

Library used and their work

#Opencv

Image/video I/O, processing, display (core, imgproc, highgui)

Object/feature detection (objdetect, features2d, nonfree)

Geometry-based monocular or stereo computer vision (calib3d, stitching, videostab)

#Numpy

An array object of arbitrary homogeneous items

Fast mathematical operations over arrays

Linear Algebra, Fourier Transforms, Random Number Generation

#Matplotlib

Plotting library

data visualization and graphical plotting library for Python

its numerical extension NumPy

#Sckit-image

Image processing

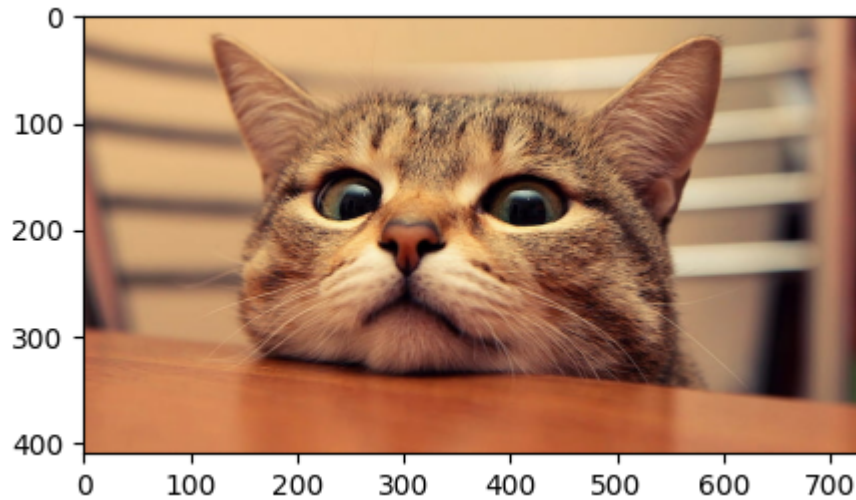
#Hog

Histogram of Oriented Gradients

```
In [451... import cv2
import numpy as np
import matplotlib.pyplot as plt
import warnings
from skimage.feature import hog
```

```
warnings.filterwarnings("ignore")
%matplotlib inline
```

```
In [452... plt.figure(figsize=(5,3))
img = plt.imread('Cute.jpg')
plt.imshow(img)
plt.grid(False)
plt.show()
```



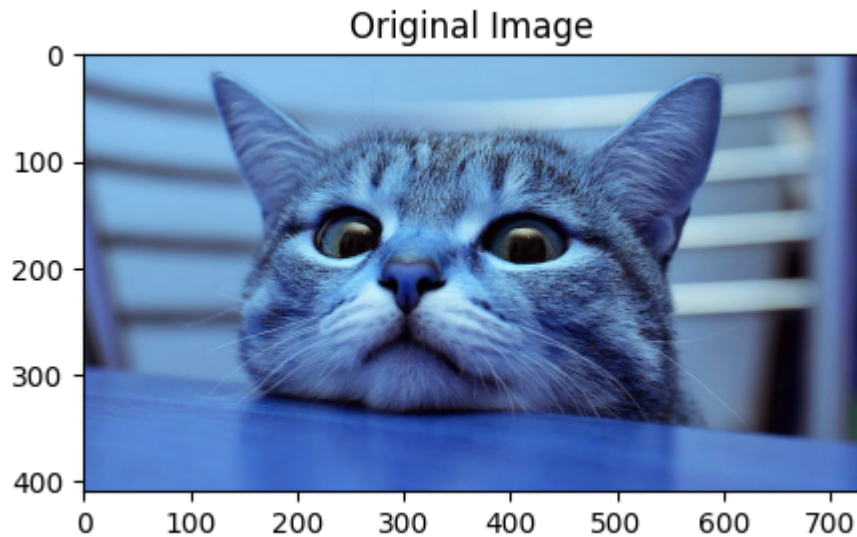
```
In [453... len(img.shape)
```

Out[453]: 3

```
In [454... img.shape
```

Out[454]: (410, 728, 3)

```
In [455... def catImageshow(imageTitle,image):
    color_convert = cv2.cvtColor(image,cv2.COLOR_BGR2RGB)
    plt.figure(figsize=(5,3))
    plt.imshow(color_convert)
    plt.title(imageTitle)
    plt.show()
catImageshow("Original Image",img)
```



In [456... `cv2.split(img)`

```
Out[456]: (array([[234, 234, 234, ..., 98, 87, 80],
                  [234, 234, 234, ..., 98, 87, 80],
                  [234, 234, 234, ..., 98, 88, 80],
                  ...,
                  [199, 199, 199, ..., 185, 185, 184],
                  [198, 198, 200, ..., 186, 186, 185],
                  [198, 198, 200, ..., 186, 186, 185]], dtype=uint8),
          array([[189, 189, 189, ..., 52, 45, 39],
                  [189, 189, 189, ..., 53, 45, 39],
                  [189, 189, 189, ..., 54, 46, 40],
                  ...,
                  [119, 119, 119, ..., 102, 102, 101],
                  [119, 119, 121, ..., 101, 101, 100],
                  [119, 119, 121, ..., 101, 101, 100]], dtype=uint8),
          array([[134, 134, 134, ..., 55, 47, 43],
                  [134, 134, 134, ..., 56, 47, 43],
                  [134, 134, 134, ..., 55, 47, 41],
                  ...,
                  [ 82,  82,  82, ...,  58,  58,  57],
                  [ 80,  80,  80, ...,  60,  60,  59],
                  [ 80,  80,  80, ...,  60,  60,  59]], dtype=uint8))
```

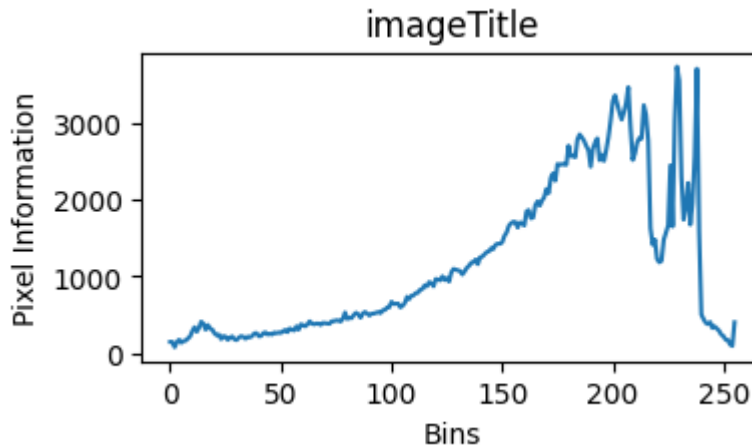
```
In [457... def imageHistogram(image, imageTitle, mask = None):
    color_Channel=cv2.split(img)
    color_com = ('b', 'g', 'r')
    plt.figure(figsize = (4, 2))
    plt.title(imageTitle)
    plt.xlabel("Bins")
    plt.ylabel("Pixel Information")
    for (color_Channel, color_com) in zip(color_Channel, color_com):
        histogram = cv2.calcHist([color_Channel], [0], mask, [256], [0, 256])
        plt.plot(histogram, color = color_com)
        plt.xlim([0,256])
    plt.show()
```

```
In [458... image_argument = {"Image": "Cute.jpg"}
```

```

image = cv2.imread(image_argument["Image"])
image= cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
smamplehisto = cv2.calcHist([image],[0], None, [256], [0, 256])
plt.figure(figsize = (4, 2))
plt.title("imageTitle")
plt.xlabel("Bins")
plt.ylabel("Pixel Information")
plt.plot(smamplehisto)
plt.show()

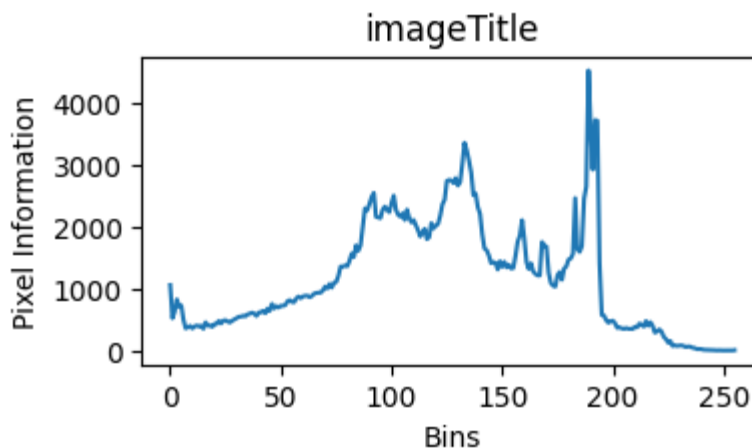
```



```

In [459... image_argument = {"Image":"Cute.jpg"}
image = cv2.imread(image_argument["Image"])
image= cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
smamplehisto = cv2.calcHist([image],[1], None, [256], [0, 256])
plt.figure(figsize = (4, 2))
plt.title("imageTitle")
plt.xlabel("Bins")
plt.ylabel("Pixel Information")
plt.plot(smamplehisto)#Line
plt.show()

