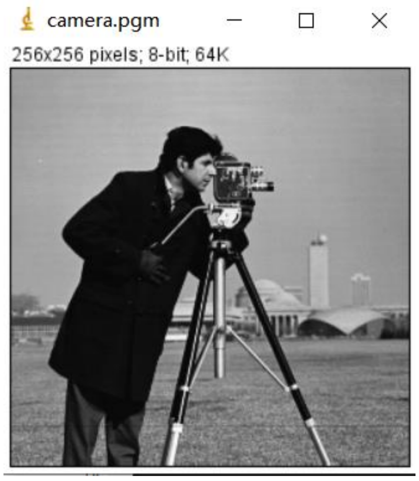
Source After

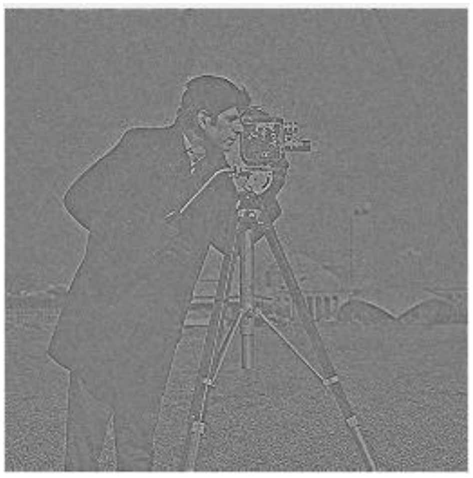
2. IDFT Using Phase spectrum





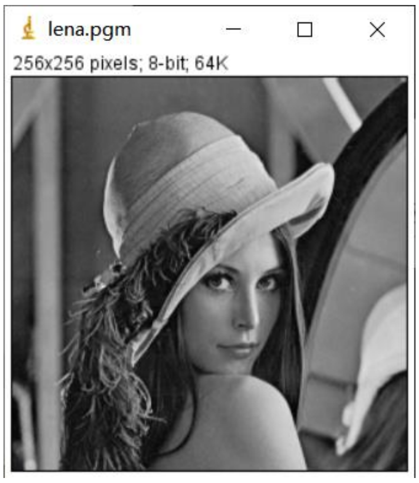
Source After



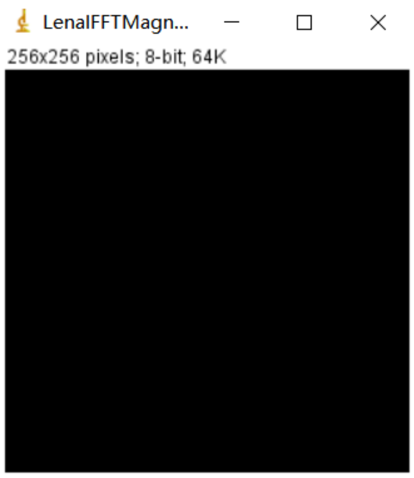


Source After

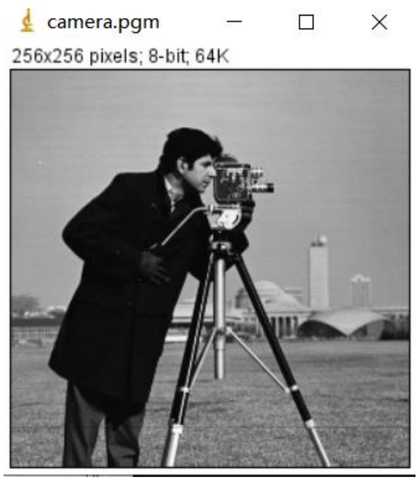
3. IDFT Using Magnitude spectrum



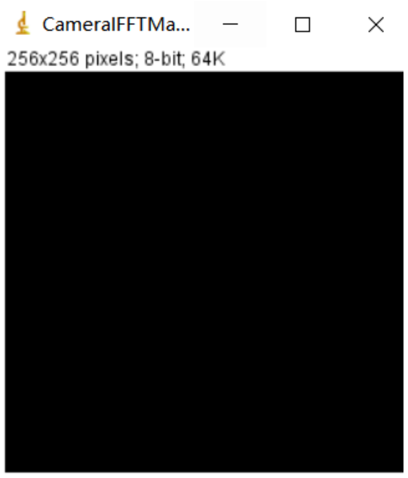
Source



After



Source



