

Lab 1: Filtering operations

Python packages

NumPy: Used functions related to linear algebra and matrix representation.

SciPy: Used for convolution

Matplotlib: Used for displaying images and graphs

Functions: Used to access files related to the course

```
In [ ]: import numpy as np
import matplotlib.pyplot as plt
from numpy.fft import fft2, ifft2, fftshift
from scipy.signal import convolve2d
from Functions import *
from fftwave import fftwave
from gaussfft import gaussfft
```

Section 1.3

This block contains some of the pre-work files used to understand the functions.

```
In [ ]: # this is used to set the values for p and q
p = 5
q = 9

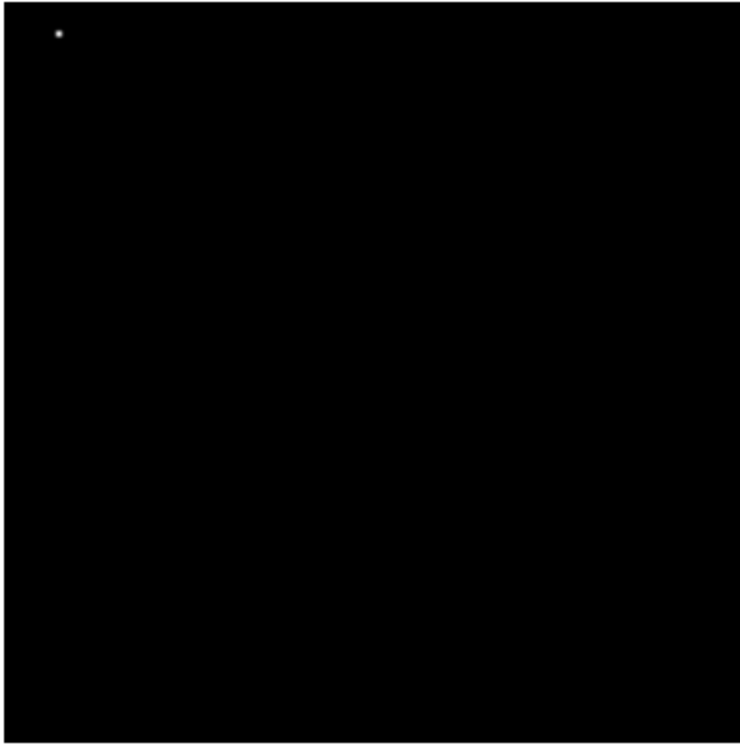
# function is zero everywhere except point (p,q).
Fhat = np.zeros((128,128))
Fhat[p, q] = 1

# this is used to show the image
print("Fhat")
showgrey(Fhat, True)

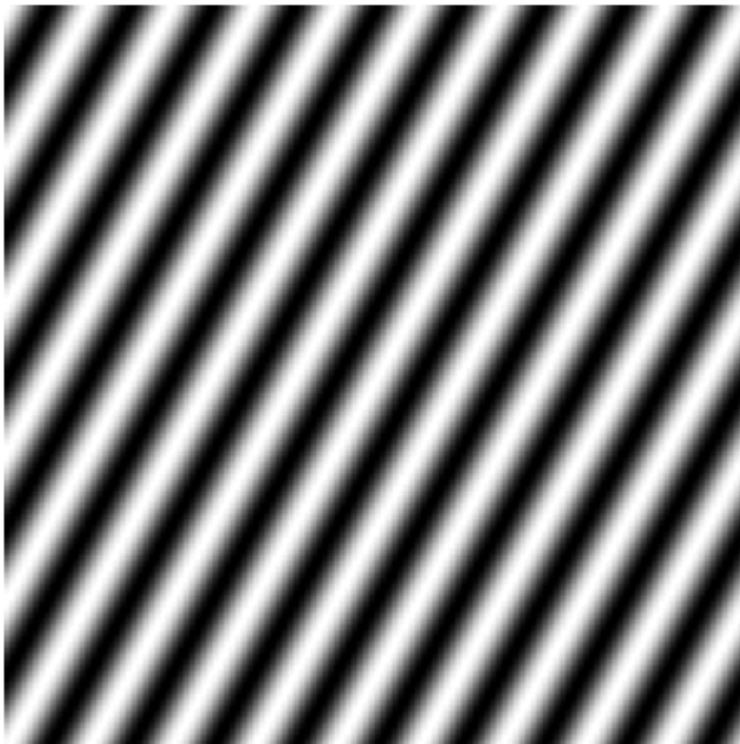
# compute inverse discrete fourier transform
F = ifft2(Fhat)

# to look at the real and imaginary parts as well as magnitude
# and phase
Fabsmax = np.max(np.abs(F))
print("real part")
showgrey(np.real(F), True, 64, -Fabsmax, Fabsmax)
print("imaginary part")
showgrey(np.imag(F), True, 64, -Fabsmax, Fabsmax)
print("magnitude")
showgrey(np.abs(F), True, 64, -Fabsmax, Fabsmax)
print("phase")
showgrey(np.angle(F), True, 64, -np.pi, np.pi)
```

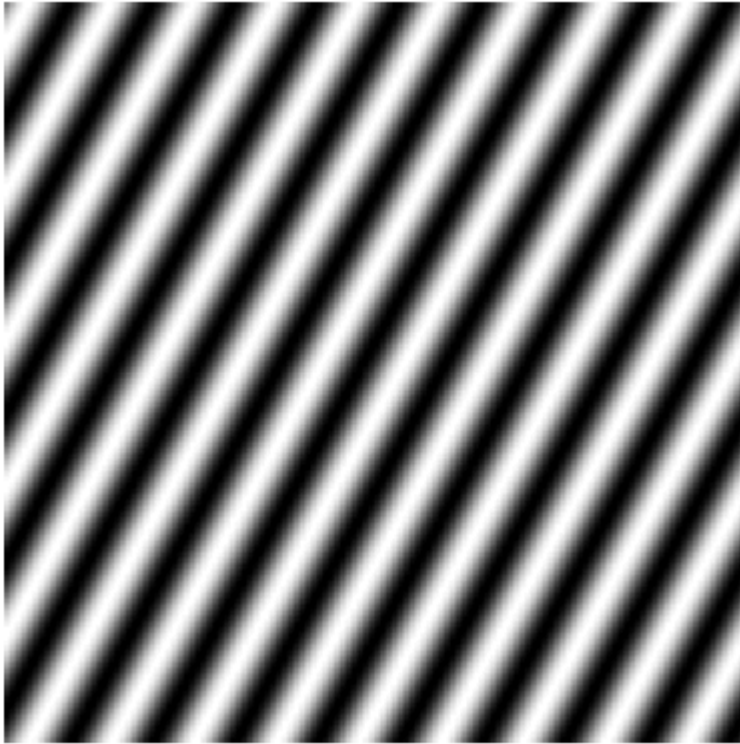
Fhat



real part



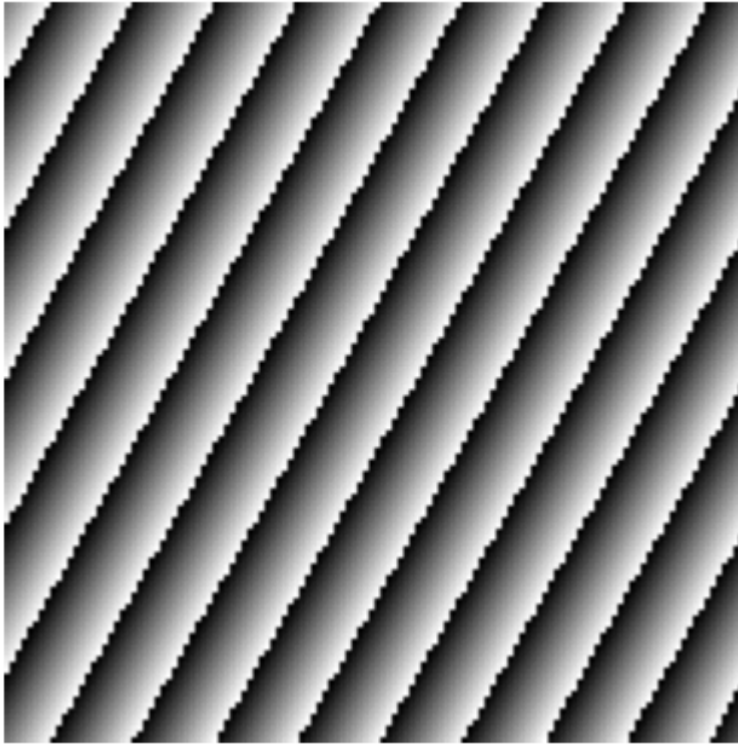
imaginary part



magnitude



phase



Question 1 Repeat this exercise with the coordinates p and q set to (5, 9), (9, 5), (17, 9), (17, 121), (5, 1) and (125, 1) respectively. What do you observe?

Answer

As per my understanding, the points all correspond to the frequency component at a point in the image. The p value stands for the x-component of the frequency while the q value stands for the y-component of the frequency.

Fhat: shows where the frequency component lies on the frequency plane given that the top left corner denotes zero frequency

Centered Fhat: shows where the frequency component lies on the frequency plane given that the centre denotes zero frequency

Real: this corresponds to the cosine component of the fourier transform and denotes the magnitude of the signal at that point

Imaginary: this corresponds to the sine component of the fourier transform and denotes the phase shift of the signal

Absolute: this is overall amplitude of the fourier transform

Angle: this is the overall wavelength of the fourier transform

As the points move further away from the zero frequency component, it can be seen that the wavelength decreases, hence showing that the frequency increases as the point moves further away from the origin.

```
In [ ]: def question_1():
    p = [5,9,17,17,5,1]
    q = [9,5,9,121,1,1]

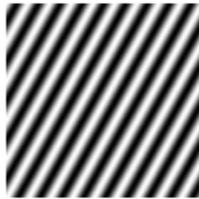
    for counter in range(6):
        fftwave(p[counter], q[counter])
```

```
question_1()
```

Fhat: $(u, v) = (5, 9)$



real(F)



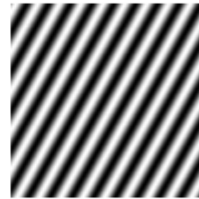
abs(F) (amplitude 0.007812)



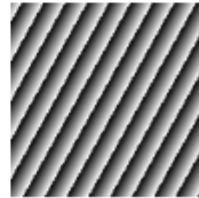
centered Fhat: $(uc, vc) = (5, 9)$



imag(F)



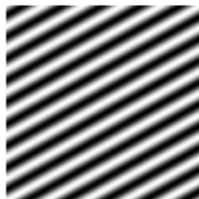
angle(F) (wavelength 0.097129)



Fhat: $(u, v) = (9, 5)$



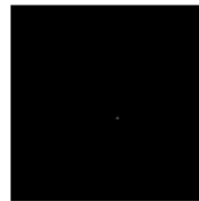
real(F)



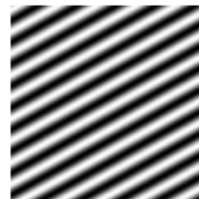
abs(F) (amplitude 0.007812)



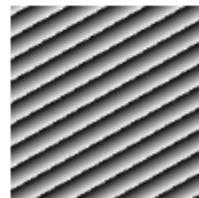
centered Fhat: $(uc, vc) = (9, 5)$



imag(F)