```

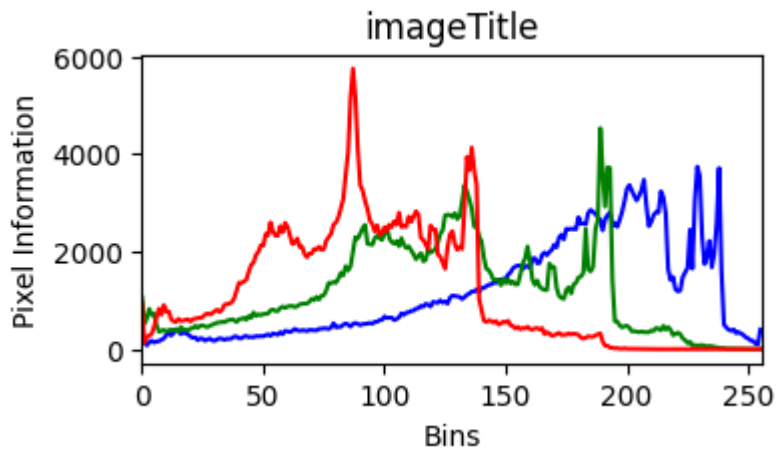


```

In [460... color_Channel = cv2.split(image)
color_com = ('b', 'g', 'r')
plt.figure(figsize = (4, 2))
plt.title("imageTitle")
plt.xlabel("Bins")

```

```
plt.ylabel("Pixel Information")
for (color_Channel, color_com) in zip(color_Channel, color_com):
    histogram = cv2.calcHist([color_Channel], [0],None, [256], [0, 256])
    plt.plot(histogram, color = color_com)
    plt.xlim([0, 256])
    #plt.show()
```

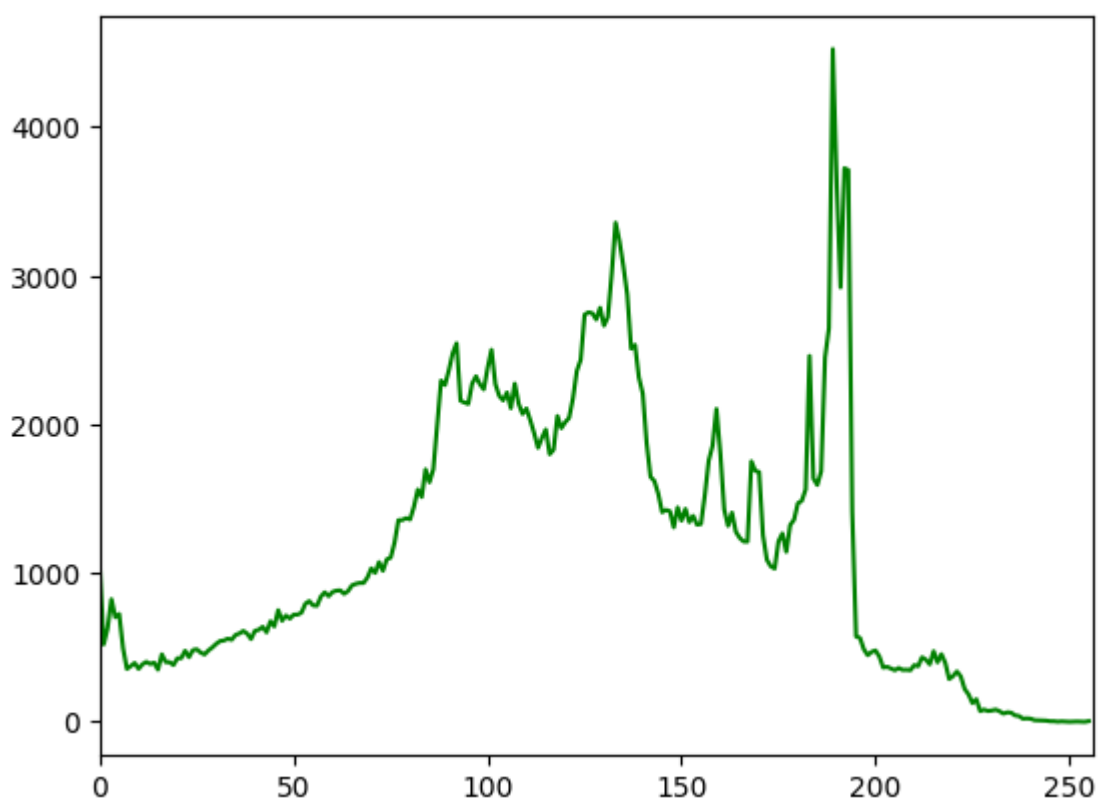
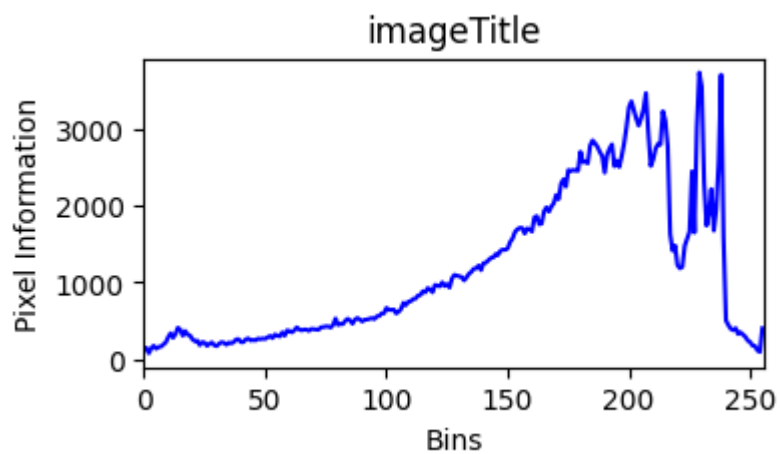


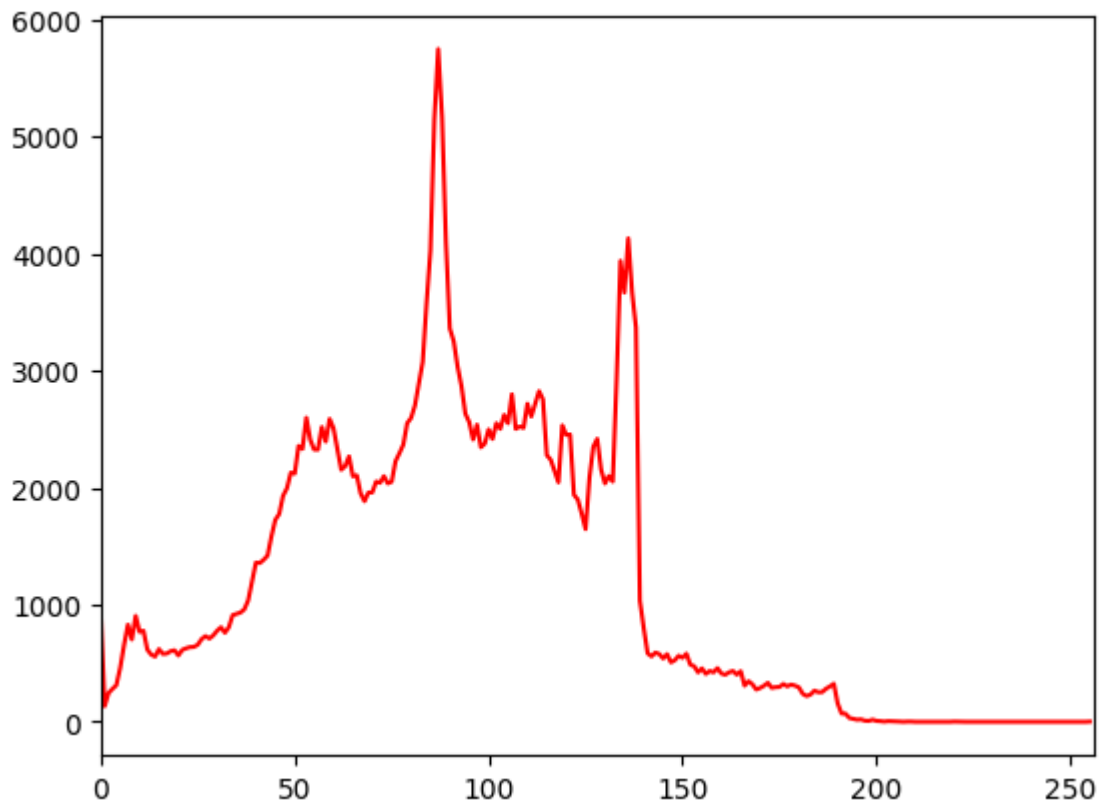
In [461... `cv2.split(image)`

```
Out[461]: (array([[234, 234, 234, ..., 98, 87, 80],
                  [234, 234, 234, ..., 98, 87, 80],
                  [234, 234, 234, ..., 98, 88, 80],
                  ...,
                  [199, 199, 199, ..., 185, 185, 184],
                  [198, 198, 200, ..., 186, 186, 185],
                  [198, 198, 200, ..., 186, 186, 185]], dtype=uint8),
          array([[189, 189, 189, ..., 52, 45, 39],
                  [189, 189, 189, ..., 53, 45, 39],
                  [189, 189, 189, ..., 54, 46, 40],
                  ...,
                  [119, 119, 119, ..., 102, 102, 101],
                  [119, 119, 121, ..., 101, 101, 100],
                  [119, 119, 121, ..., 101, 101, 100]], dtype=uint8),
          array([[134, 134, 134, ..., 55, 47, 43],
                  [134, 134, 134, ..., 56, 47, 43],
                  [134, 134, 134, ..., 55, 47, 41],
                  ...,
                  [ 82,  82,  82, ...,  58,  58,  57],
                  [ 80,  80,  80, ...,  60,  60,  59],
                  [ 80,  80,  80, ...,  60,  60,  59]], dtype=uint8))
```