After

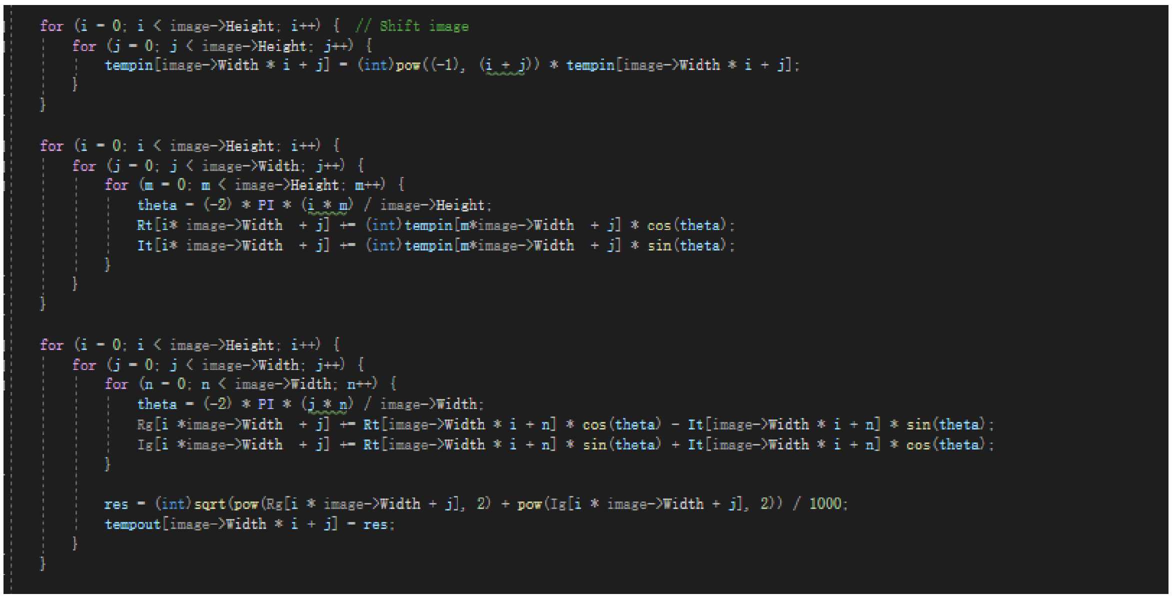
Discussion:

1. After shifting the image, the frequency spectrum can move to the central of the image
2. The original image cannot be reconstructed from the amplitude spectrum, which also verifies that the amplitude spectrum only contains the gray information of the image, and the phase spectrum plays a decisive role in the image content.