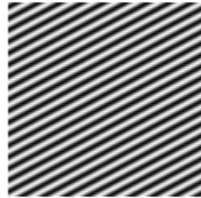
angle(F) (wavelength 0.097129)



Fhat: $(u, v) = (17, 9)$



real(F)



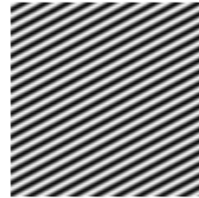
abs(F) (amplitude 0.007812)



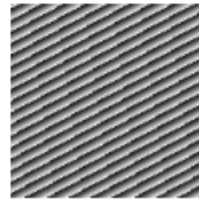
centered Fhat: $(uc, vc) = (17, 9)$



imag(F)



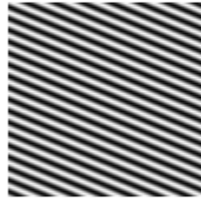
angle(F) (wavelength 0.051988)



Fhat: $(u, v) = (17, 121)$



real(F)



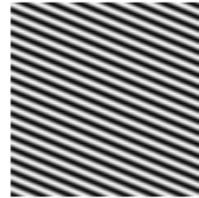
abs(F) (amplitude 0.007812)



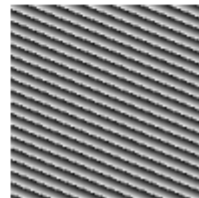
centered Fhat: $(uc, vc) = (17, -7)$

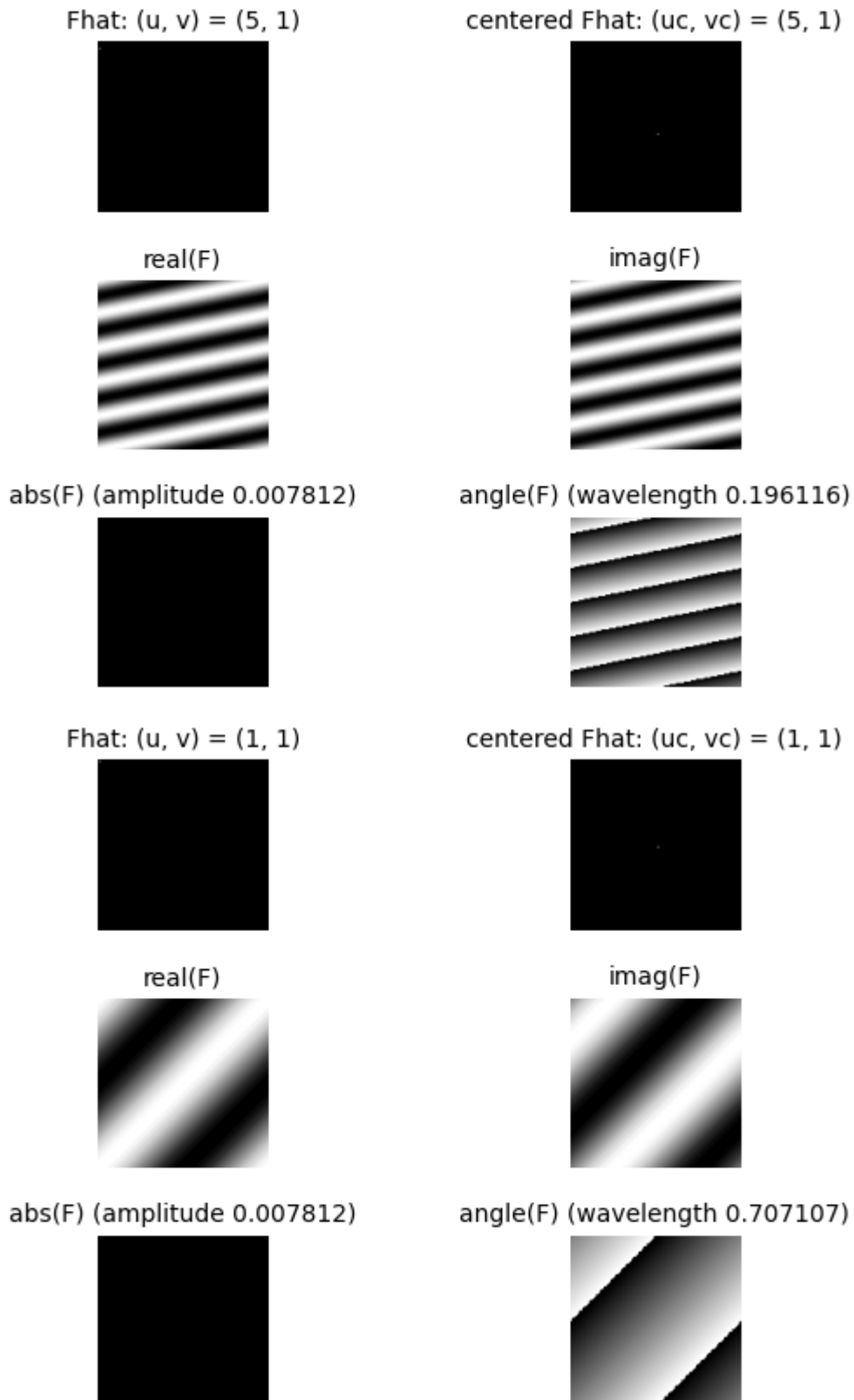


imag(F)



angle(F) (wavelength 0.054393)





Question 2 Explain how a position (p, q) in the Fourier domain will be projected as a sine wave in the spatial domain. Illustrate with a figure.

Answer

The position represents the frequency and orientation of the sinusoidal waves. The below equation represents the inverse Fourier transform in the discrete domain.

$$F(x) = \frac{1}{N} \sum_{u \in [0:N-1]^2} \hat{F}(u) e^{\left(\frac{2(\pi)iu^T x}{N}\right)}$$

Applying the inverse Fourier transform at the point (p,q), we get,

$$F(x) = \frac{1}{N} e^{\left(\frac{2(\pi)i(p,q)^T x}{N}\right)}$$

The below equation shows the Euler identity formula.

$$\exp(i\omega^T x) = \cos(\omega^T x) + i\sin(\omega^T x)$$

By applying the Euler Identity to the inverse Fourier transform, we get,

$$F(x) = \frac{1}{N} \cos\left(\frac{2\pi(p,q)^T x}{N}\right) + i \frac{1}{N} \sin\left(\frac{2\pi(p,q)^T x}{N}\right)$$

```
In [ ]: def question_2(p, q, size):
    x = np.arange(size)
    y = np.arange(size)
    X, Y = np.meshgrid(x, y)

    # Create a 2D Sine wave
    sine_wave = 1/size * np.sin(2 * np.pi * (p * X + q * Y) / size)

    plt.imshow(sine_wave, cmap="gray")
    plt.title(f"2D Sine Wave from Fourier Domain Point ({p}, {q})")
    plt.axis("off")
    plt.show()

question_2(19,3,128)
```



Question 3 How large is the amplitude? Write down the expression derived from Equation (4) in these notes. Complement the code (variable amplitude) accordingly.

Answer

The below equation represents the inverse Fourier transform in the discrete domain.

$$F(x) = \frac{1}{N} \sum_{u \in [0:N-1]^2} \hat{F}(u) e^{\left(\frac{2\pi i u^T x}{N}\right)}$$

Applying the inverse Fourier transform at the point (p,q), we get,

$$F(x) = \frac{1}{N} e^{\left(\frac{2\pi i(p,q)^T x}{N}\right)}$$

By applying the Euler Identity to the inverse Fourier transform, we get,

$$F(x) = \frac{1}{N} \cos\left(\frac{2\pi(p,q)^T x}{N}\right) + i \frac{1}{N} \sin\left(\frac{2\pi(p,q)^T x}{N}\right)$$

The amplitude of the inverse Fourier Transform is, $|F(x)| = \frac{1}{N}$

Question 4 How does the direction and length of the sine wave depend on p and q?

Write down the explicit expression that can be found in the lecture notes. Complement the code (variable wavelength) accordingly.

Answer

To find the phase of the sine wave (direction), we use,

$$\phi = \tan^{-1}\left(\frac{\text{Im}[\text{Fhat}(u,v)]}{\text{Re}[\text{Fhat}(u,v)]}\right)$$

The wavelength of the sine wave is given by,

$$\lambda = \frac{1}{\sqrt{u^2 + v^2}}$$

Question 5 What happens when we pass the point in the center and either p or q exceeds half the image size? Explain and illustrate graphically with a figure!

Answer

When we center Fourier Transform, we can take advantage of the symmetry property of the frequency components. This means that the component can be remapped in the Fourier domain.

Question 6 What is the purpose of the instructions following the question: What is done by these instructions? in the code?

Answer

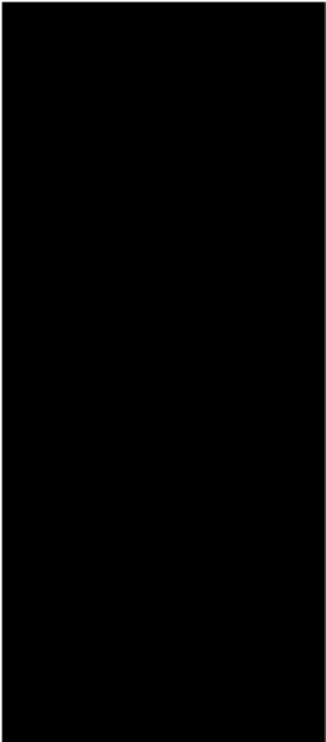
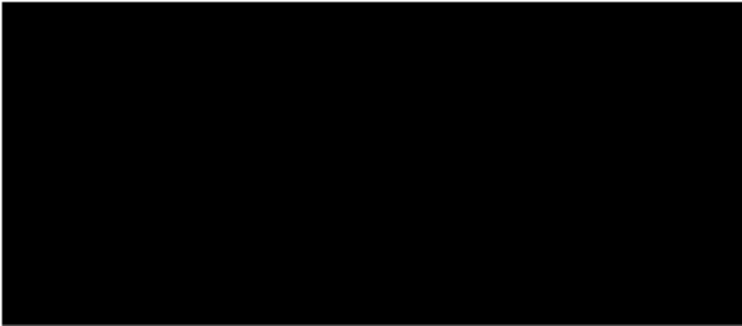
These instructions provide the coordinates for the x and y frequency components after centering the zero-frequency component on the graph.