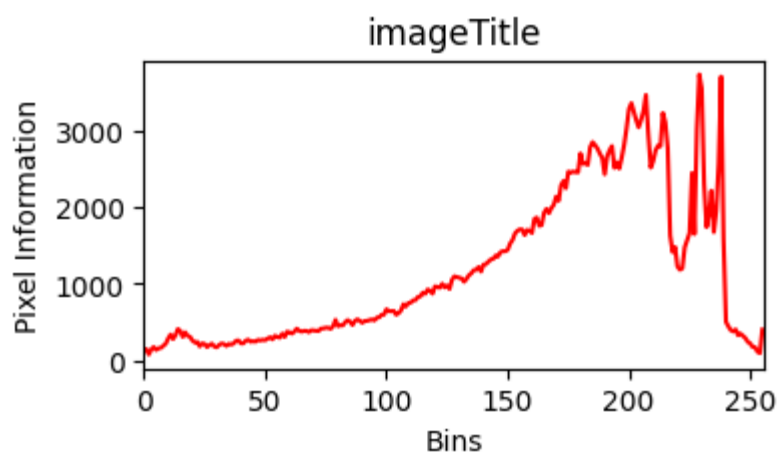
In [462... `color_Channel = cv2.split(image)`
`color_com = ('b', 'g', 'r')`
`plt.figure(figsize = (4, 2))`
`plt.title("imageTitle")`
`plt.xlabel("Bins")`
`plt.ylabel("Pixel Information")`
`for (color_Channel, color_com) in zip(color_Channel, color_com):`
 `histogram = cv2.calcHist([color_Channel], [0],None, [256], [0, 256])`
 `plt.plot(histogram, color = color_com)`

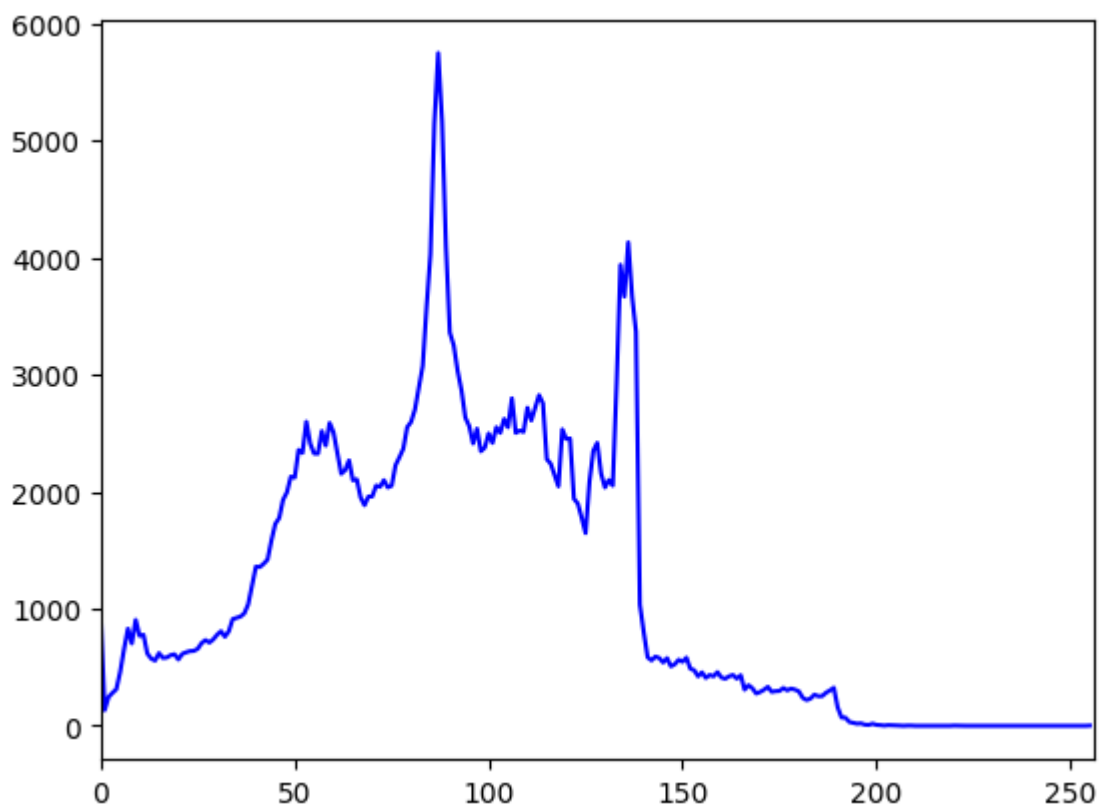
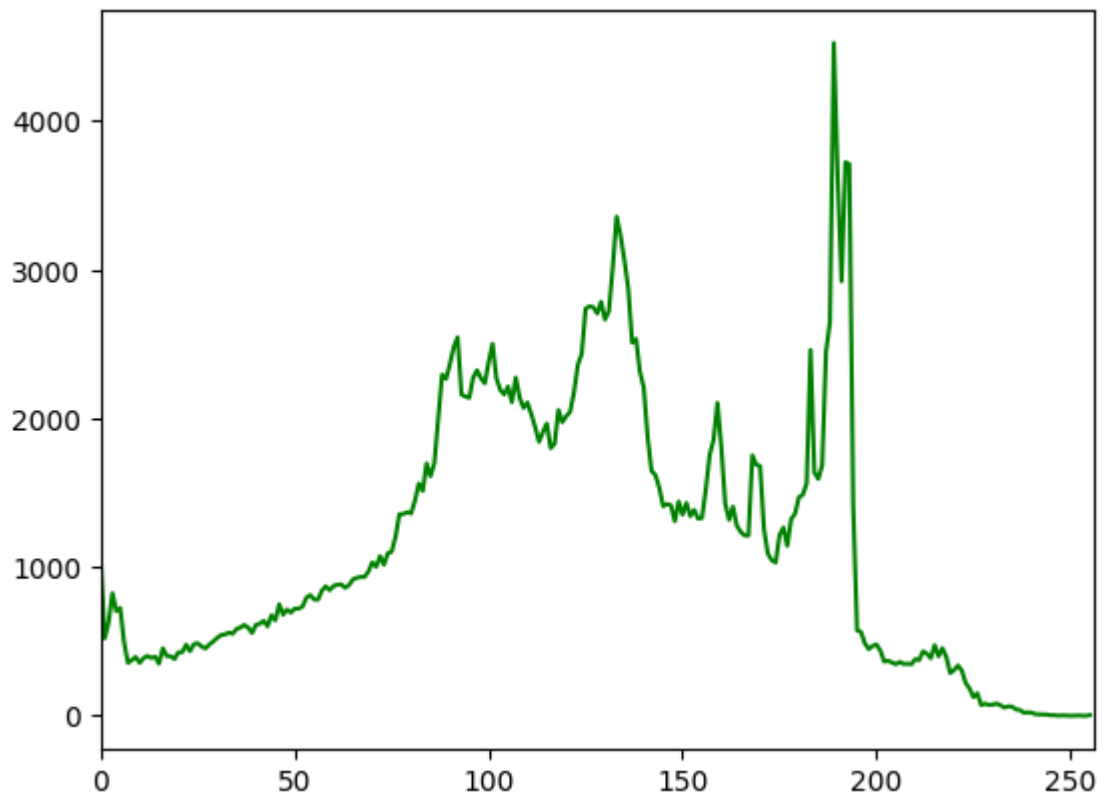
```
plt.xlim([0, 256])  
plt.show()
```





```
In [463... color_Channel = cv2.split(image)
color_com = ('r', 'g', 'b')
plt.figure(figsize = (4, 2))
plt.title("imageTitle")
plt.xlabel("Bins")
plt.ylabel("Pixel Information")
for (color_Channel, color_com) in zip(color_Channel, color_com):
    histogram = cv2.calcHist([color_Channel], [0],None, [256], [0, 256])
    plt.plot(histogram, color = color_com)
plt.xlim([0, 256])
plt.show()
```