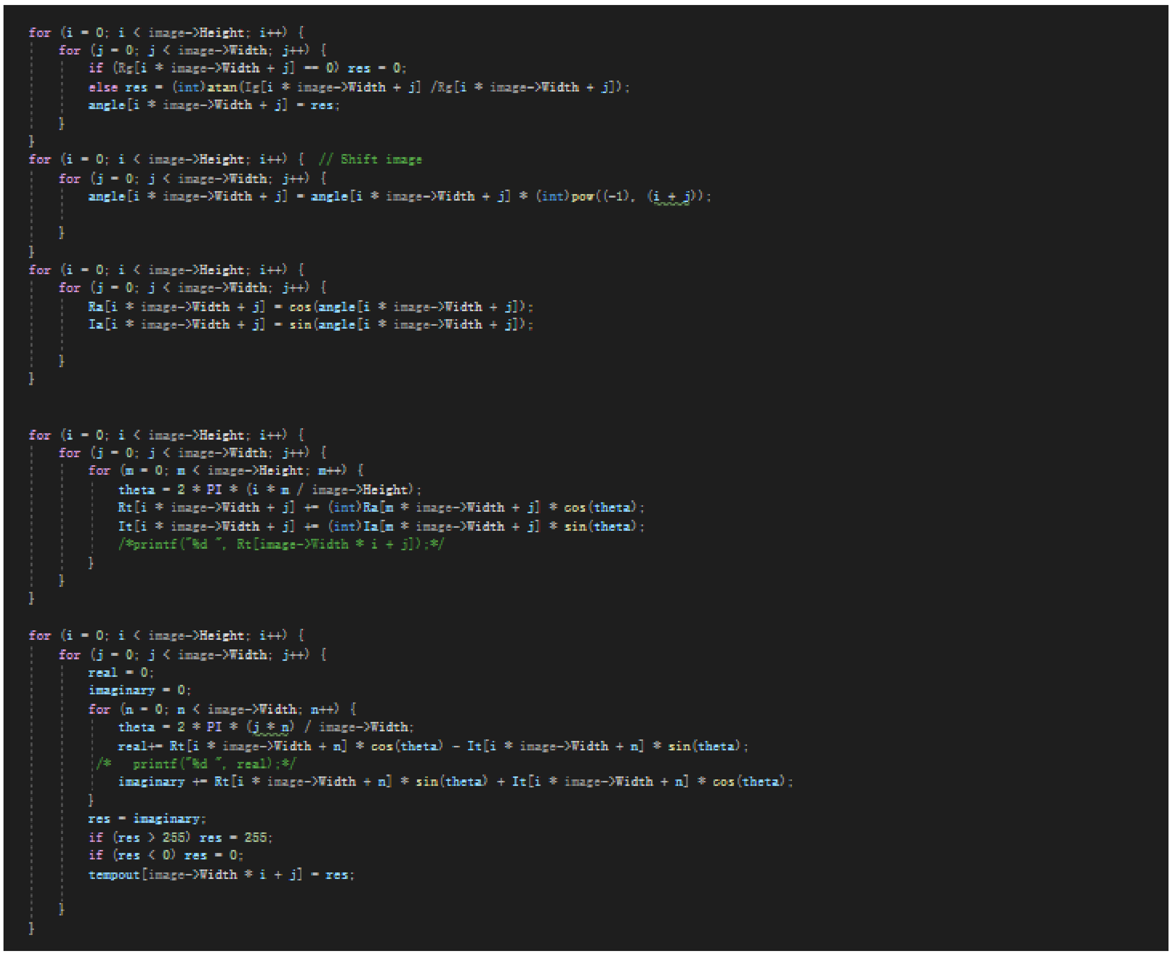
3. The phase spectrum is used to reconstruct the contour information of the image, so it is easy to verify that the phase spectrum of the image stores the position information of the image. However, due to the lack of amplitude spectrum, the reconstructed image lacks the variation in pixel value.

Codes:

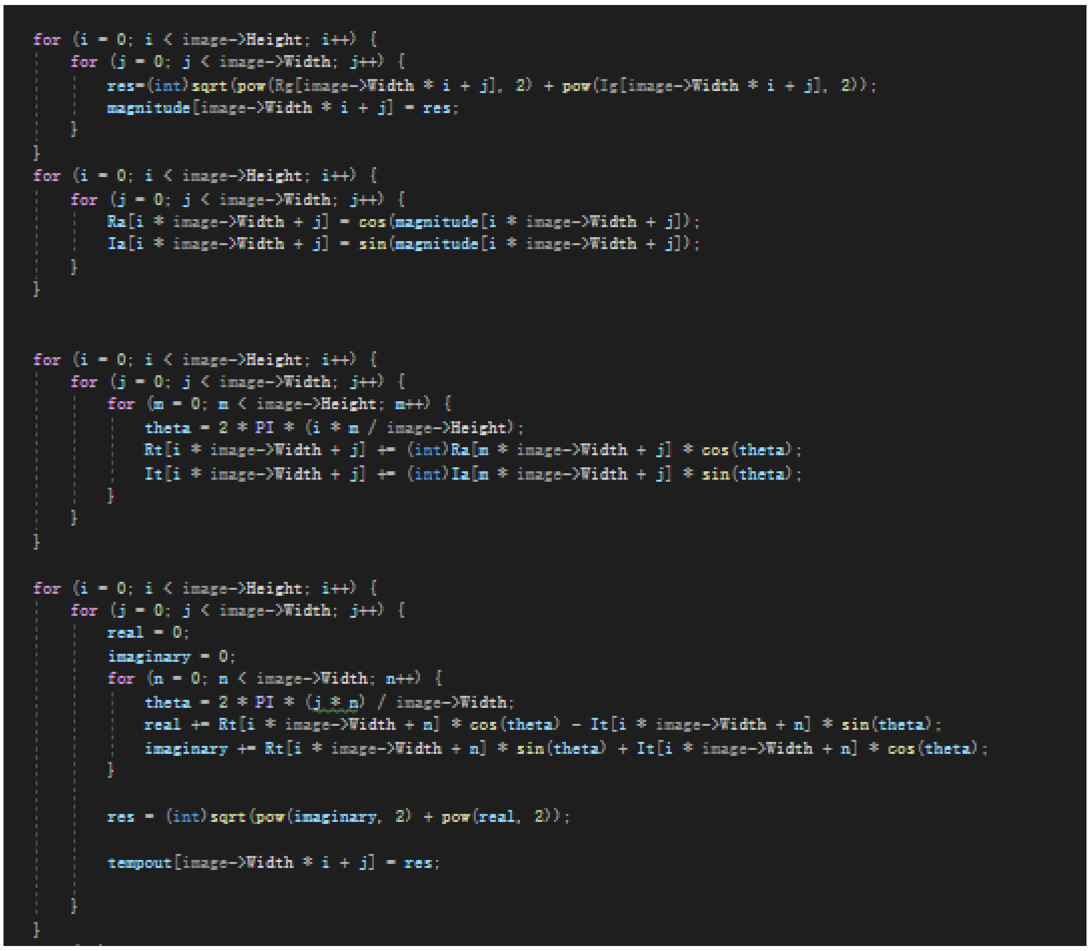
1. Fourier Transform


```
for (i = 0; i < image->Height; i++) {  // Shift image
    for (j = 0; j < image->Height; j++) {
        tempin[image->Width * i + j] = (int)pow((-1), (i + j)) * tempin[image->Width * i + j];
    }
}

for (i = 0; i < image->Height; i++) {
    for (j = 0; j < image->Width; j++) {
        for (m = 0; m < image->Height; m++) {
            theta = (-2) * PI * (i * m) / image->Height;
            Rt[i* image->Width  + j] += (int)tempin[m*image->Width  + j] * cos(theta);
            It[i* image->Width  + j] += (int)tempin[m*image->Width  + j] * sin(theta);
        }
    }
}

for (i = 0; i < image->Height; i++) {
    for (j = 0; j < image->Width; j++) {
        for (n = 0; n < image->Width; n++) {
            theta = (-2) * PI * (j * n) / image->Width;
            Rg[i *image->Width  + j] += Rt[image->Width * i + n] * cos(theta) - It[image->Width * i + n] * sin(theta);
            Ig[i *image->Width  + j] += Rt[image->Width * i + n] * sin(theta) + It[image->Width * i + n] * cos(theta);
        }

        res = (int)sqrt(pow(Rg[i * image->Width + j], 2) + pow(Ig[i * image->Width + j], 2)) / 1000;
        tempout[image->Width * i + j] = res;
    }
}
```

