Exp 1

Name: Drumil Kotecha

Roll no. C052

Aim: Write a code to detect the edges of an image

In [2]:
```
import numpy as np
import cv2
import matplotlib.pyplot as plt
from skimage import data
```

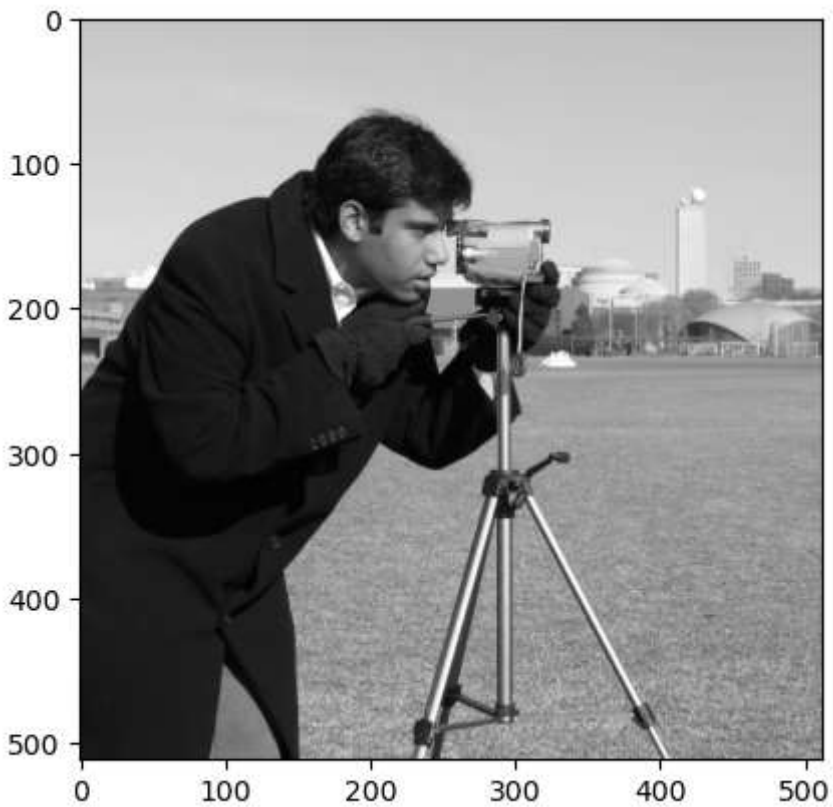
```
C:\Users\admin\anaconda3\Lib\site-packages\paramiko\transport.py:219: CryptographyDep
recationWarning: Blowfish has been deprecated
  "class": algorithms.Blowfish,
```

In [3]:
```
img = data.camera()
img.shape
```

Out[3]:
```
(512, 512)
```

In [4]:
```
plt.imshow(img, cmap='gray')
```

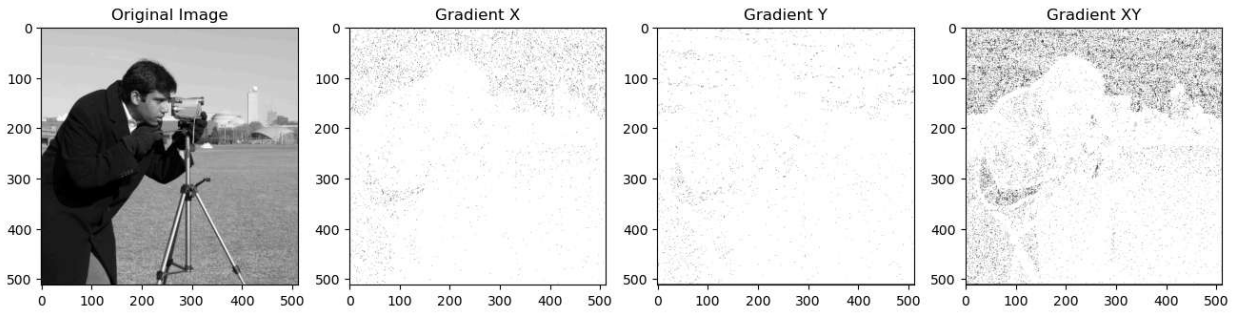
Out[4]:
```
<matplotlib.image.AxesImage at 0x2e94a4ded50>
```



In [5]:
```
grad_x = cv2.Sobel(img, ddepth=cv2.CV_32F, dx=1,dy=0, ksize=9)
grad_y = cv2.Sobel(img, ddepth=cv2.CV_32F, dx=0,dy=1, ksize=9)
grad_xy = cv2.Sobel(img, ddepth=cv2.CV_32F, dx=1, dy=1, ksize=9)
```