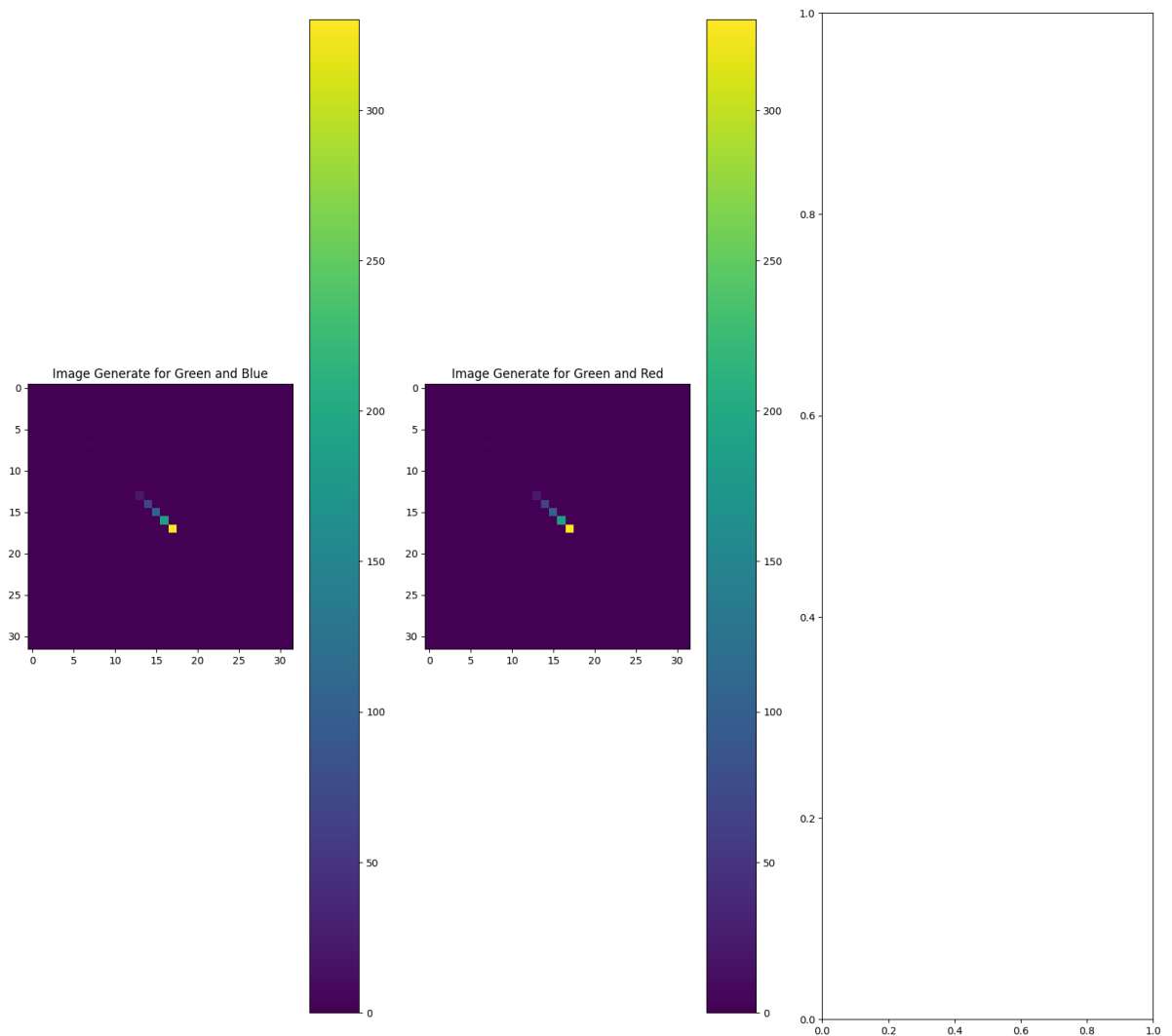
```
In [464... color = ["B", "G", "R"]
fig = plt.figure(figsize=(20, 18))
ax = fig.add_subplot(131)
hist = cv2.calcHist([color_Channel[1], color_Channel[0]], [0, 1], None, [32,
32], [0, 256, 0, 256])
p = ax.imshow(hist, interpolation = "nearest")
```



```

ax.set_title("Image Generate for Green and Blue")
plt.colorbar(p)
ax = fig.add_subplot(132)
hist = cv2.calcHist([color_Channel[1], color_Channel[2]], [0, 1], None, [32,
32], [0, 256, 0, 256])
p = ax.imshow(hist, interpolation = "nearest")
ax.set_title("Image Generate for Green and Red")
plt.colorbar(p)
ax = fig.add_subplot(133)
hist = cv2.calcHist([color_Channel[0], color_Channel[2]], [0, 1], None, [32,
32], [0, 256, 0, 256])

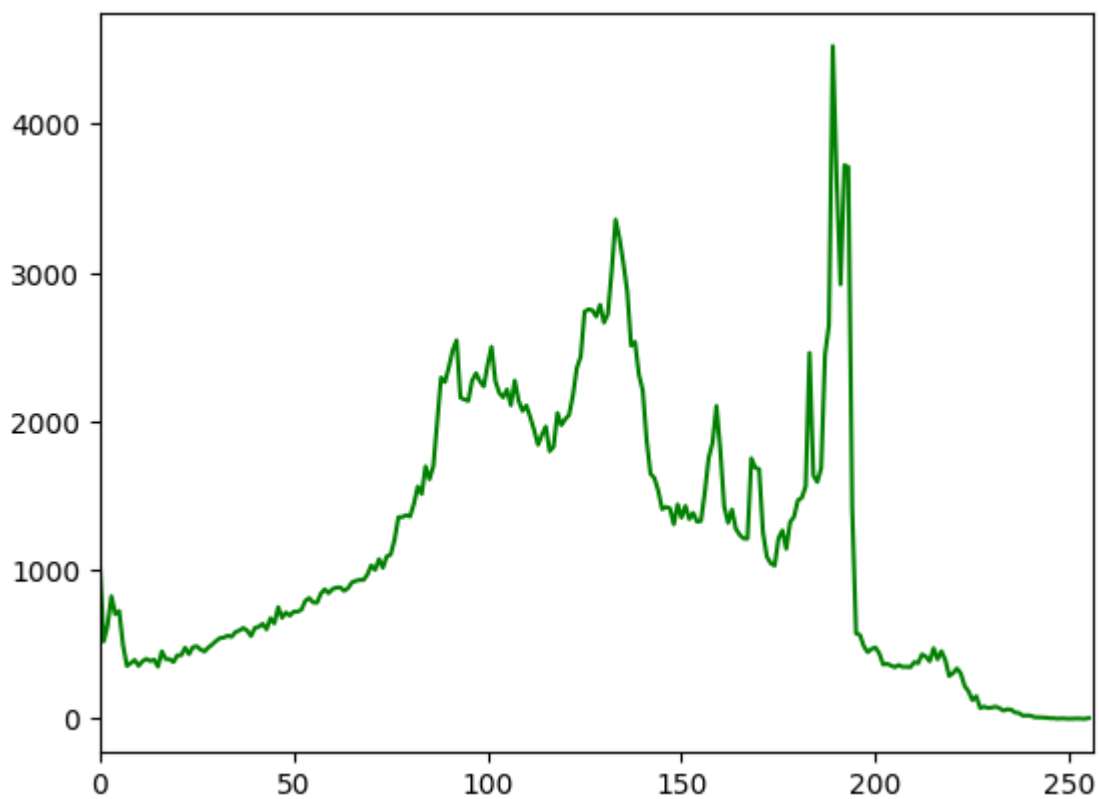
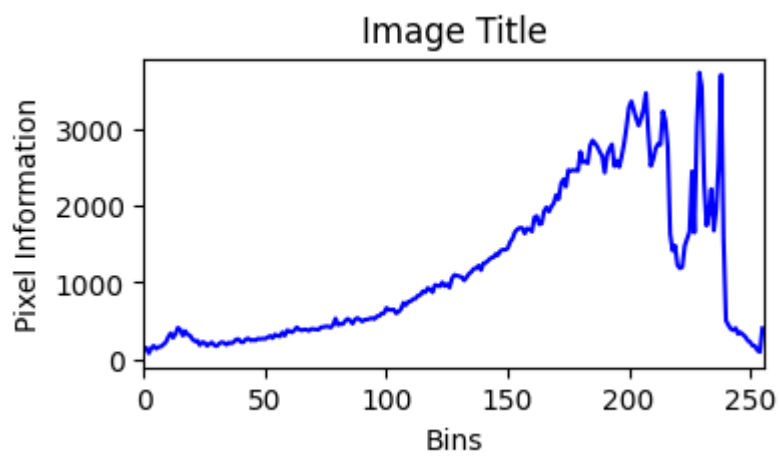
```