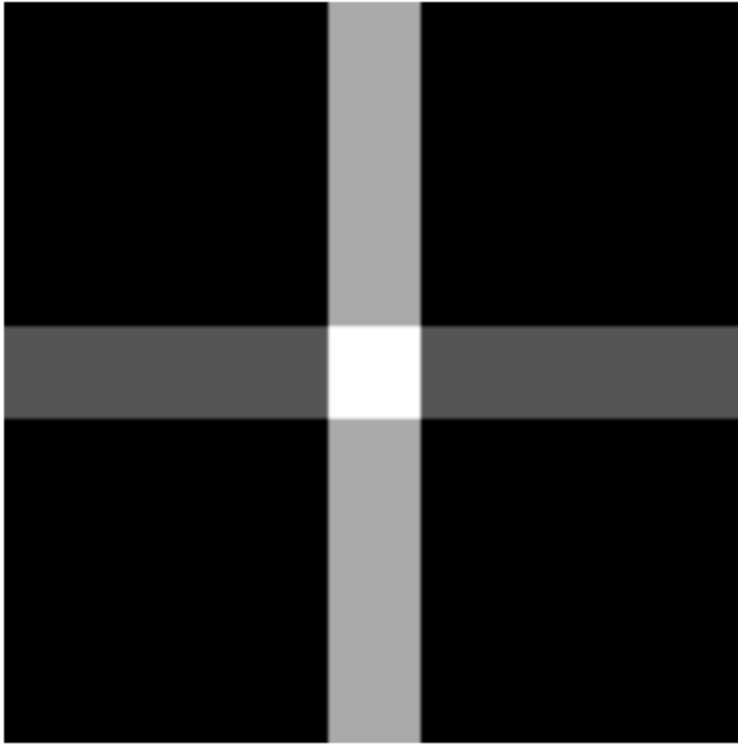
Section 1.4

This block contains some of the pre-work files used to understand the functions.

```
In [ ]: F = np.concatenate([np.zeros((56,128)), np.ones((16,128)), np.zeros((56,128))])
G = F.T
H = F + 2*G

showgrey(F, True)
showgrey(G, True)
showgrey(H, True)
```



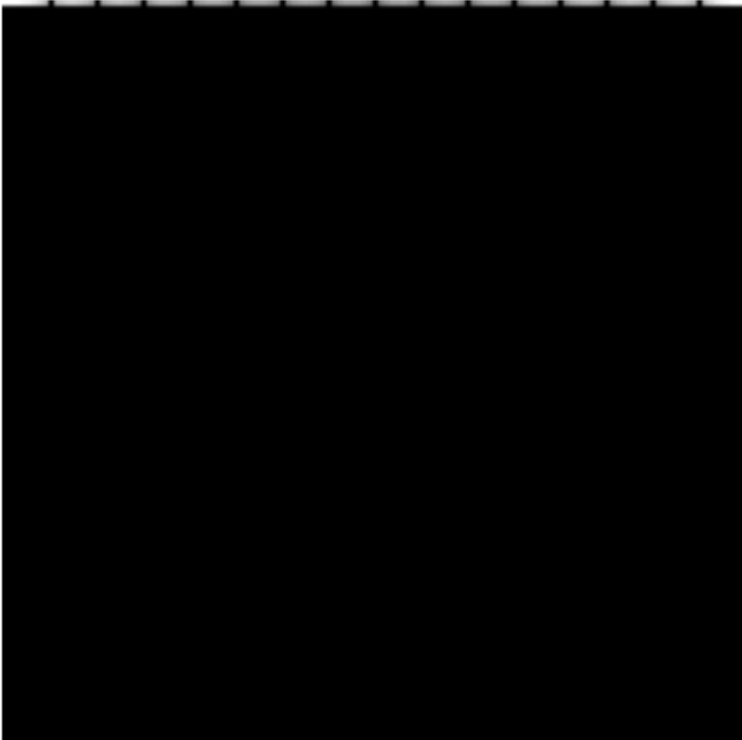


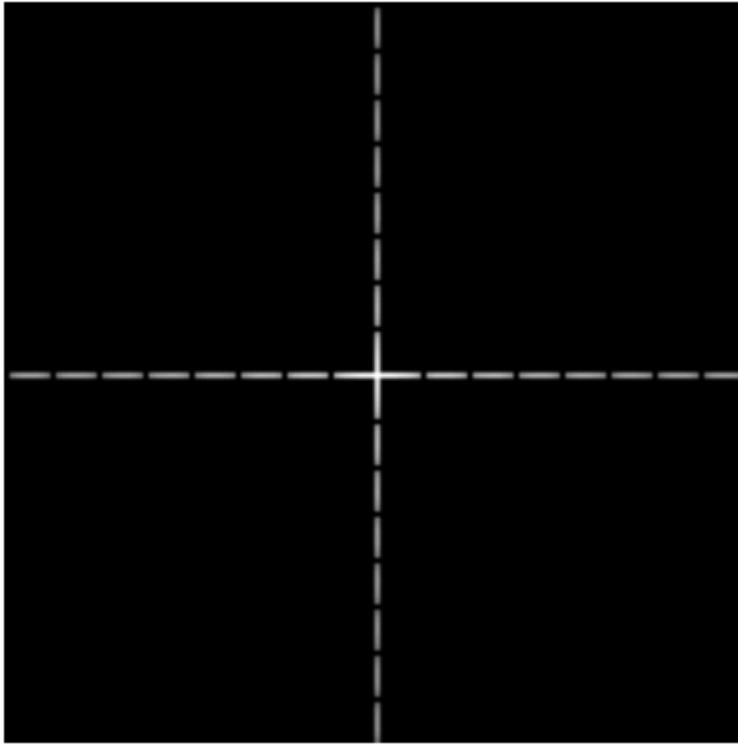
```
In [ ]: Fhat = fft2(F)
        Ghat = fft2(G)
        Hhat = fft2(H)

        showgrey(np.log(1 + np.abs(Fhat)))
        showgrey(np.log(1 + np.abs(Ghat)))
        showgrey(np.log(1 + np.abs(Hhat)))

        showgrey(np.log(1 + np.abs(fftshift(Hhat))))
```







Question 7 Why are these Fourier spectra concentrated to the borders of the images? Can you give a mathematical interpretation? Hint: think of the frequencies in the source image and consider the resulting image as a Fourier transform applied to a 2D function. It might be easier to analyze each dimension separately!

Answer

For the discrete fourier transform in two dimensions, we have,

$$\hat{F}(u, v) = \frac{1}{\sqrt{MN}} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) e^{-2\pi i (\frac{xu}{M} + \frac{yv}{N})}$$

For the above signal, by applying it to our image, we get,

$$\begin{aligned} \hat{F}(u, v) &= \frac{1}{128} \sum_{y=57}^{72} \sum_{x=0}^{127} e^{-2\pi i (\frac{xu}{128} + \frac{yv}{128})} \\ \hat{F}(u, v) &= \frac{1}{128} \sum_{y=57}^{72} \sum_{x=0}^{127} e^{-2\pi i (\frac{xu}{128})} e^{-2\pi i (\frac{yv}{128})} \\ \hat{F}(u, v) &= \frac{1}{128} \sum_{y=57}^{72} e^{-2\pi i (\frac{yv}{128})} \sum_{x=0}^{127} e^{-2\pi i (\frac{xu}{128})} \\ \hat{F}(u, v) &= \sum_{y=57}^{72} e^{-2\pi i (\frac{yv}{128})} \delta \end{aligned}$$

The Dirac function is defined such that the peak is only present at $x = 0$ hence the Fourier transform is concentrated at the border. This is the reference to the proof used to understand the presence of the Dirac function.

<https://math.stackexchange.com/questions/911959/the-result-of-exponential-sum-formula>

Question 8 Why is the logarithm function applied?

Answer

The logarithm function is used to make more details visible by accounting for a large dynamic range and allowing lower ranges to become more visible.

Question 9 What conclusions can be drawn regarding linearity? From your observations can you derive a mathematical expression in the general case?

Answer

In the Fourier domain, the linearity of the signal is preserved. The fourier transforms of F and G can be seen within the first two graphs. It can be observed that adding the Fourier transforms of the individual signals is equivalent to evaluating the Fourier transform of the combined signal. We can write the general mathematical form as,

$$\mathcal{F}(f(t) + g(t)) = F(\omega) + G(\omega)$$

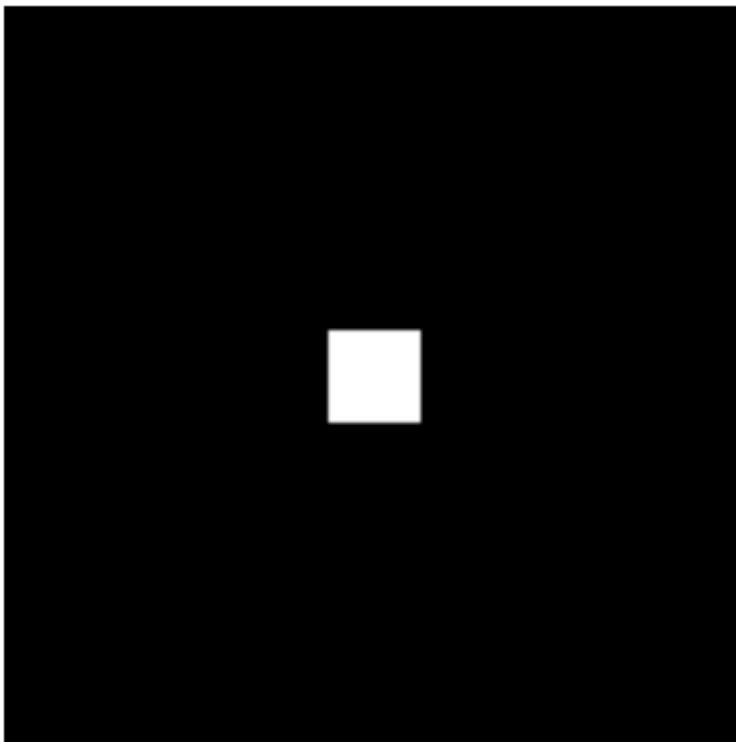
Section 1.5

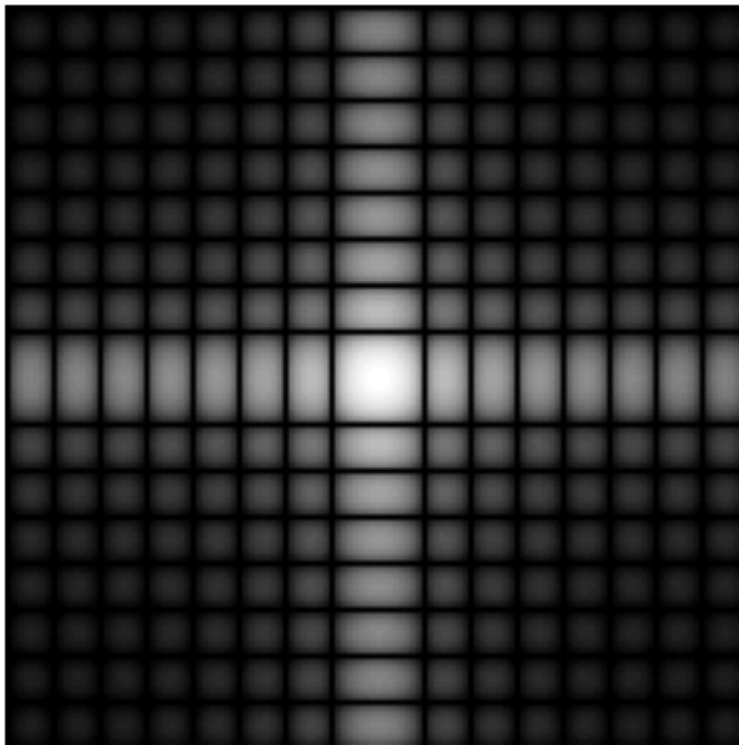
This code block contains some of the prework files.

```
In [ ]: F = np.concatenate([np.zeros((56,128)), np.ones((16,128)), np.zeros((56,128))])
        G = F.T

        showgrey(F*G)
        showfs(fft2(F*G))

        # fft(g) * fft(G)
```





Question 10 Are there any other ways to compute the last image? Remember what multiplication in Fourier domain equals to in the spatial domain! Try to perform these alternative computations in practice, if you want.