2. IDFT Using Phase spectrum


```
for (i = 0; i < image->Height; i++) {
    for (j = 0; j < image->Width; j++) {
        if (Rg[i * image->Width + j] == 0) res = 0;
        else res = (int)atan(Ig[i * image->Width + j] /Rg[i * image->Width + j]);
        angle[i * image->Width + j] = res;
    }
}
for (i = 0; i < image->Height; i++) {  // Shift image
    for (j = 0; j < image->Width; j++) {
        angle[i * image->Width + j] = angle[i * image->Width + j] * (int)pow((-1), (i + j));

    }
}
for (i = 0; i < image->Height; i++) {
    for (j = 0; j < image->Width; j++) {
        Ra[i * image->Width + j] = cos(angle[i * image->Width + j]);
        Ia[i * image->Width + j] = sin(angle[i * image->Width + j]);

    }
}


for (i = 0; i < image->Height; i++) {
    for (j = 0; j < image->Width; j++) {
        for (m = 0; m < image->Height; m++) {
            theta = 2 * PI * (i * m / image->Height);
            Rt[i * image->Width + j] += (int)Ra[m * image->Width + j] * cos(theta);
            It[i * image->Width + j] += (int)Ia[m * image->Width + j] * sin(theta);
            /*printf("%d ", Rt[image->Width * i + j]);*/
        }
    }
}

for (i = 0; i < image->Height; i++) {
    for (j = 0; j < image->Width; j++) {
        real = 0;
        imaginary = 0;
        for (n = 0; n < image->Width; n++) {
            theta = 2 * PI * (j * n) / image->Width;
            real+= Rt[i * image->Width + n] * cos(theta) - It[i * image->Width + n] * sin(theta);
        /*   printf("%d ", real);*/
            imaginary += Rt[i * image->Width + n] * sin(theta) + It[i * image->Width + n] * cos(theta);
        }
        res = imaginary;
        if (res > 255) res = 255;
        if (res < 0) res = 0;
        tempout[image->Width * i + j] = res;

    }
}
```

3. IDFT Using Magnitude spectrum



```
for (i = 0; i < image->Height; i++) {
    for (j = 0; j < image->Width; j++) {
        res=(int)sqrt(pow(Rg[image->Width * i + j], 2) + pow(Ig[image->Width * i + j], 2));
        magnitude[image->Width * i + j] = res;
    }
}
for (i = 0; i < image->Height; i++) {
    for (j = 0; j < image->Width; j++) {
        Ra[i * image->Width + j] = cos(magnitude[i * image->Width + j]);
        Ia[i * image->Width + j] = sin(magnitude[i * image->Width + j]);
    }
}


for (i = 0; i < image->Height; i++) {
    for (j = 0; j < image->Width; j++) {
        for (m = 0; m < image->Height; m++) {
            theta = 2 * PI * (i * m / image->Height);
            Rt[i * image->Width + j] += (int)Ra[m * image->Width + j] * cos(theta);
            It[i * image->Width + j] += (int)Ia[m * image->Width + j] * sin(theta);
        }
    }
}

for (i = 0; i < image->Height; i++) {
    for (j = 0; j < image->Width; j++) {
        real = 0;
        imaginary = 0;
        for (n = 0; n < image->Width; n++) {
            theta = 2 * PI * (j * n) / image->Width;
            real += Rt[i * image->Width + n] * cos(theta) - It[i * image->Width + n] * sin(theta);
            imaginary += Rt[i * image->Width + n] * sin(theta) + It[i * image->Width + n] * cos(theta);
        }

        res = (int)sqrt(pow(imaginary, 2) + pow(real, 2));

        tempout[image->Width * i + j] = res;

    }
}
```