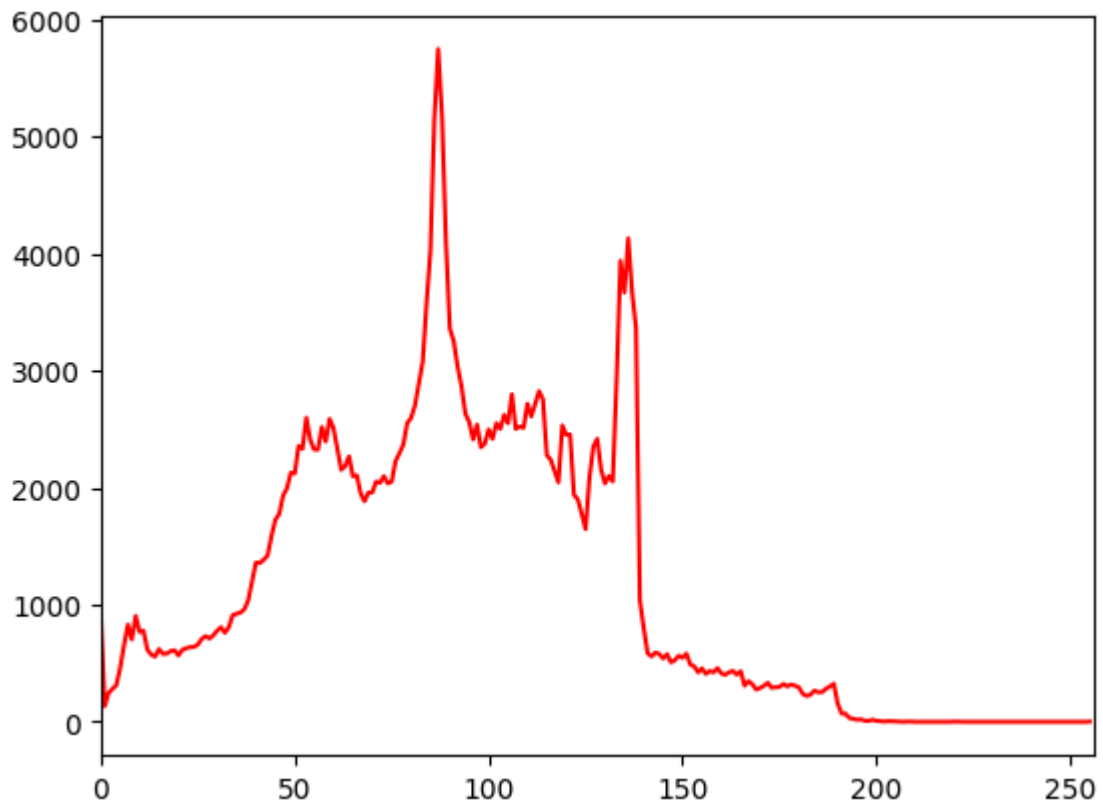


```

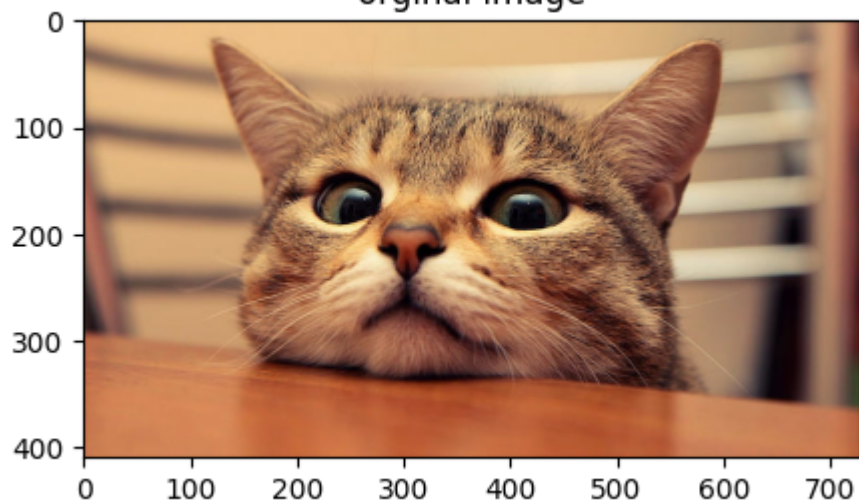
In [465... image = cv2.imread("Cute.jpg")
imageHistogram(image, "Image Title")
catImageshow("original image", image)

```





original image



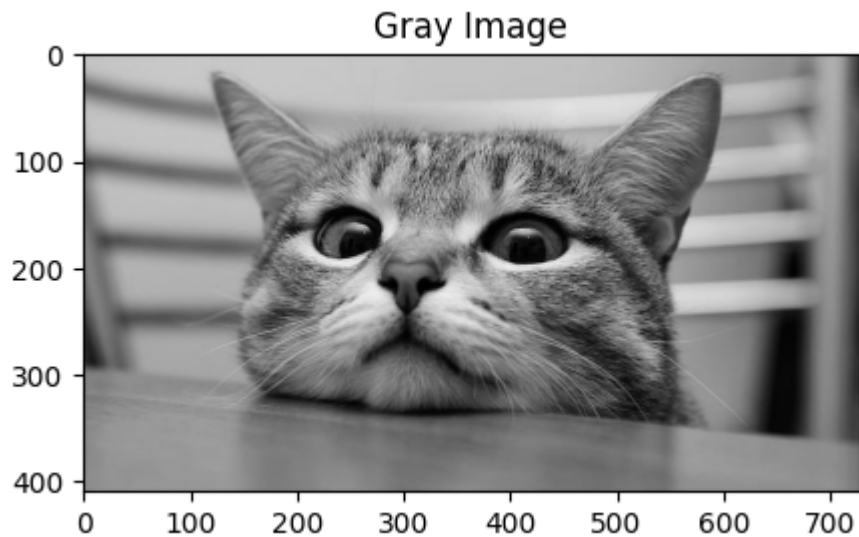
Gray Scale

```
In [466...] gray = cv2.cvtColor(img,cv2.COLOR_RGB2GRAY)
```

```
In [467...] gray.shape
```

```
Out[467]: (410, 728)
```

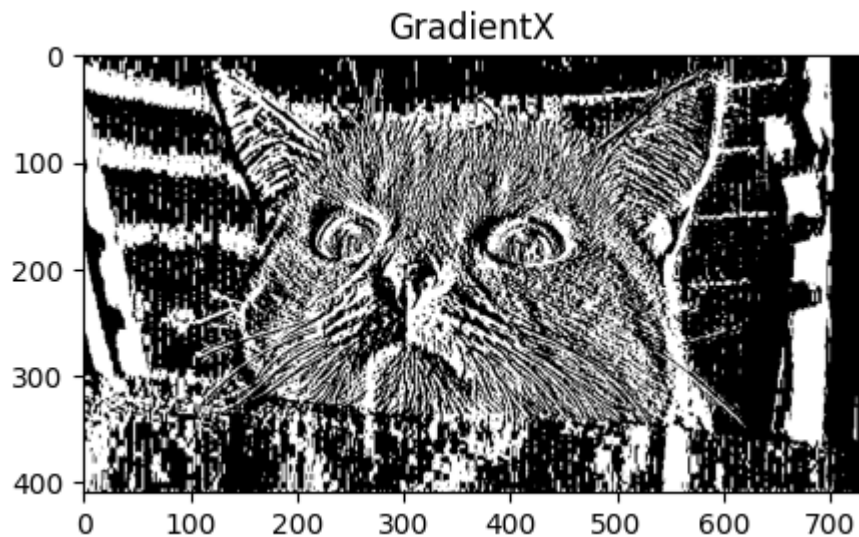
```
In [468...] catImageShow("Gray Image",gray)
```



```
In [469...] gradientX = cv2.Sobel(gray,ddepth = cv2.CV_32F,dx=1,dy=0,ksize = 3)
            gradientY = cv2.Sobel(gray,ddepth = cv2.CV_32F,dx=0,dy=1,ksize=3)
```

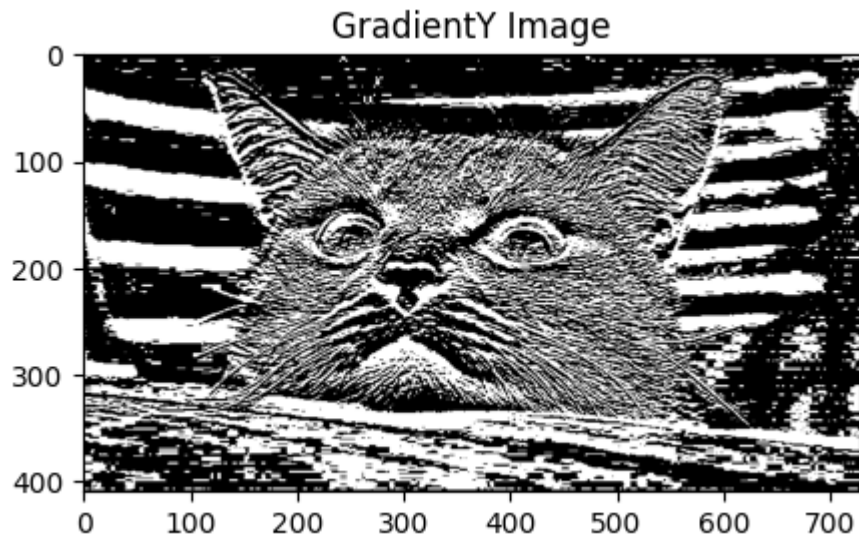
```
In [470...] catImageshow("GradientX",gradientX)
```

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



```
In [471...] catImageshow("GradientY Image",gradientY)
```

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



Masking

```
In [472...] img.shape
```

```
Out[472]: (410, 728, 3)
```

```
In [473...] img.shape[:2]
```