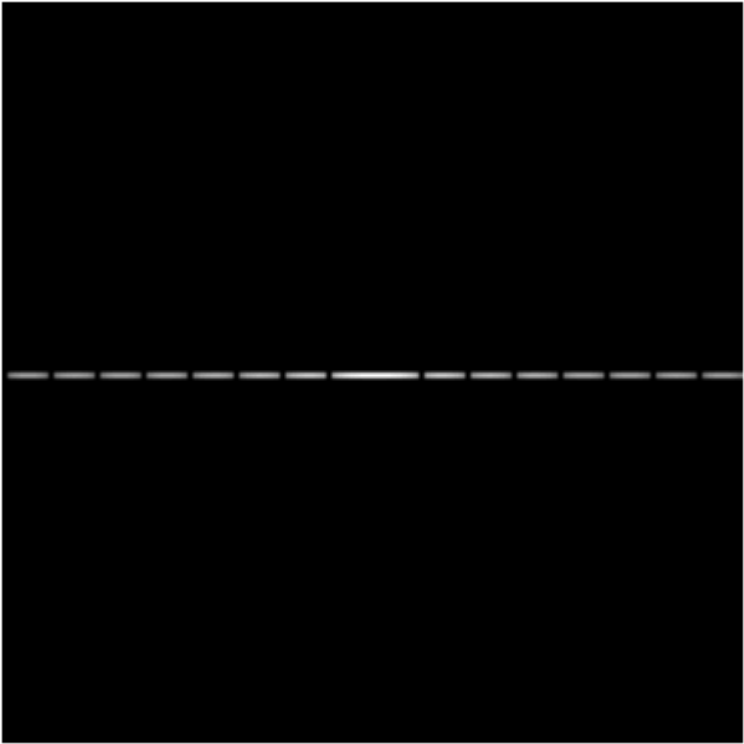
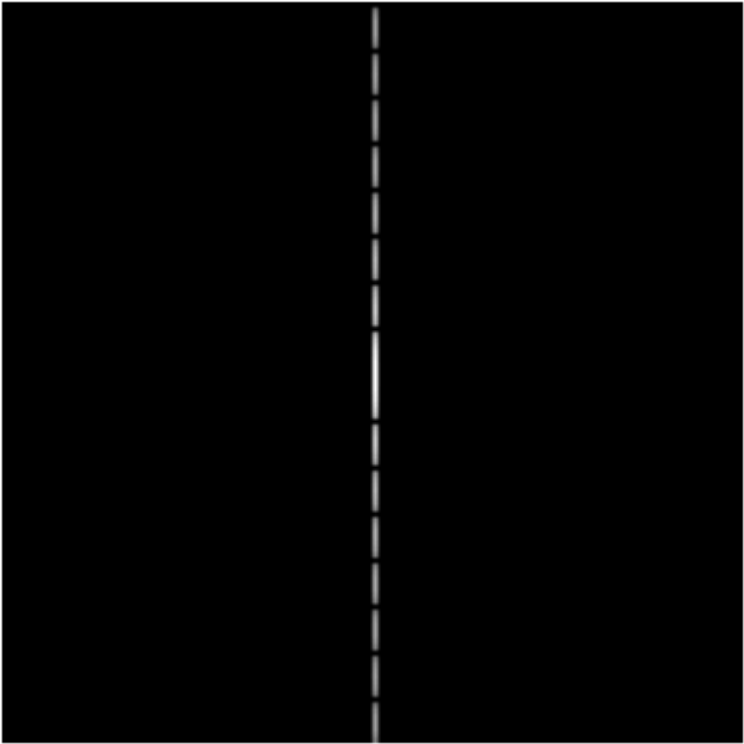
Answer

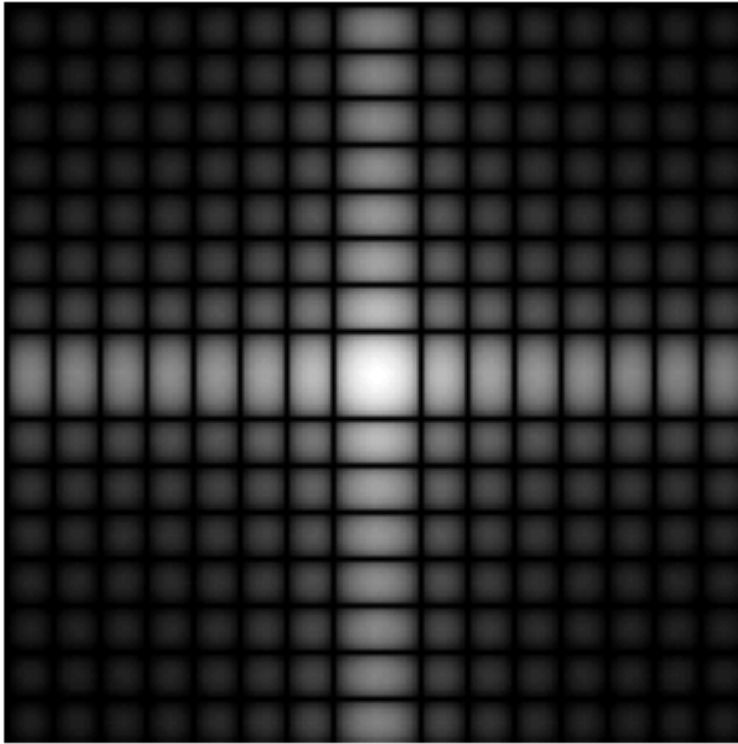
When multiplying in the Fourier domain, it is an equivalent of convolution in the Spatial domain. Another way to compute the same image is to convolve the two signals, then perform a Fourier transform on it to obtain the image above.

$$f(x) * g(x) = F(u)G(u)$$

```
In [ ]: def question_10():
    FHat = fft2(F)
    GHat = fft2(G)
    showfs(FHat)
    showfs(GHat)
    # Here we normalize the convolved Fourier Transform
    FGHat = convolve2d(FHat, GHat) / 128**2
    FGHat = FGHat[0:127, 0:127]
    showfs(FGHat)

question_10()
```

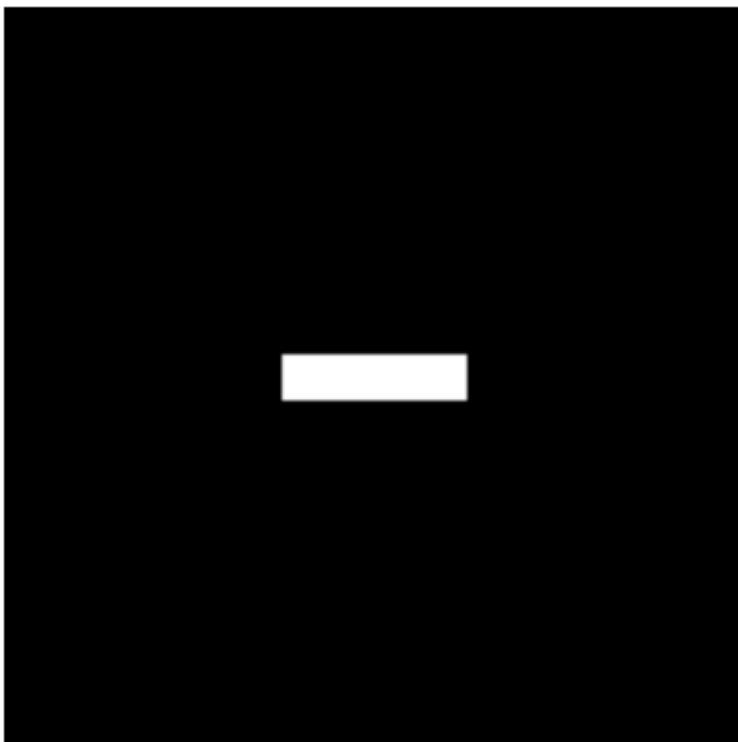


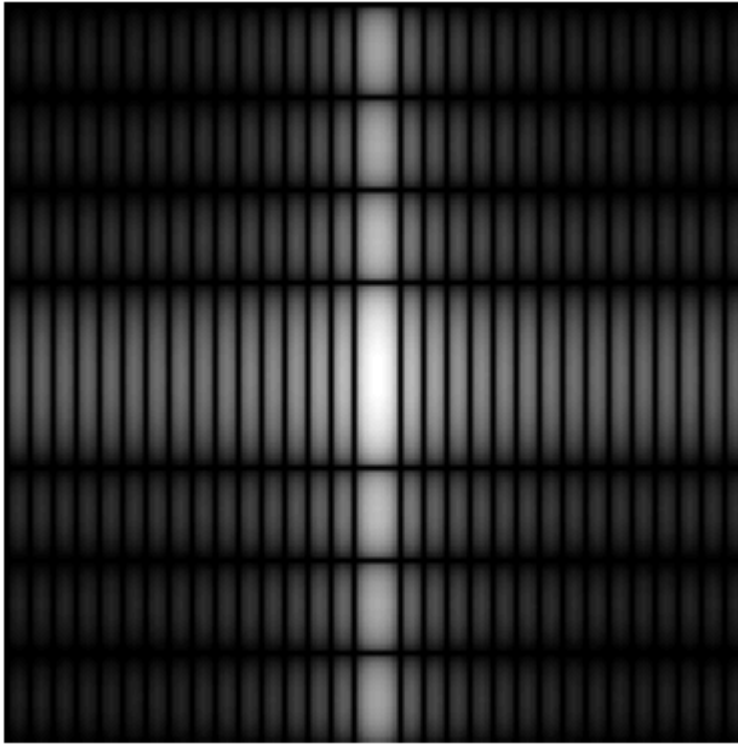


Section 1.6

This code block contains some of the prework files.

```
In [ ]: F = np.concatenate([np.zeros((60,128)), np.ones((8,128)), np.zeros((60,128))]) *  
np.concatenate([np.zeros((128,48)), np.ones((128,32)), np.zeros((128,48))], axis  
showgrey(F, True)  
showfs(fft2(F))
```





Question 11 What conclusions can be drawn from comparing the results with those in the previous exercise? See how the source images have changed and analyze the effects of scaling.