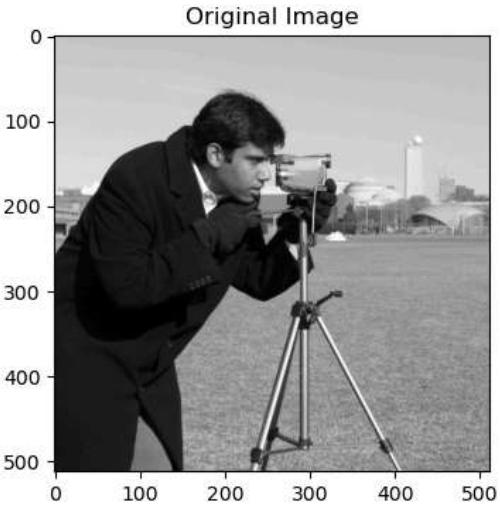
In [6]:
```
grad_x_abs = cv2.convertScaleAbs(grad_x)
grad_y_abs = cv2.convertScaleAbs(grad_y)
grad_xy_abs = cv2.convertScaleAbs(grad_xy)
```

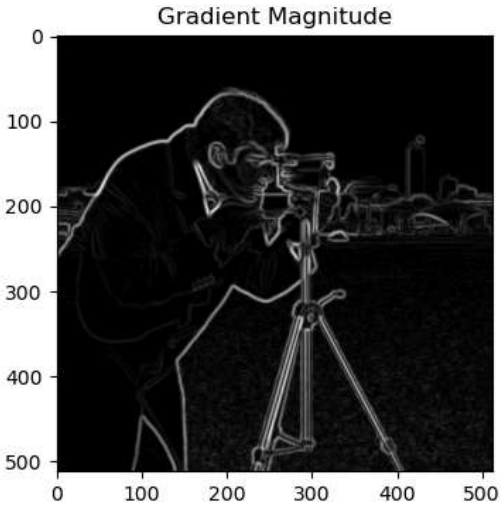
In [7]:
```
plt.figure(figsize=(16, 4))
plt.subplot(1,4,1)
plt.imshow(img, cmap='gray')
plt.title('Original Image')

plt.subplot(1,4,2)
plt.imshow(grad_x_abs, cmap='gray')
plt.title('Gradient X')

plt.subplot(1,4,3)
plt.imshow(grad_y_abs, cmap='gray')
plt.title('Gradient Y')

plt.subplot(1,4,4)
plt.imshow(grad_xy_abs, cmap='gray')
plt.title('Gradient XY')

plt.show()
```



In [8]:
```
grad_mag = np.sqrt(grad_x**2 + grad_y**2)
# grad_mag = grad_mag/np.max(grad_mag)
plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
plt.imshow(img, cmap='gray')
plt.title('Original Image')
plt.subplot(1, 2, 2)
plt.title('Gradient Magnitude')
plt.imshow(grad_mag, cmap='gray')
```

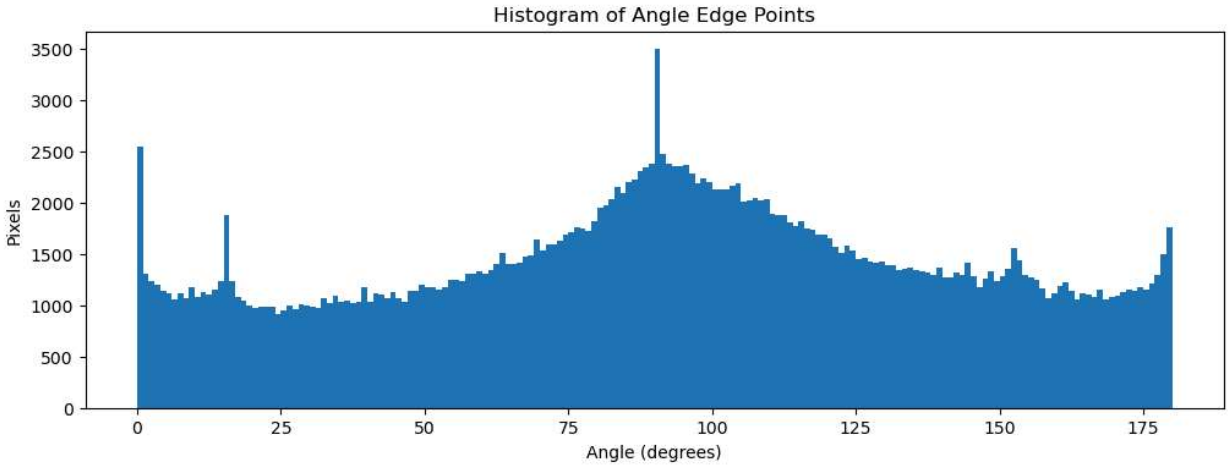
Out[8]:
```
<matplotlib.image.AxesImage at 0x2e94cd3ec50>
```