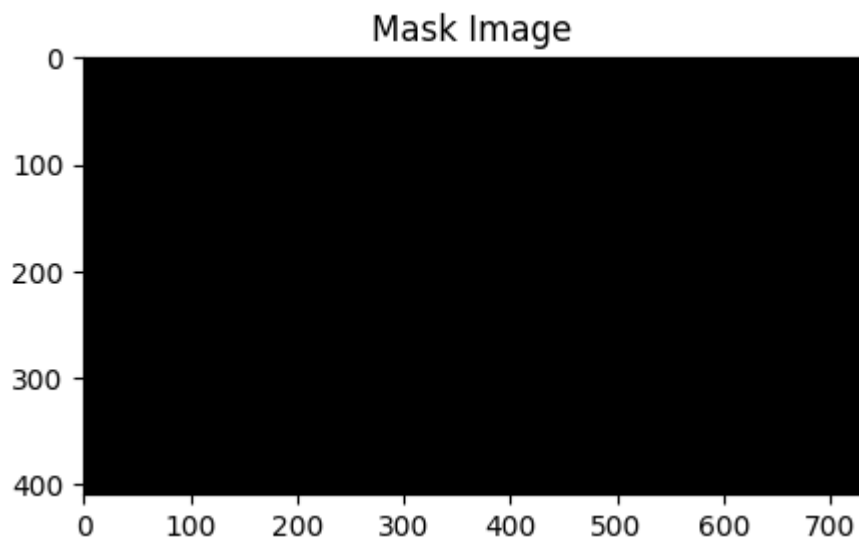
```
Out[473]: (410, 728)
```

```
In [474...] mask = np.zeros(img.shape[:2],dtype="uint8")
```

```
In [475...] mask
```

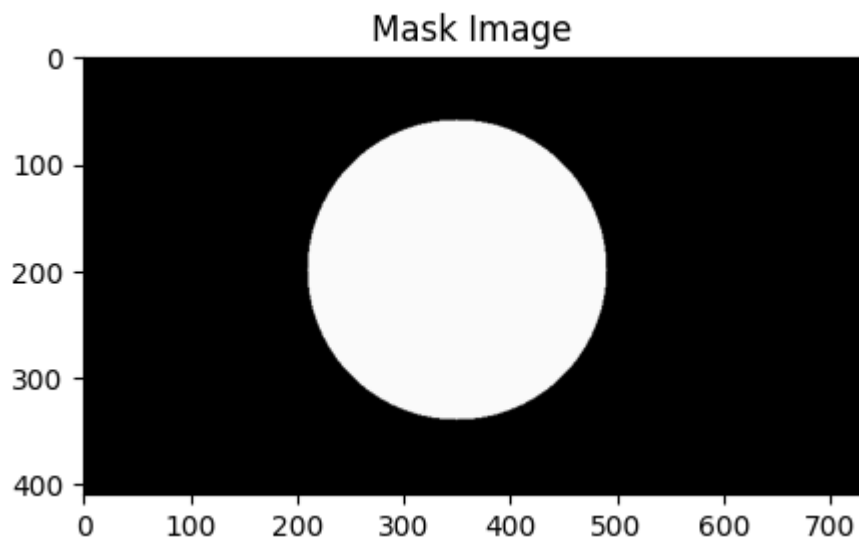
```
Out[475]: array([[0, 0, 0, ..., 0, 0, 0],
                [0, 0, 0, ..., 0, 0, 0],
                [0, 0, 0, ..., 0, 0, 0],
                ...,
                [0, 0, 0, ..., 0, 0, 0],
                [0, 0, 0, ..., 0, 0, 0],
                [0, 0, 0, ..., 0, 0, 0]], dtype=uint8)
```

```
In [476...] cv2.rectangle(mask,(100, 500), (150, 100), 0)
catImageShow("Mask Image",mask)
```



Bit_Mask

```
In [477... cv2.circle(mask, (350, 200), 140, 255, -1)
bit_mask = cv2.bitwise_and(img, img, mask = mask)
catImageshow("Mask Image", mask)
catImageshow("Bit_mask Image", bit_mask)
```



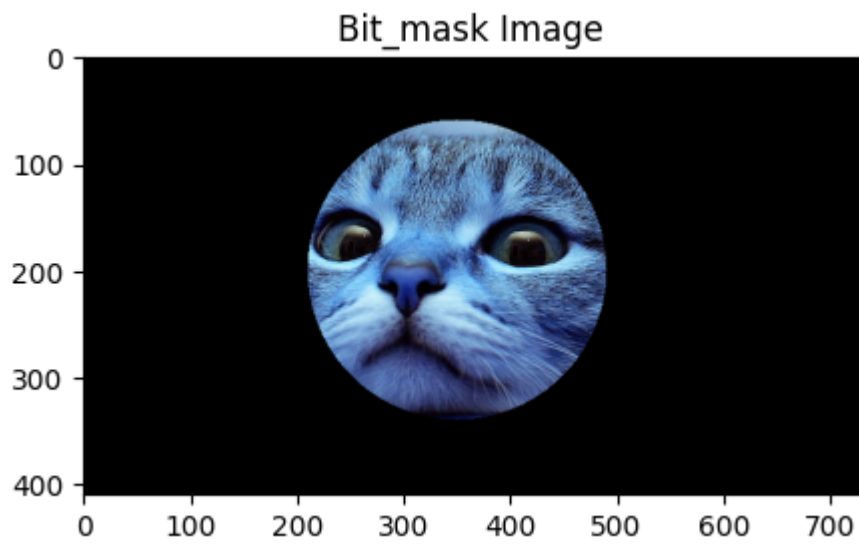


Image Scaling

In [478... `img.shape`

Out[478]: (410, 728, 3)

In [479... `img/255`

```

Out[479]: array([[0.91764706, 0.74117647, 0.5254902 ],
                 [0.91764706, 0.74117647, 0.5254902 ],
                 [0.91764706, 0.74117647, 0.5254902 ],
                 ...,
                 [0.38431373, 0.20392157, 0.21568627],
                 [0.34117647, 0.17647059, 0.18431373],
                 [0.31372549, 0.15294118, 0.16862745]],

                [[0.91764706, 0.74117647, 0.5254902 ],
                 [0.91764706, 0.74117647, 0.5254902 ],
                 [0.91764706, 0.74117647, 0.5254902 ],
                 ...,
                 [0.38431373, 0.20784314, 0.21960784],
                 [0.34117647, 0.17647059, 0.18431373],
                 [0.31372549, 0.15294118, 0.16862745]],

                [[0.91764706, 0.74117647, 0.5254902 ],
                 [0.91764706, 0.74117647, 0.5254902 ],
                 [0.91764706, 0.74117647, 0.5254902 ],
                 ...,
                 [0.38431373, 0.21176471, 0.21568627],
                 [0.34509804, 0.18039216, 0.18431373],
                 [0.31372549, 0.15686275, 0.16078431]],

                ...,

                [[0.78039216, 0.46666667, 0.32156863],
                 [0.78039216, 0.46666667, 0.32156863],
                 [0.78039216, 0.46666667, 0.32156863],
                 ...,
                 [0.7254902 , 0.4        , 0.22745098],
                 [0.7254902 , 0.4        , 0.22745098],
                 [0.72156863, 0.39607843, 0.22352941]],

                [[0.77647059, 0.46666667, 0.31372549],
                 [0.77647059, 0.46666667, 0.31372549],
                 [0.78431373, 0.4745098 , 0.31372549],
                 ...,
                 [0.72941176, 0.39607843, 0.23529412],
                 [0.72941176, 0.39607843, 0.23529412],
                 [0.7254902 , 0.39215686, 0.23137255]],

                [[0.77647059, 0.46666667, 0.31372549],
                 [0.77647059, 0.46666667, 0.31372549],
                 [0.78431373, 0.4745098 , 0.31372549],
                 ...,
                 [0.72941176, 0.39607843, 0.23529412],
                 [0.72941176, 0.39607843, 0.23529412],
                 [0.7254902 , 0.39215686, 0.23137255]]])

```

```
In [480]: img.shape
```

```
Out[480]: (410, 728, 3)
```


Resize Image

```
In [481... customsizeH = 120/img.shape[0]
```

```
In [482... customsizeW = 120/img.shape[1]
```

```
In [483... customsizeW
```

```
Out[483]: 0.16483516483516483
```

```
In [484... customsizeH
```

```
Out[484]: 0.2926829268292683
```

```
In [485... imgDimension = (120,int(img.shape[0]*customsizeW))
```