Answer

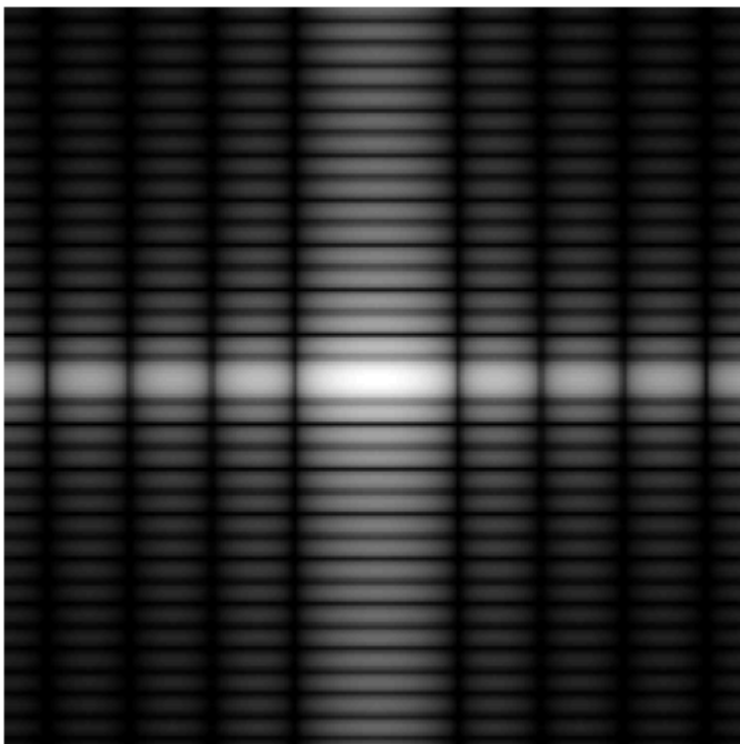
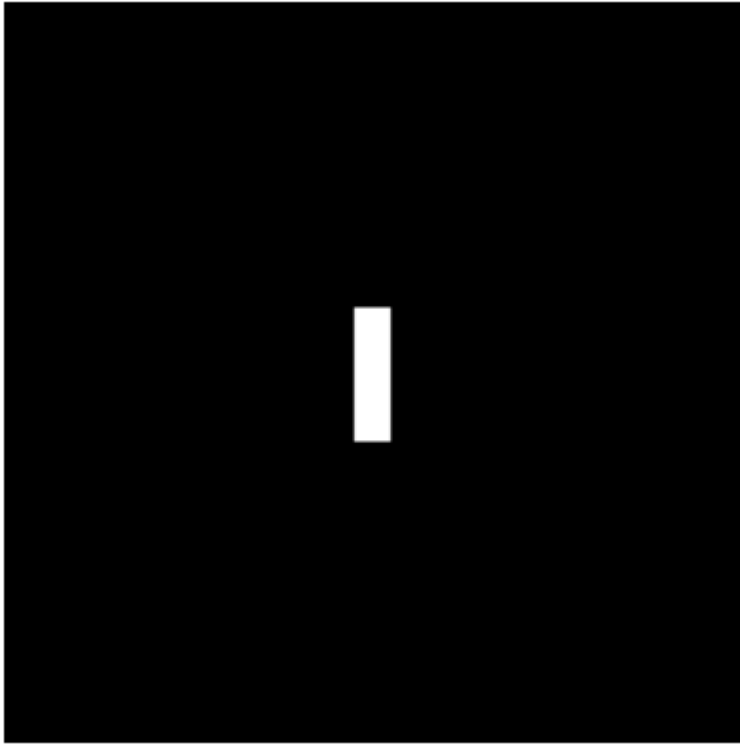
The image, in spatial domain, has been shrunk in the y-axis and expanded in the x-axis. This corresponds to expanding in the y-axis of the Fourier domain and the shrinking in the x-axis in the Fourier domain.

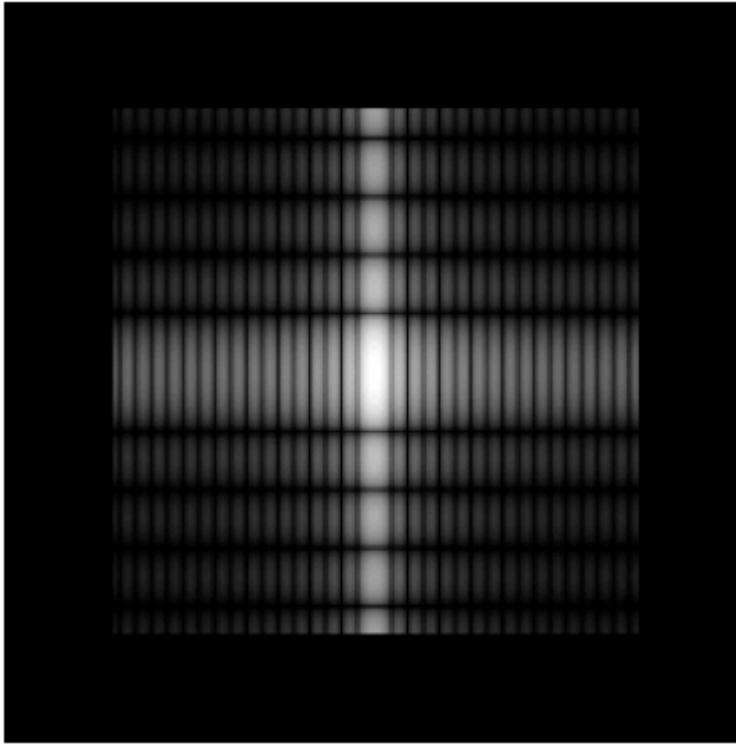
Section 1.7

This code block contains some of the prework files.

```
In [ ]: alpha = 90
G = rot(F, alpha )
showgrey(G)
Ghat = fft2(G)
showfs(Ghat)

Hhat = rot(fftshift(Ghat), -alpha )
showgrey(np.log(1 + abs(Hhat)))
```





Question 12 What can be said about possible similarities and differences? Hint: think of the frequencies and how they are affected by the rotation.

When an image is rotated in the spatial domain, its equivalent representation in the frequency domain is also rotated. The value of the frequencies at each point remain the same except that they are rotated with the spatial image. When rotating an image at non-orthogonal angles, aliasing artifacts can occur. These can be seen when alpha has the value of 30 and 60 degrees. The frequency components are not an exact rotation of the signal when alpha is 0 degrees.

Section 1.8

Question 13 What information is contained in the phase and in the magnitude of the Fourier transform?

```
In [ ]: def question_13():
    image_arr = [np.load("Images-npy/phonecalc128.npy"), np.load("Images-npy/few128.npy"), np.load("Images-npy/nalloy128.npy")]
    title_arr = ["phonecalc128", "few128", "nalloy128"]

    # Create a 3x3 plot array
    fig, axes = plt.subplots(3, 3, figsize=(10, 10))

    # Plot data on each subplot
    for i in range(3):
        axes[i, 0].imshow(image_arr[i], cmap="gray")
        axes[i, 0].set_title(title_arr[i])
        axes[i, 0].axis("off")

    for i in range(3):
        axes[i, 1].imshow(pow2image(image_arr[i]), cmap="gray")
        axes[i, 1].set_title(title_arr[i] + "- keep phase")
        axes[i, 1].axis("off")
```

```

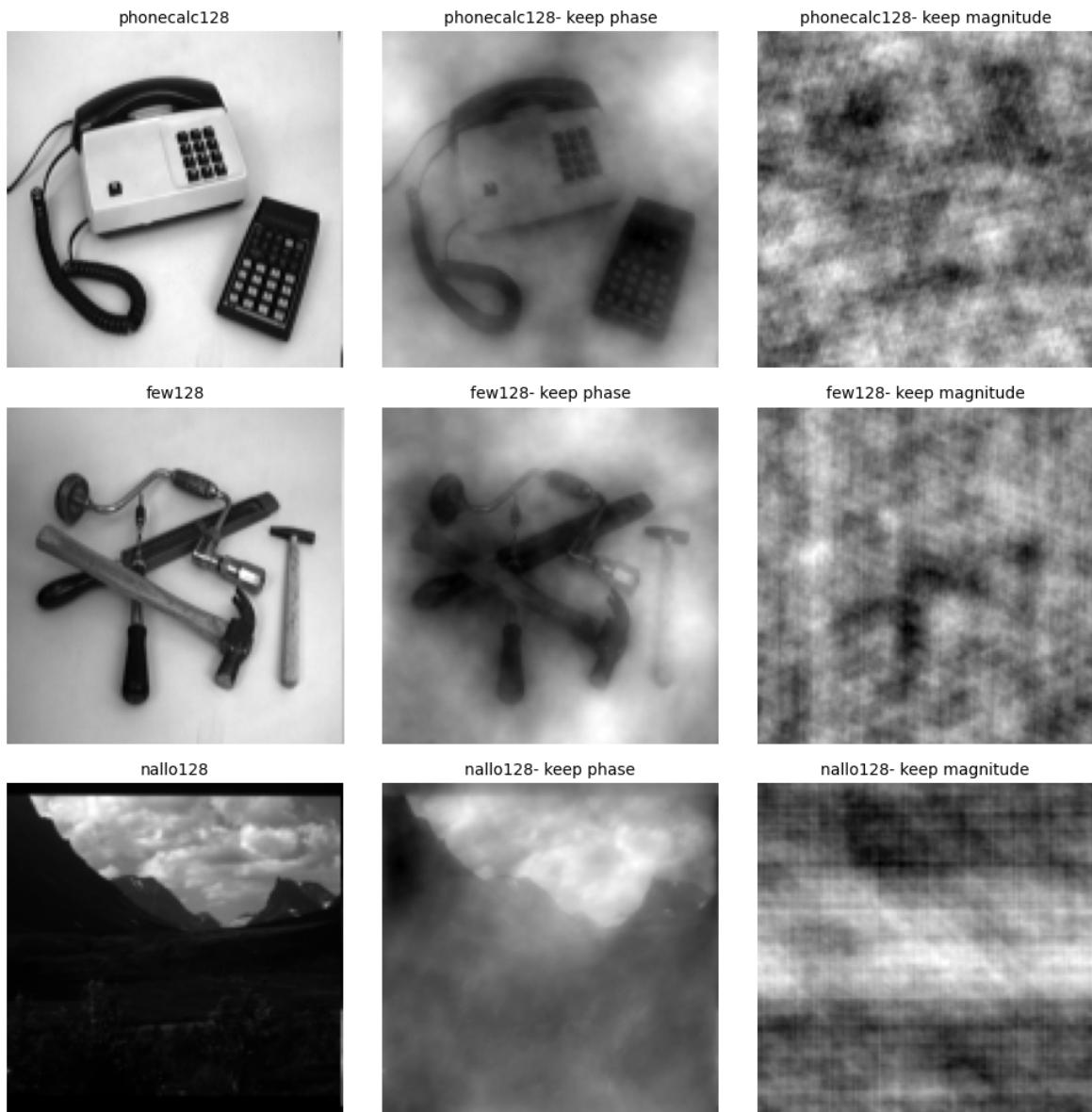
for i in range(3):
    axes[i, 2].imshow(randphaseimage(image_arr[i]), cmap="gray")
    axes[i, 2].set_title(title_arr[i] + "- keep magnitude")
    axes[i, 2].axis("off")

# Adjust spacing between subplots
plt.tight_layout()

# Show the plot
plt.show()

question_13()

```



Answer

The magnitude of the Fourier transform represents the amplitude of each frequency component in the image. The higher the magnitude component, the more dominant the frequency component within the signal. The magnitude is a real and non-negative number.

The phase represents the phase shift or time delay of each frequency component and is important in the reconstruction of the original image as it contains information about the position and alignment of the edges and contours in the spatial domain of the image.

The phase of an image is generally unique. This means that different images with the same magnitude spectrum can have significantly different phase spectra.

When using `pow2image`, we see that the position of the edges of the image are preserved but the magnitude has changed. When using `randphaseimage`, we see that the images cannot be recognized as the positions of the edges have not been preserved within the image. This means that the image can be recognized if the phase is preserved even if the magnitude is lost.

