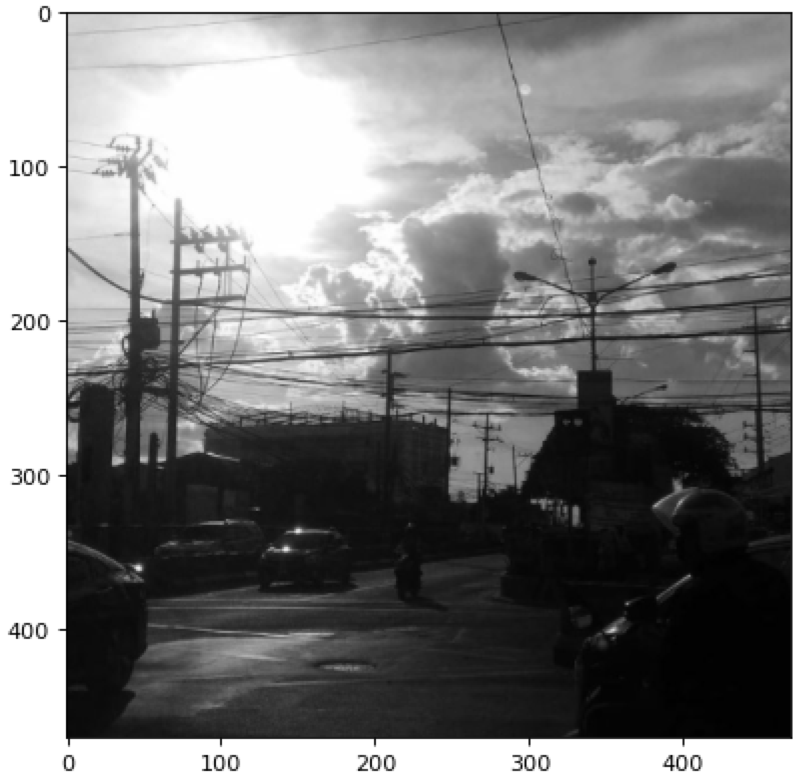
Ref: https://towardsdatascience.com/image-processing-with-python-application-of-fourier-transformation-5a8584dc175b required Python libraries

In [13]:
```
import numpy as np
import matplotlib.pyplot as plt
from skimage.io import imread, imshow
from skimage.color import rgb2hsv, rgb2gray, rgb2yuv
from skimage import color, exposure, transform
from skimage.exposure import equalize_hist
```

load the image

In [2]:
```
dark_image = imread('against_the_light.png')
```

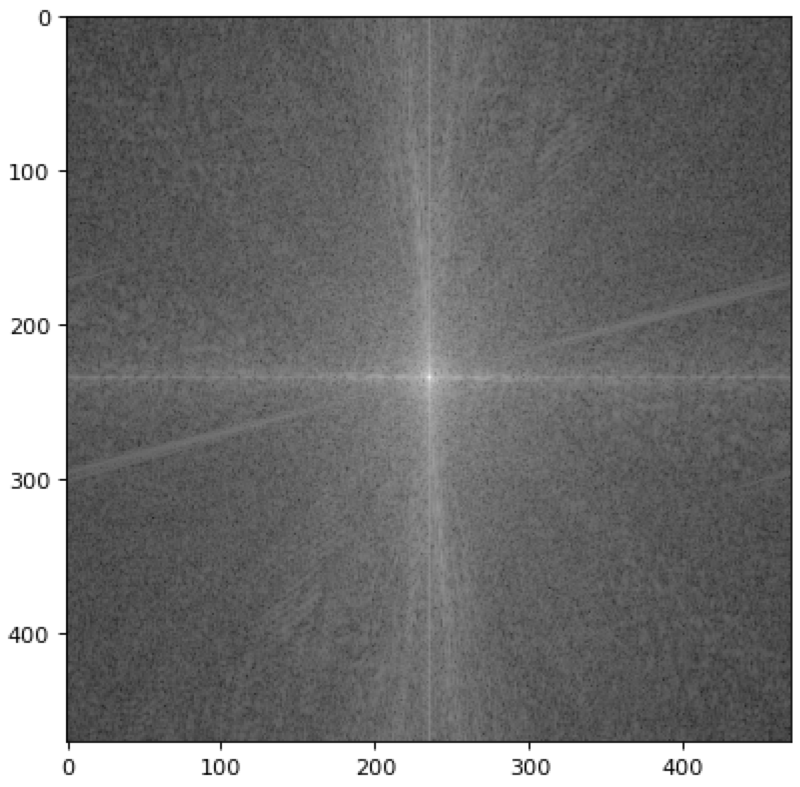
In [3]:
```
dark_image_grey = rgb2gray(dark_image)
plt.figure(num=None, figsize=(8, 6), dpi=80)
plt.imshow(dark_image_grey, cmap='gray');
```