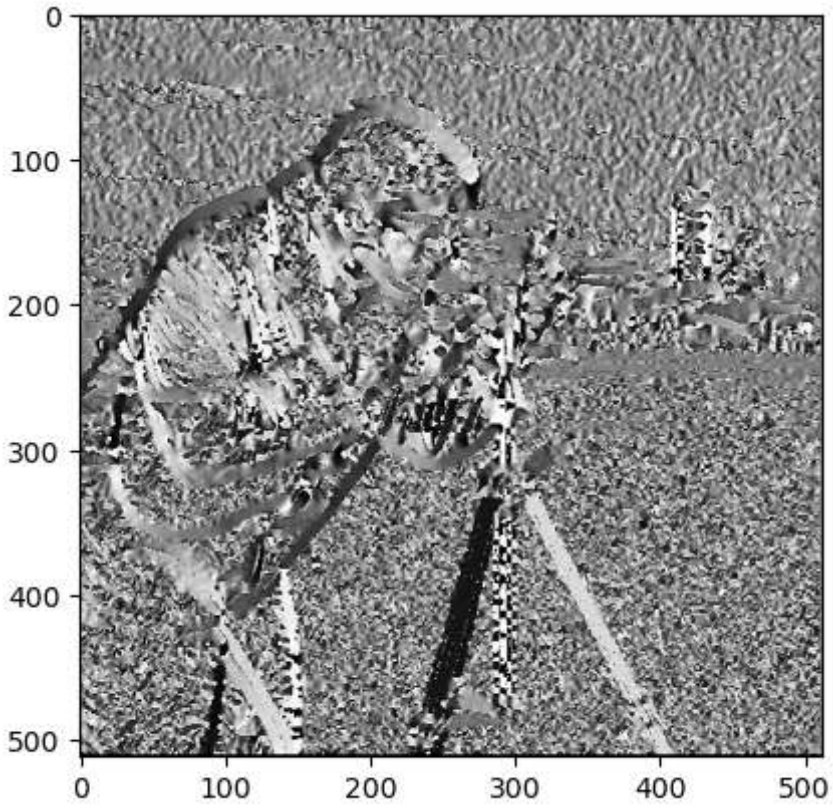
In [9]:
```
angle = (np.arctan2(grad_y, grad_x)*180/np.pi)%180
plt.figure(figsize=(12, 4))
[r,c] = img.shape
angle1D = np.reshape(angle, [r*c,1])
plt.hist(angle1D, bins=180)
plt.title('Histogram of Angle Edge Points')
plt.xlabel('Angle (degrees)')
plt.ylabel('Pixels')
```

Out[9]:
```
Text(0, 0.5, 'Pixels')
```



In [10]:
```
plt.imshow(angle, cmap="gray")
```

Out[10]:
```
<matplotlib.image.AxesImage at 0x2e94d176f90>
```



Conclusion

1. Sobel filter is used identify horizontal, vertical, diagonal and magnitude of gradients at each pixel of the given image.
2. For each pixel, histogram of angle of each pixel is plotted. It is observed that most of the pixels are having their gradients along 0, 45, 90, 135.
3. If size of filter is increased from 3,3 to 9,9 and 31,31 - then it is observed that pixels along small edges are not captured or highlighted by filter of large size. However, large edges with more possibilites of angles can be identified by these filters.
4. Sobel filter of large size can used to reduce the effect of small edges which are considered as noisy objects.