Section 2.3

Question 14 Show the impulse response and variance for the above-mentioned t-values. What are the variances of your discretized Gaussian kernel for $t = 0.1, 0.3, 1.0, 10.0$ and 100.0 ?

```
In [ ]: def question_14():
        t_arr = [0.1, 0.3, 1.0, 10.0, 100.0]

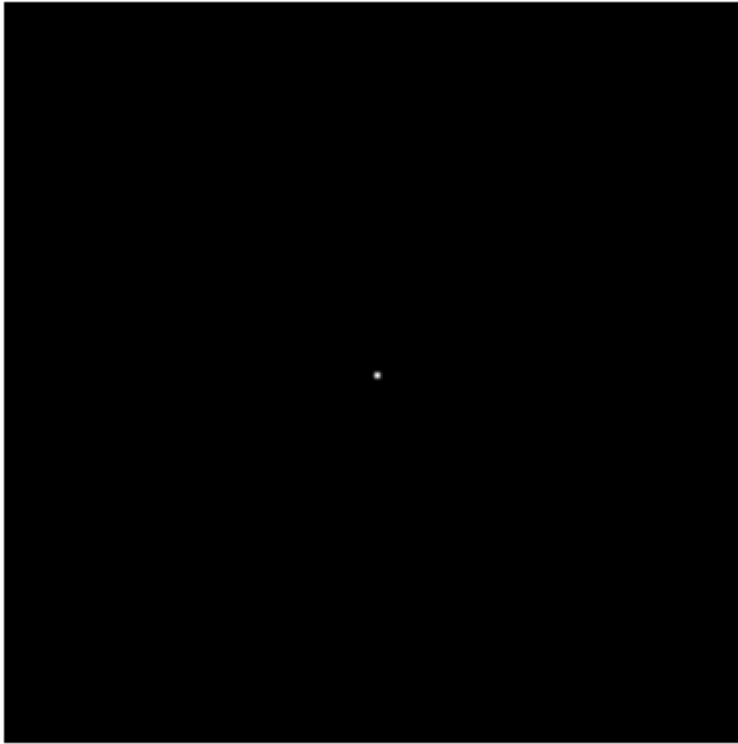
        print("Source Image")
        showgrey(deltafcn(128, 128), True)

        for t in t_arr:
            psf = gaussfft(deltafcn(128, 128), t)
            discpsf = discgaussfft(deltafcn(128,128), t)

            print("t: " + str(t))
            print("gaussfft")
            print("variance" + str(variance(psf)))
            showgrey(psf, True)
            print("discgaussfft")
            print("variance" + str(variance(discpsf)))
            showgrey(discpsf, True)

        question_14()
```

Source Image



```
(128, 128) (128, 128) (128, 128)
```

```
t: 0.1
```

```
gaussfft
```

```
variance[[1.32967252e-02 4.96824804e-15]
```

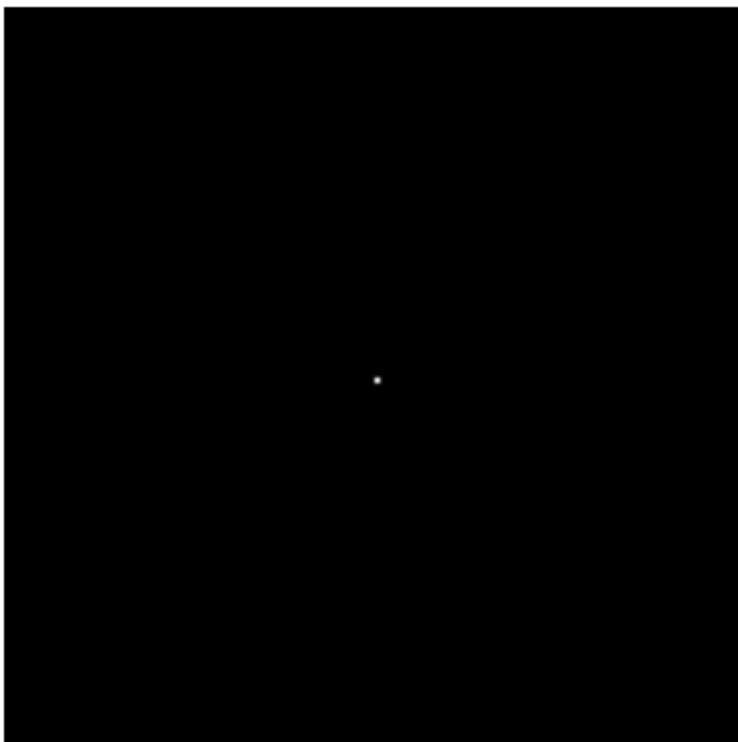
```
[4.96824804e-15 1.32967252e-02]]
```

```
c:\Users\karen\git\Computer-Vision-course\Labs\Lab 1\Karen\Functions.py:46: Compl  
exWarning: Casting complex values to real discards the imaginary part
```

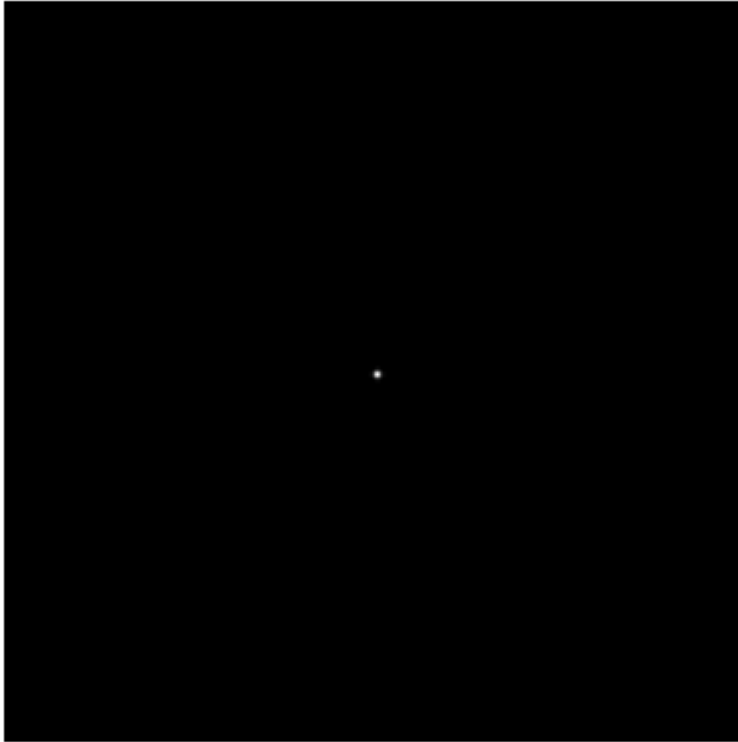
```
im = res*(Image.astype(np.float32) - zmin)/float(range)
```

```
c:\Users\karen\git\Computer-Vision-course\Labs\Lab 1\Karen\Functions.py:49: Compl  
exWarning: Casting complex values to real discards the imaginary part
```

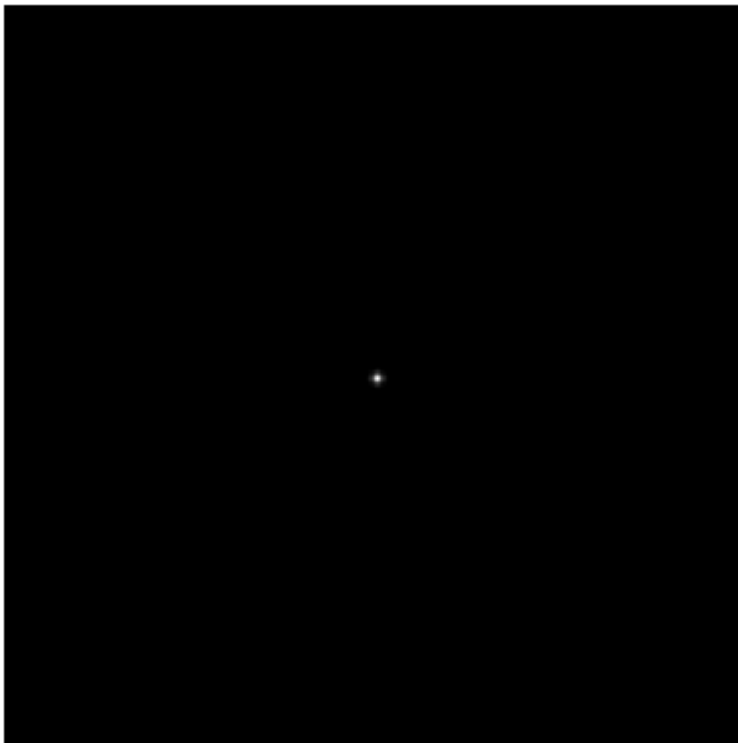
```
plt.imshow(np.array(im, dtype=np.uint8), cmap)
```



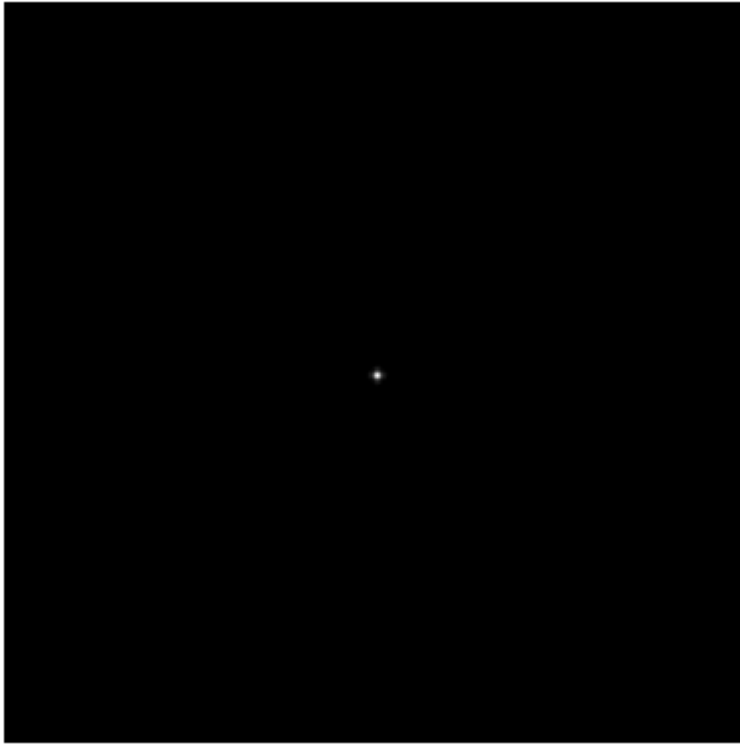
```
discgaussfft  
variance[[1.00000000e-01 9.90874049e-15]  
[9.90874049e-15 1.00000000e-01]]
```



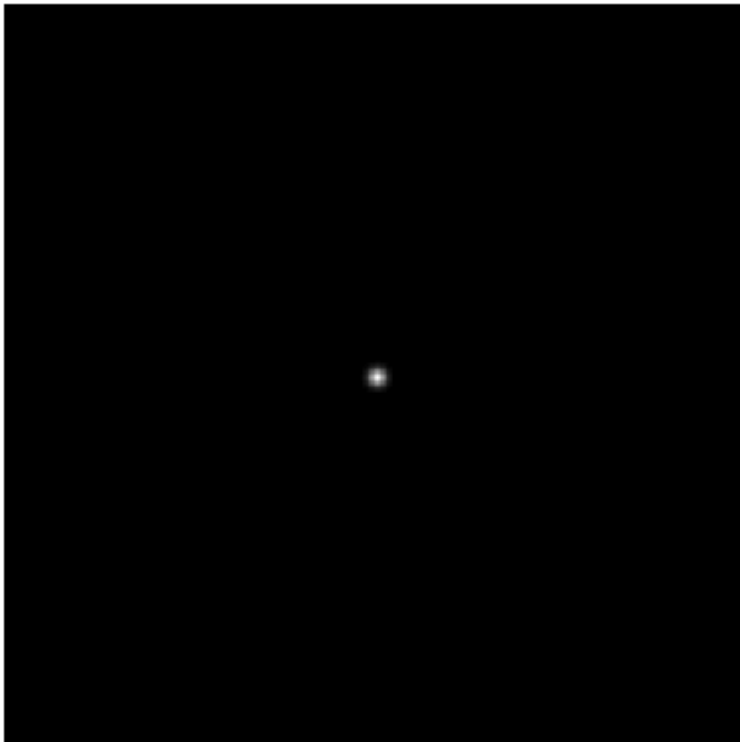
```
(128, 128) (128, 128) (128, 128)  
t: 0.3  
gaussfft  
variance[[2.81053830e-01 1.47382107e-14]  
[1.47382107e-14 2.81053830e-01]]
```