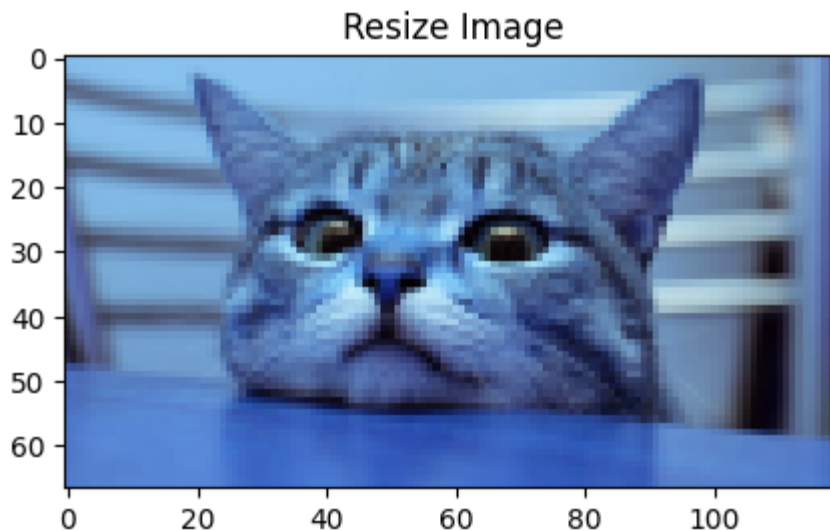
```
In [486... imgDimension
```

```
Out[486]: (120, 67)
```

```
In [487... img.shape
```

```
Out[487]: (410, 728, 3)
```

```
In [488... resizeImage = cv2.resize(img,imgDimension,interpolation = cv2.INTER_AREA)  
catImageshow("Resize Image",resizeImage)
```



```
In [489... resizeImage.shape
```

```
Out[489]: (67, 120, 3)
```

Rotate Image

```
In [490...] (imageH,imageW) = img.shape[:2]
```

```
In [491...] imageH
```

```
Out[491]: 410
```

```
In [492...] imageW
```

```
Out[492]: 728
```

```
In [493...] (centerX,centerY) = (imageH//2,imageW//2)
```

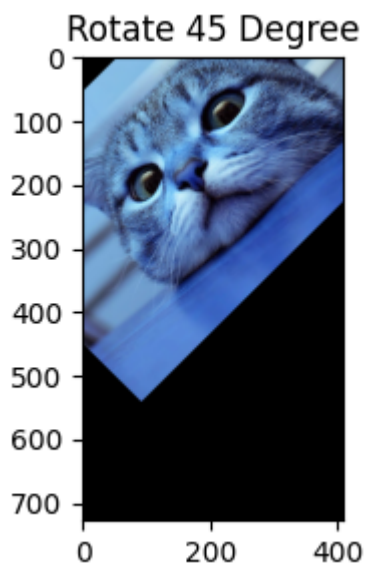
```
In [494...] centerX
```

```
Out[494]: 205
```

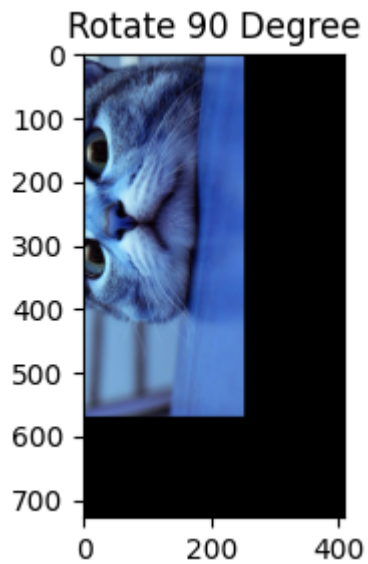
```
In [495...] centerY
```

```
Out[495]: 364
```

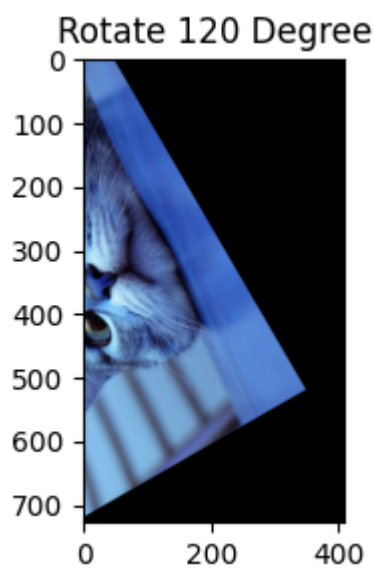
```
In [496...] imageRorate = cv2.getRotationMatrix2D((centerX,centerY),45,1.0)  
rotateNow = cv2.warpAffine(img,imageRorate,(imageH,imageW))  
catImageshow("Rotate 45 Degree",rotateNow)
```



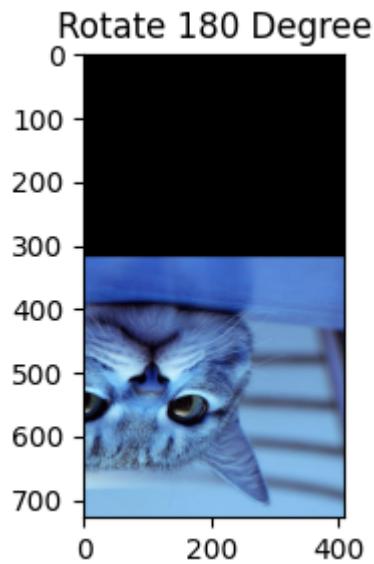
```
In [497...] imageRorate = cv2.getRotationMatrix2D((centerX,centerY),90,1.0)  
rotateNow = cv2.warpAffine(img,imageRorate,(imageH,imageW))  
catImageshow("Rotate 90 Degree",rotateNow)
```



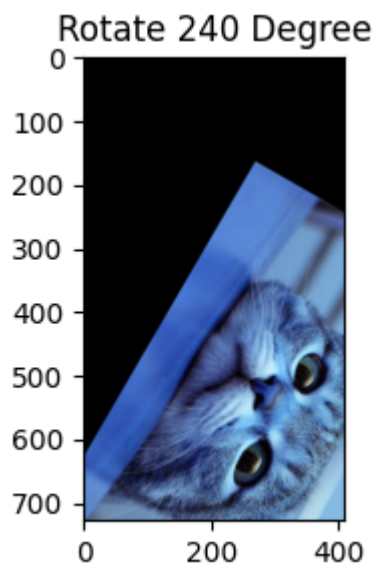
```
In [498... imageRorate = cv2.getRotationMatrix2D((centerX,centerY),120,1.0)
rotateNow = cv2.warpAffine(img,imageRorate,(imageH,imageW))
catImageshow("Rotate 120 Degree",rotateNow)
```



```
In [499... imageRorate = cv2.getRotationMatrix2D((centerX,centerY),180,1.0)
rotateNow = cv2.warpAffine(img,imageRorate,(imageH,imageW))
catImageshow("Rotate 180 Degree",rotateNow)
```



```
In [500... imageRorate = cv2.getRotationMatrix2D((centerX,centerY),240,1.0)
rotateNow = cv2.warpAffine(img,imageRorate,(imageH,imageW))
catImageshow("Rotate 240 Degree",rotateNow)
```



```
In [501... imageRorate = cv2.getRotationMatrix2D((centerX,centerY),360,1.0)
rotateNow = cv2.warpAffine(img,imageRorate,(imageH,imageW))
catImageshow("Rotate again 360 Degree",rotateNow)
```

Rotate again 360 Degree

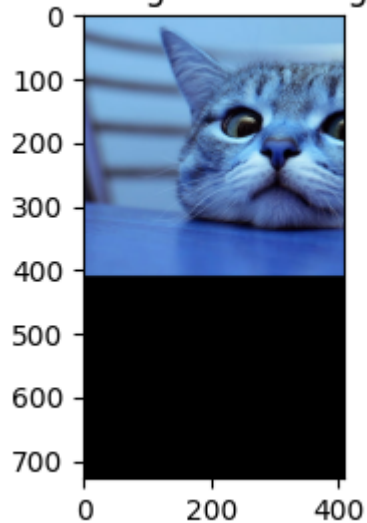


Image blurring

```
In [502... image = cv2.imread('cute.jpg')

cv2.imshow('Original Image', image)
cv2.waitKey(0)

# Gaussian Blur
Gaussian = cv2.GaussianBlur(image, (7, 7), 0)
cv2.imshow('Gaussian Blurring', Gaussian)
cv2.waitKey(0)

# Median Blur
median = cv2.medianBlur(image, 5)
cv2.imshow('Median Blurring', median)
cv2.waitKey(0)

# Bilateral Blur
bilateral = cv2.bilateralFilter(image, 9, 75, 75)
cv2.imshow('Bilateral Blurring', bilateral)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Scale

```
In [503... maxScaleUp = 100
scaleFactor = 1
#
windowName = "Resize Image"
trackbarValue = "Scale"

# read the image
```

```

image = cv2.imread("cute.jpg")

# Create a window to display results and set the flag to Autosize
cv2.namedWindow(windowName, cv2.WINDOW_AUTOSIZE)

# Callback functions
def scaleImage(*args):
    # Get the scale factor from the trackbar
    scaleFactor = 1+ args[0]/100.0
    # Resize the image
    scaledImage = cv2.resize(image, None, fx=scaleFactor, fy = scaleFactor, interpo
    cv2.imshow(windowName, scaledImage)

# Create trackbar and associate a callback function
cv2.createTrackbar(trackbarValue, windowName, scaleFactor, maxScaleUp, scaleImage)
top_left_corner = None
bottom_right_corner = None