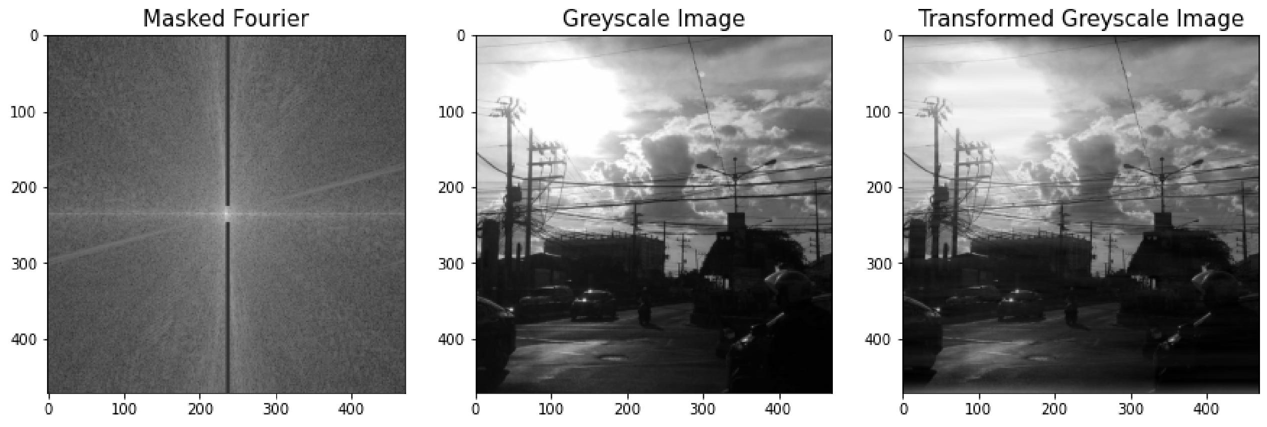
The image we will be using is the one above. It as image of a street taken when the sun was facing directly at the camera. For this simple exercise we shall try to find a way to eliminate (or least drastically reduce) the powerlines in the back. In the image we can see two very clear distortions. The white vertical and horizontal lines refer to the sharp horizontal and vertical elements of the image. Let us see what happens if we mask one of them.

use the fft function

In [4]:
```
dark_image_grey_fourier = np.fft.fftshift(np.fft.fft2(dark_image_grey))
plt.figure(num=None, figsize=(8, 6), dpi=80)
plt.imshow(np.log(abs(dark_image_grey_fourier)), cmap='gray');
```



```
In [7]: def fourier_masker(image, i):
            f_size = 15
            dark_image_grey_fourier = np.fft.fftshift(np.fft.fft2(rgb2gray(image)))
            dark_image_grey_fourier[:225, 235:240] = i
            dark_image_grey_fourier[-225:,235:240] = i
            fig, ax = plt.subplots(1,3,figsize=(15,15))
            ax[0].imshow(np.log(abs(dark_image_grey_fourier)), cmap='gray')
            ax[0].set_title('Masked Fourier', fontsize = f_size)
            ax[1].imshow(rgb2gray(image), cmap = 'gray')
            ax[1].set_title('Greyscale Image', fontsize = f_size);
            ax[2].imshow(abs(np.fft.ifft2(dark_image_grey_fourier)),
                             cmap='gray')
            ax[2].set_title('Transformed Greyscale Image',
                             fontsize = f_size);

        fourier_masker(dark_image, 1)
```



Python programming language supports negative indexing of arrays, something which is not available in arrays in most other programming languages. This means that the index value of -1 gives the last element, and -2 gives the second last element of an array. The negative indexing starts from where the array ends. This means that the last element of the array is the first element in the negative indexing which is -1.