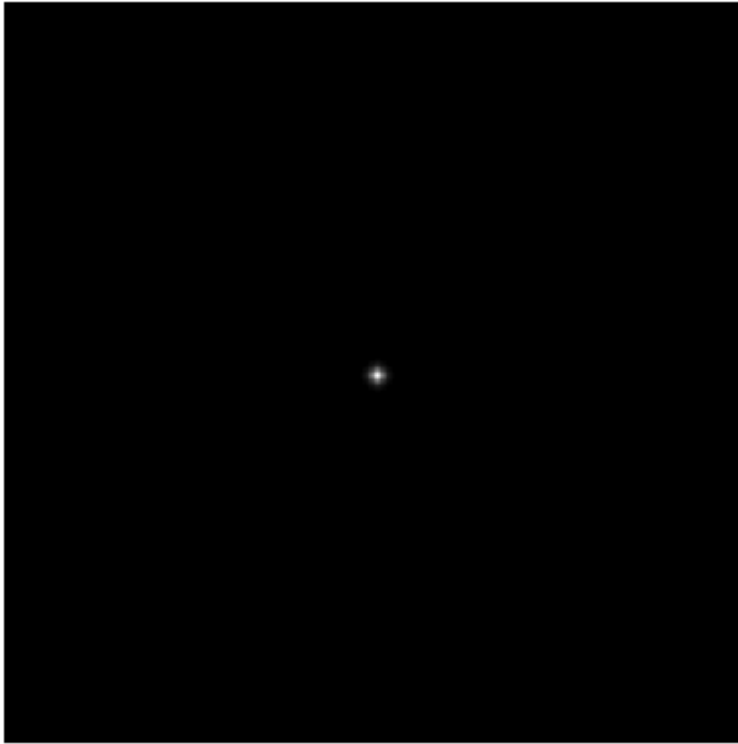
```
discgaussfft  
variance[[ 3.00000000e-01 -8.54039062e-14]  
[-8.54039062e-14 3.00000000e-01]]
```

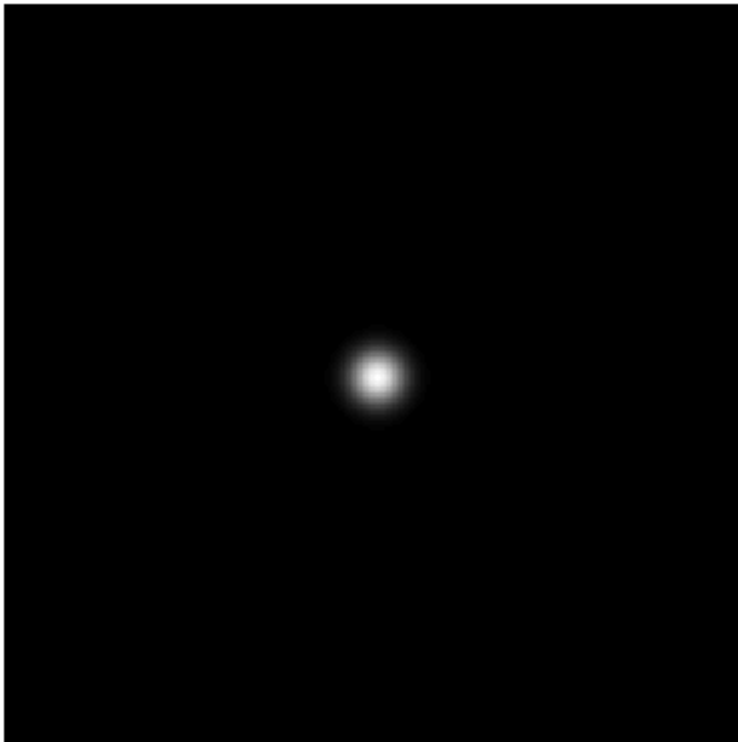
```
(128, 128) (128, 128) (128, 128)
t: 1.0
gaussfft
variance[[9.99999789e-01 6.32827124e-15]
[6.32827124e-15 9.99999789e-01]]
```



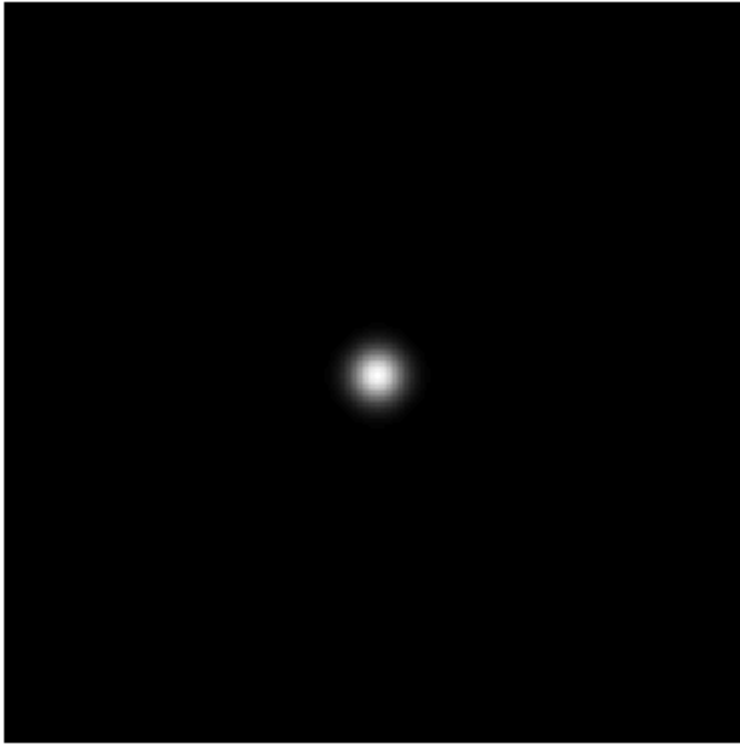
```
discgaussfft
variance[[1.00000000e+00 1.80133686e-14]
[1.80133686e-14 1.00000000e+00]]
```



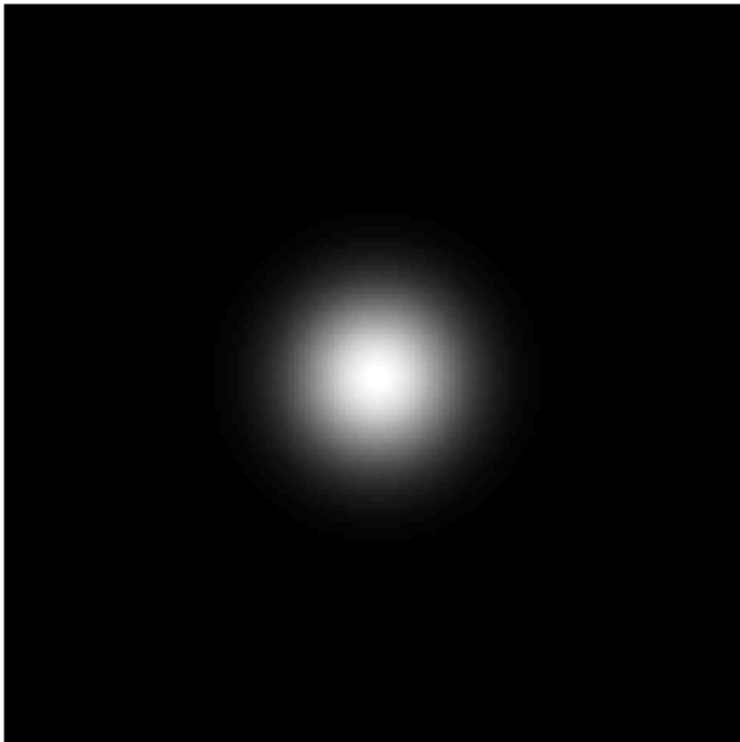
```
(128, 128) (128, 128) (128, 128)
t: 10.0
gaussfft
variance[[ 1.00000000e+01 -9.71445147e-15]
          [-9.71445147e-15  1.00000000e+01]]
```



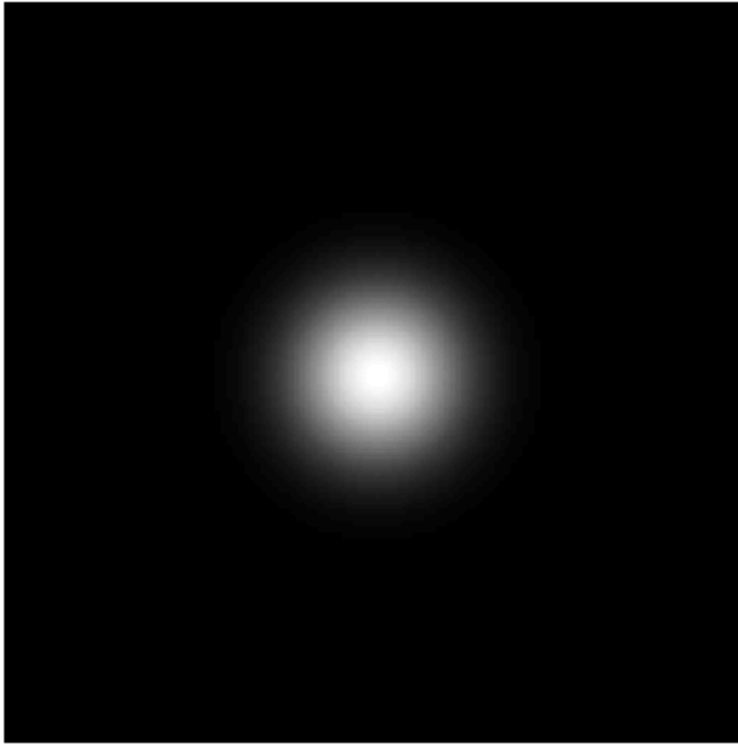
```
discgaussfft
variance[[1.00000000e+01 1.48214774e-14]
          [1.48214774e-14 1.00000000e+01]]
```



```
(128, 128) (128, 128) (128, 128)
t: 100.0
gaussfft
variance[[ 9.99999993e+01 -1.66533454e-16]
          [-1.66533454e-16  9.99999993e+01]]
```



```
discgaussfft
variance[[ 9.99999999e+01 -9.50073353e-14]
          [-9.50073353e-14  9.99999999e+01]]
```



Question 15 Are the results different from or similar to the estimated variance? How does the result correspond to the ideal continuous case? Lead: think of the relation between spatial and Fourier domains for different values of t .

Answer

In the ideal case, the variance is the product of t and the identity matrix. When $t < 1$, the variance differs from the ideal case due to incorrect Gaussian approximation and sampling. The discretized Gaussian does not estimate the continuous Gaussian, When $t > 1$, the variance matches the ideal case.