In [12]:
```
leo = cv2.imread('leo.webp', cv2.IMREAD_GRAYSCALE)
plt.imshow(leo, cmap='gray')
```

Out[12]:
```
<matplotlib.image.AxesImage at 0x2e94f65ef90>
```

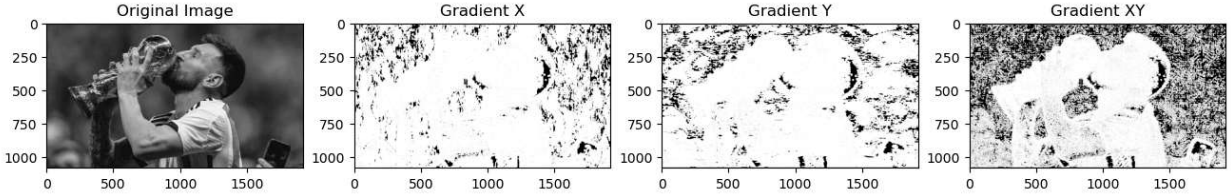
In [13]:
```
leo.shape
```

Out[13]:
```
(1080, 1920)
```

In [14]:
```
grad_x = cv2.Sobel(leo, ddepth=cv2.CV_32F, dx=1,dy=0, ksize=9)
grad_y = cv2.Sobel(leo, ddepth=cv2.CV_32F, dx=0,dy=1, ksize=9)
grad_xy = cv2.Sobel(leo, ddepth=cv2.CV_32F, dx=1, dy=1, ksize=9)
```

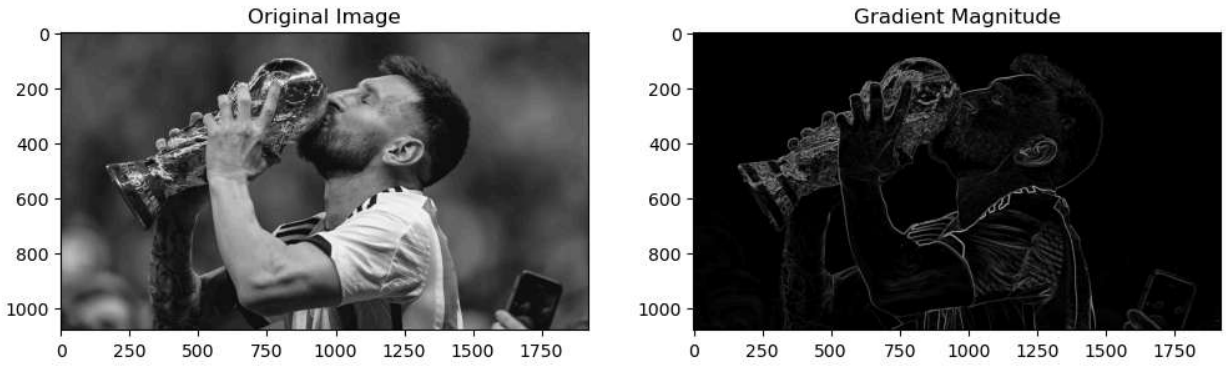
In [15]:
```
grad_x_abs = cv2.convertScaleAbs(grad_x)
grad_y_abs = cv2.convertScaleAbs(grad_y)
grad_xy_abs = cv2.convertScaleAbs(grad_xy)
```

In [16]:
```
plt.figure(figsize=(16, 4))
plt.subplot(1,4,1)
plt.imshow(leo, cmap='gray')
plt.title('Original Image')

plt.subplot(1,4,2)
plt.imshow(grad_x_abs, cmap='gray')
plt.title('Gradient X')

plt.subplot(1,4,3)
plt.imshow(grad_y_abs, cmap='gray')
plt.title('Gradient Y')

plt.subplot(1,4,4)
plt.imshow(grad_xy_abs, cmap='gray')
plt.title('Gradient XY')

plt.show()
```



In [17]:
```
grad_mag = np.sqrt(grad_x**2 + grad_y**2)
 # grad_mag = grad_mag/np.max(grad_mag)
plt.figure(figsize=(12, 4))

plt.subplot(1, 2, 1)
plt.imshow(leo, cmap='gray')
plt.title('Original Image')
plt.subplot(1, 2, 2)
plt.title('Gradient Magnitude')
plt.imshow(grad_mag, cmap='gray')
```

Out[17]:
```
<matplotlib.image.AxesImage at 0x2e94a67c390>
```



In [19]:
```
angle = (np.arctan2(grad_y, grad_x)*180/np.pi)%180

plt.figure(figsize=(12, 4))
[r,c] = leo.shape

angle1D = np.reshape(angle, [r*c,1])
plt.hist(angle1D, bins=180)
plt.title('Histogram of Angle Edge Points')
plt.xlabel('Angle (degrees)')
plt.ylabel('Pixels')
```

Out[19]:
```
Text(0, 0.5, 'Pixels')
```