Example: arr = [10, 20, 30, 40, 50] print (arr[-1]) print (arr[-2])

Output: 50 40

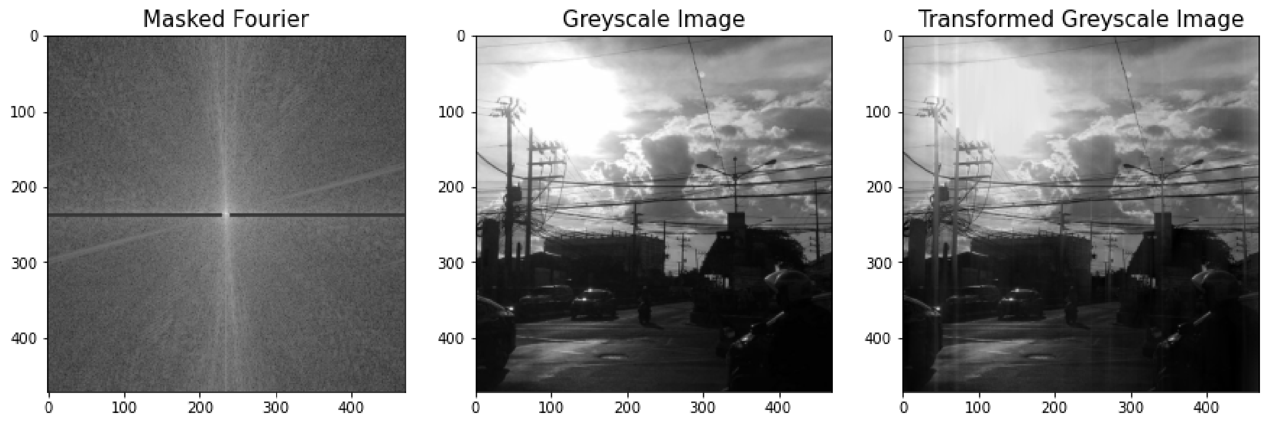
In [14]:
```
arr = [10, 20, 30, 40, 50]
print (arr[-2:])
```

```
[40, 50]
```