Question 16 Convolve a couple of images with Gaussian functions of different variances (like $t = 1.0, 4.0, 16.0, 64.0$ and 256.0) and present your results. What effects can you observe?

```
In [ ]: def question_16():
    image_arr = [np.load("Images-npy/phonecalc128.npy"), np.load("Images-npy/few128.npy"), np.load("Images-npy/nalloy128.npy")]
    title_arr = ["phonecalc128", "few128", "nalloy128"]
    t_arr = [1.0, 4.0, 16.0, 64.0, 256.0]

    # Create a 3x10 plot array (3 rows, 10 columns)
    fig, axes = plt.subplots(3, 10, figsize=(20, 10))

    # Loop over original images
    for i in range(3):
        axes[i, 0].imshow(image_arr[i], cmap="gray")
        axes[i, 0].set_title(title_arr[i])
        axes[i, 0].axis("off")

    # Apply Gaussian and discrete Gaussian filtering
    for j, t in enumerate(t_arr):
        gauss_filtered = gaussfft(gaussfft(image_arr[i], t), t)
        disc_gauss_filtered = discgaussfft(image_arr[i], t)
```

```

# Display Gaussian filtered image
axes[i, j + 1].imshow(np.abs(gauss_filtered), cmap="gray") # Display
axes[i, j + 1].set_title("t=" + str(t) + " (Gaussian)")
axes[i, j + 1].axis("off")

# Display discrete Gaussian filtered image
axes[i, j + 5].imshow(np.abs(disc_gauss_filtered), cmap="gray") # Display
axes[i, j + 5].set_title("t=" + str(t) + " (Discrete Gaussian)")
axes[i, j + 5].axis("off")

# Adjust spacing between subplots
plt.tight_layout()

# Show the plot
plt.show()

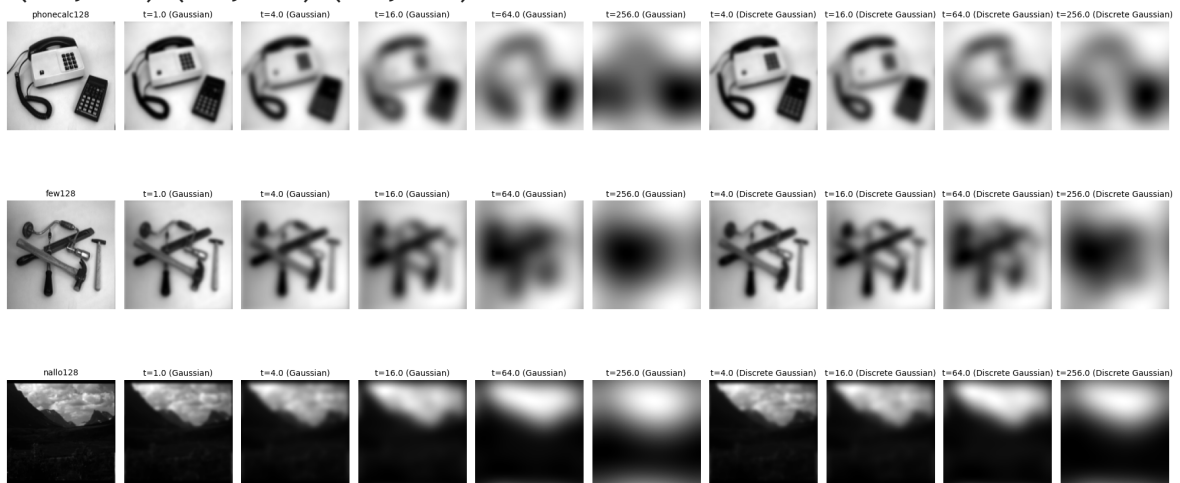
question_16()

```

```

(128, 128) (128, 128) (128, 128)
(128, 128) (128, 128) (128, 128)
(128, 128) (128, 128) (128, 128)
(128, 128) (128, 128) (128, 128)
(128, 128) (128, 128) (128, 128)
(128, 128) (128, 128) (128, 128)
(128, 128) (128, 128) (128, 128)
(128, 128) (128, 128) (128, 128)
(128, 128) (128, 128) (128, 128)
(128, 128) (128, 128) (128, 128)
(128, 128) (128, 128) (128, 128)
(128, 128) (128, 128) (128, 128)
(128, 128) (128, 128) (128, 128)
(128, 128) (128, 128) (128, 128)
(128, 128) (128, 128) (128, 128)

```



Answer

As the value of t increases, the recreated image becomes more and more blur. This is because the variance of the Gaussian filter increases and the higher frequency components are removed from the image. High frequency components correspond to the edges and contours within the image.

Section 3.1

This block contains some of the pre-work files used to understand the functions.

```
In [ ]: office = np.load("Images-npy/office256.npy")
showgrey(office)

add = gaussnoise(office, 16)
sap = sapnoise(office, 0.1, 255)

showgrey(add)
showgrey(sap)

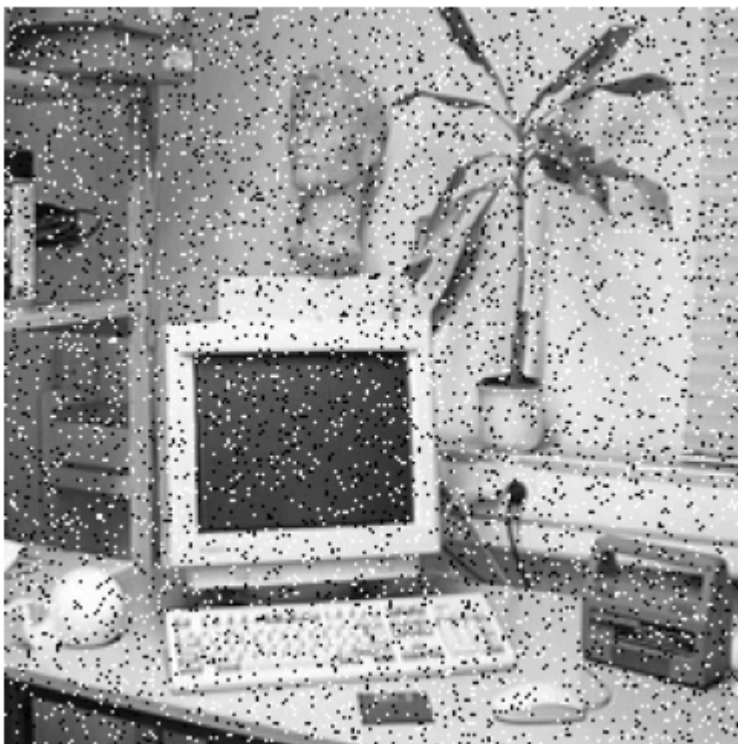
check = gaussfft(add,1)

print("Gaussian Filtering")
print("Image with Gaussian Noise")
showgrey(gaussfft(add,1),1)
showgrey(discgaussfft(add,1))
print("Image with Sap Noise")
showgrey(gaussfft(sap,1),1)
showgrey(discgaussfft(sap,2))

print("")
print("Median Filtering")
print("Image with Gaussian Noise")
showgrey(medfilt(add,1))
print("Image with Sap Noise")
showgrey(medfilt(sap,3))

print("")
print("Ideal low-pass Filtering")
print("Image with Gaussian Noise")
showgrey(ideal(add, 1))
print("Image with Sap Noise")
showgrey(ideal(sap, 0.4))
```





Gaussian Filtering
Image with Gaussian Noise



(256, 256) (256, 256) (256, 256)



Image with Sap Noise



(256, 256) (256, 256) (256, 256)



Median Filtering
Image with Gaussian Noise



Image with Sap Noise



Ideal low-pass Filtering
Image with Gaussian Noise



Image with Sap Noise



Question 17 What are the positive and negative effects for each type of filter? Describe what you observe and name the effects that you recognize. How do the results depend on the filter parameters? Illustrate with Matplotlib figure(s).

Answer

Results of Gaussian Filtering:

Gaussian filtering results in the most blurring for the image. Most of the Gaussian noise generated is smoothed off. As the value of t increases, the image becomes more blurred and the edges are lost. When applied to the salt and pepper noise, the noise is blurred

out along with the edges. Noise not removed from the image, making the image look more noisy. As the value of t increases, the image is more blurred.

Results of Median Filtering:

This filter seems most effective and removes all the salt and pepper noise. The edges of all the objects are also preserved in both images for small window sizes. The "add" image seems noisier than the "sap" image. Some of the contours are lost as the image looks more like a painting (this applies to the "sap" image). As the window size increases, more edges and contours are lost.

Results of Low Pass Filtering:

This filter seems more effective for the gaussian noise than the salt and pepper noise. It is used to remove the frequencies about the cutoff frequency which means that high frequency noise can be removed. In the case of the salt and pepper noise, the noise is smoothed but not removed from the image. Both images are smoothed. As the cutoff frequency increases, fewer high frequency components are removed from the image.

Question 18 What conclusions can you draw from comparing the results of the respective methods?

Answer

Gaussian filtering is most effective when applied to the images where only Gaussian noise is added. Median filtering is most effective when applied to the images where the Salt and Pepper noise is added. Applying the low pass filter can result in the ringing effect.

Section 3.2

This block contains some of the pre-work files used to understand the functions.

```
In [ ]: img = np.load("Images-npy/phonecalc256.npy")
smoothing = img
N = 5
f = plt.figure()
f.subplots_adjust(wspace=0, hspace=0)
for i in range(N):
    if i>0: # generate subsampled versions
        img = rawsubsample(img)
        smoothing = gaussfft(smoothing,1)
        smoothing = rawsubsample(smoothing)
    f.add_subplot(2, N, i + 1)
    showgrey(img, False)
    f.add_subplot(2, N, i + N + 1)
    showgrey(smoothing, False)

plt.show()
```



```
In [ ]: img = np.load("Images-npy/phonecalc256.npy")
        smoothing = img
        N = 5
        f = plt.figure()
        f.subplots_adjust(wspace=0, hspace=0)
        for i in range(N):
            if i>0: # generate subsampled versions
                img = rawsubsample(img)
                smoothing = ideal(smoothing, 0.4)
                smoothing = rawsubsample(smoothing)
            f.add_subplot(2, N, i + 1)
            showgrey(img, False)
            f.add_subplot(2, N, i + N + 1)
            showgrey(smoothing, False)

        plt.show()
```



Question 19 What effects do you observe when subsampling the original image and the smoothed variants? Illustrate both filters with the best results found for iteration $i = 4$.

Answer

In the fourth iteration ($i = 4$), the image becomes more blurred and the definition of the edges is lost. The Gaussian filter smoothens the image more and is more appealing. Low pass filter causes the image to still have a lot of noise in the fourth iteration when compared to the Gaussian. Subsampling alone leads to the presence of aliasing artifacts while applying the smoothing filters remove most of the artifacts and give a less noisy image.

Question 20 What conclusions can you draw regarding the effects of smoothing when combined with subsampling? Hint: think in terms of frequencies and side effects.

Answer

Smoothing the image first before subsampling is done to avoid the aliasing effect. When applying a smoothing filter, the high frequency components of the image are removed. Upon subsampling, every alternate pixel is retained. This reduces the overall frequency of the image and avoids artifacts due to the mirroring of high frequencies from being present in the image. This results in lowered information loss. Although some information is lost during smoothing, more is lost due to the aliasing effect if smoothing is not applied.