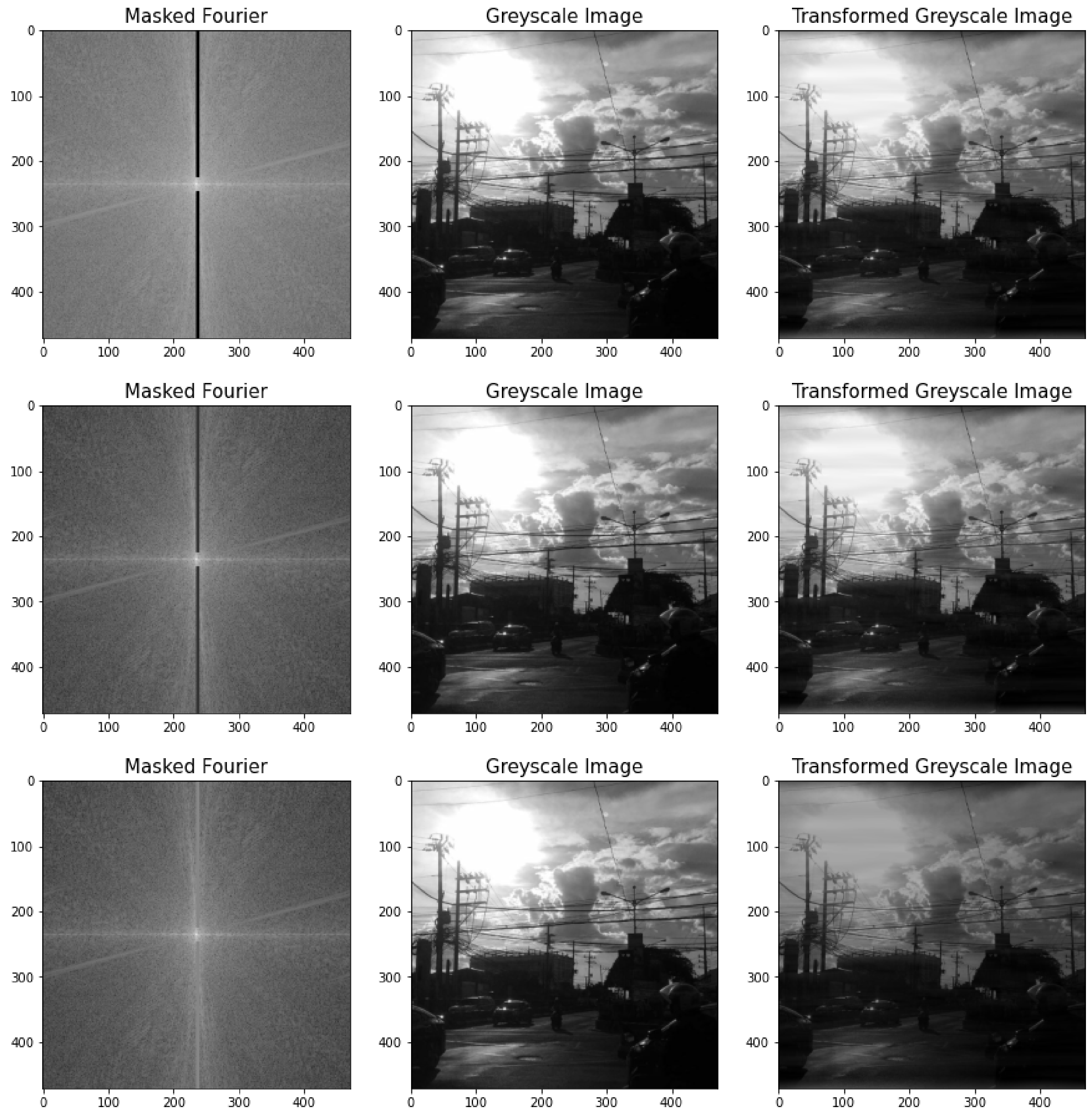
https://numpy.org/doc/stable/reference/routines.fft.html

> > > freqs = np.fft.fftfreq(10, 0.1) freqs array([0., 1., 2., 3., 4., -5., -4., -3., -2., -1.]) np.fft.fftshift(freqs) array([-5., -4., -3., -2., -1., 0., 1., 2., 3., 4.])

In [9]:
```
def fourier_masker_hor(image, i):
    f_size = 15
    dark_image_grey_fourier = np.fft.fftshift(np.fft.fft2(rgb2gray(image)))
    dark_image_grey_fourier[235:240, :230] = i
    dark_image_grey_fourier[235:240,-230:] = i
    fig, ax = plt.subplots(1,3,figsize=(15,15))
    ax[0].imshow(np.log(abs(dark_image_grey_fourier)), cmap='gray')
    ax[0].set_title('Masked Fourier', fontsize = f_size)
    ax[1].imshow(rgb2gray(image), cmap = 'gray')
    ax[1].set_title('Greyscale Image', fontsize = f_size);
    ax[2].imshow(abs(np.fft.ifft2(dark_image_grey_fourier)),
                     cmap='gray')
    ax[2].set_title('Transformed Greyscale Image',
                     fontsize = f_size);
fourier_masker_hor(dark_image, 1)
```



In [11]:
```
def fourier_iterator(image, value_list):
    for i in value_list:
        fourier_masker_ver(image, i)

fourier_iterator(dark_image, [0.001, 1, 100])
```

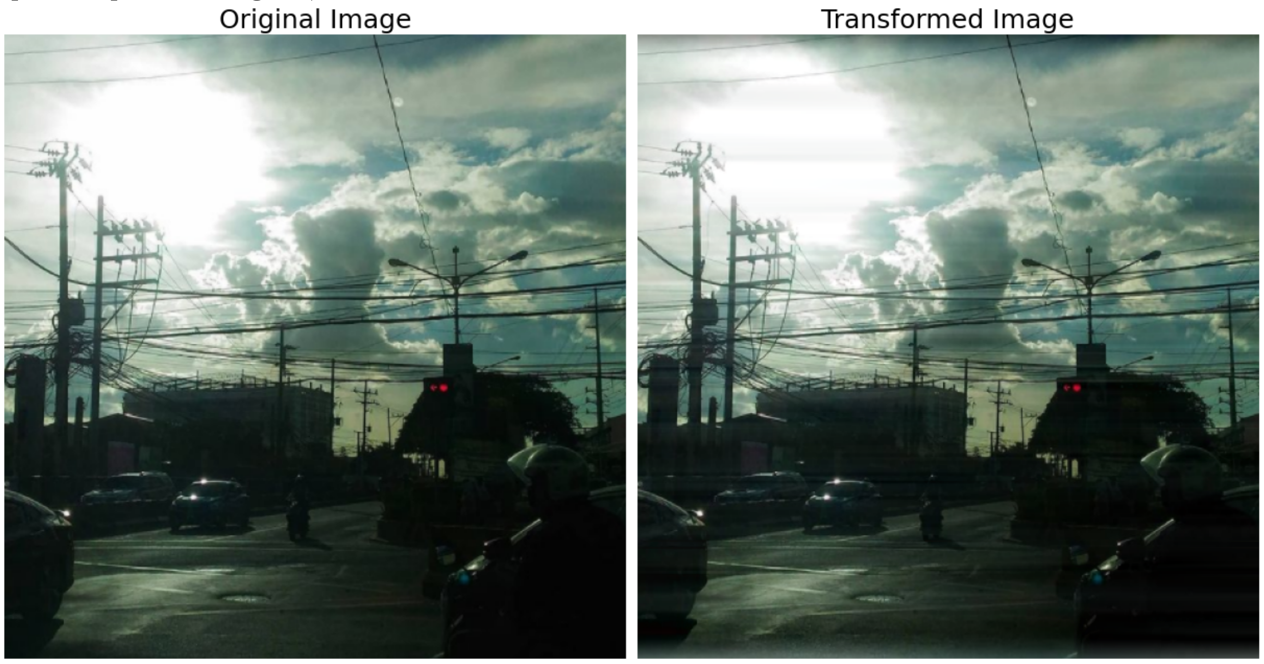


In [10]:

```
def fourier_transform_rgb(image):
    f_size = 25
    transformed_channels = []
    for i in range(3):
        rgb_fft = np.fft.fftshift(np.fft.fft2((image[:, :, i])))
        rgb_fft[:225, 235:237] = 1
        rgb_fft[-225:,235:237] = 1
        transformed_channels.append(abs(np.fft.ifft2(rgb_fft)))

    final_image = np.dstack([transformed_channels[0].astype(int),
                             transformed_channels[1].astype(int),
                             transformed_channels[2].astype(int)])

    fig, ax = plt.subplots(1, 2, figsize=(17,12))
    ax[0].imshow(image)
    ax[0].set_title('Original Image', fontsize = f_size)
    ax[0].set_axis_off()

    ax[1].imshow(final_image)
    ax[1].set_title('Transformed Image', fontsize = f_size)
    ax[1].set_axis_off()

    fig.tight_layout()
```

In [12]:
```
fourier_transform_rgb(dark_image)
```

```
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or
[0..255] for integers).
```



In [ ]:
```

```