# function which will be called on mouse input
def drawRectangle(action, x, y, flags, userdata):
    # Referencing global variables
    global top_left_corner, bottom_right_corner
    # Mark the top left corner when left mouse button is pressed
    if action == cv2.EVENT_LBUTTONDOWN:
        top_left_corner = (x, y)
        bottom_right_corner = None
    # When left mouse button is released, mark bottom right corner
    elif action == cv2.EVENT_LBUTTONUP:
        bottom_right_corner = (x, y)
    # Draw the rectangle if both corners are initialized
    if top_left_corner and bottom_right_corner:
        cv2.rectangle(image, top_left_corner, bottom_right_corner, (0, 255, 0), 2, 8)
        cv2.imshow("Window", image)

# Read Images
image = cv2.imread("Cute.jpg")
# Make a temporary image, will be useful to clear the drawing
temp = image.copy()
# Create a named window
cv2.namedWindow("Window")
# highgui function called when mouse events occur
cv2.setMouseCallback("Window", drawRectangle)

k = 0
# Close the window when key q is pressed
while k != 113:
    # Display the image
    cv2.imshow("Window", image)
    k = cv2.waitKey(0)
    # If c is pressed, clear the window, using the dummy image
    if (k == 99):
        image = temp.copy()
        top_left_corner = None
        bottom_right_corner = None
        cv2.imshow("Window", image)

```

```

# Display the image
cv2.imshow(windowName, image)
c = cv2.waitKey(0)
cv2.destroyAllWindows()

```

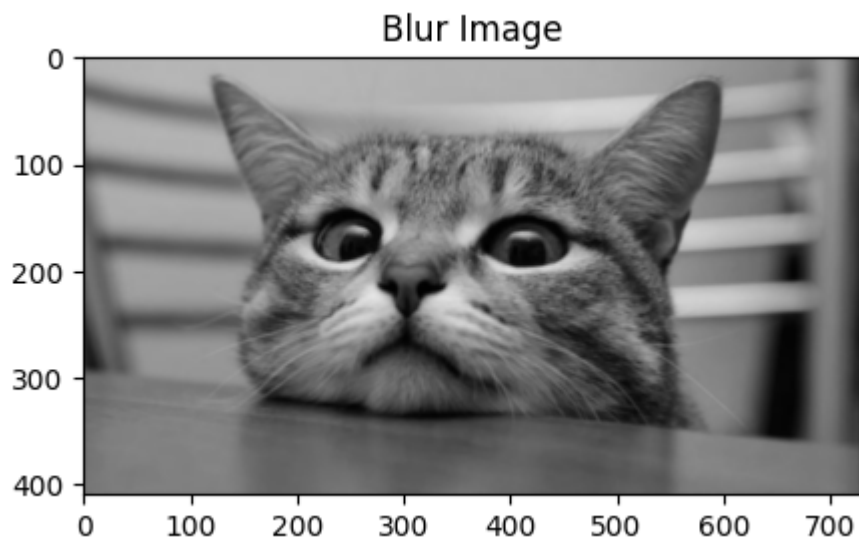
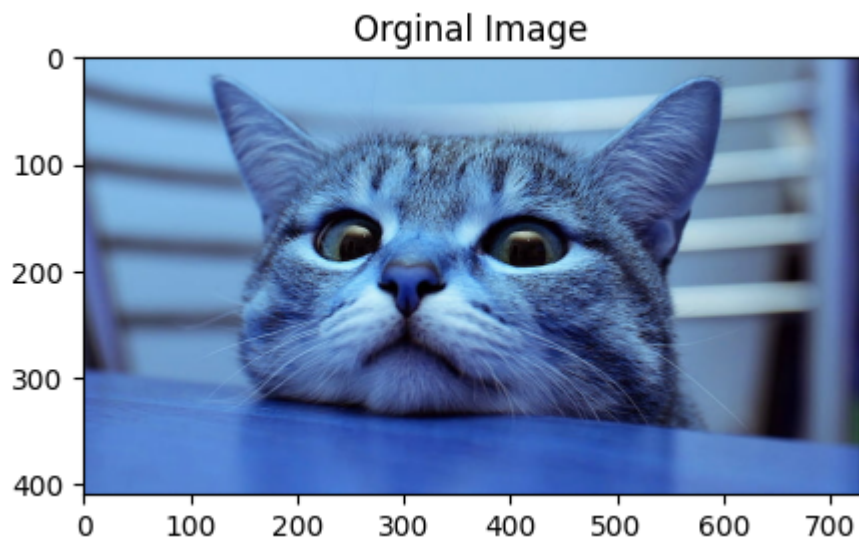
```

In [504... ## Initialize points as None
#top_left_corner = None
#bottom_right_corner = None
#
## function which will be called on mouse input
#def drawRectangle(action, x, y, flags, userdata):
# # Referencing global variables
# global top_left_corner, bottom_right_corner
# # Mark the top left corner when left mouse button is pressed
# if action == cv2.EVENT_LBUTTONDOWN:
#     top_left_corner = (x, y)
#     bottom_right_corner = None
# # When left mouse button is released, mark bottom right corner
# elif action == cv2.EVENT_LBUTTONUP:
#     bottom_right_corner = (x, y)
# # Draw the rectangle if both corners are initialized
# if top_left_corner and bottom_right_corner:
#     cv2.rectangle(image, top_left_corner, bottom_right_corner, (0, 255, 0), 2, 8)
#     cv2.imshow("Window", image)
#
## Read Images
#image = cv2.imread("Cute.jpg")
## Make a temporary image, will be useful to clear the drawing
#temp = image.copy()
## Create a named window
#cv2.namedWindow("Window")
## highgui function called when mouse events occur
#cv2.setMouseCallback("Window", drawRectangle)
#
#k = 0
## Close the window when key q is pressed
#while k != 113:
# # Display the image
# cv2.imshow("Window", image)
# k = cv2.waitKey(0)
# # If c is pressed, clear the window, using the dummy image
# if (k == 99):
#     image = temp.copy()
#     top_left_corner = None
#     bottom_right_corner = None
#     cv2.imshow("Window", image)
#
#cv2.destroyAllWindows()
#

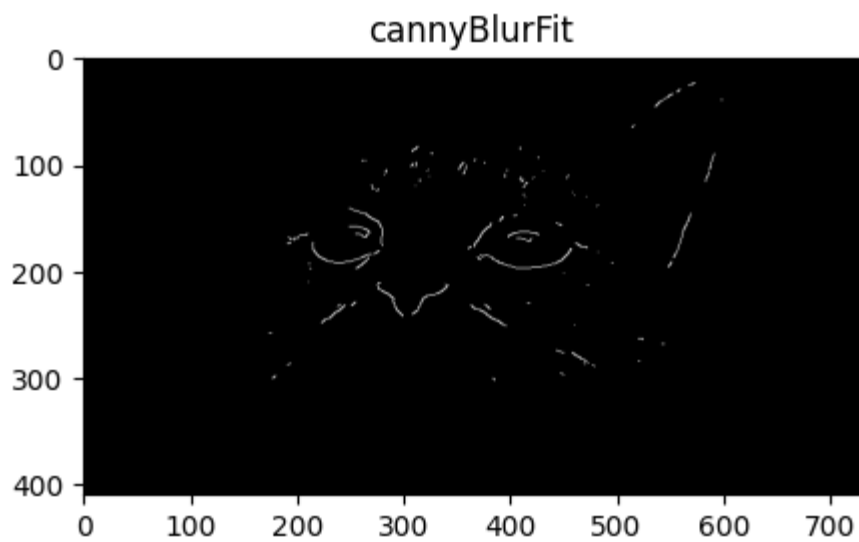
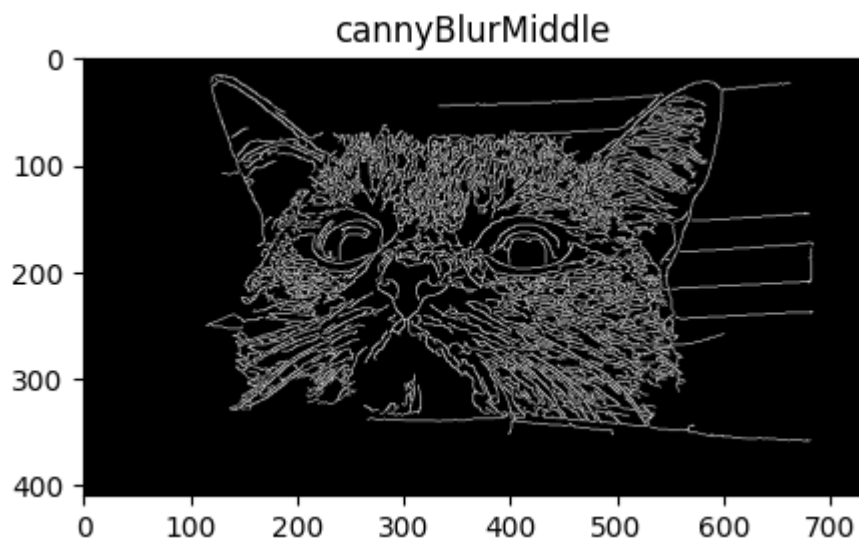
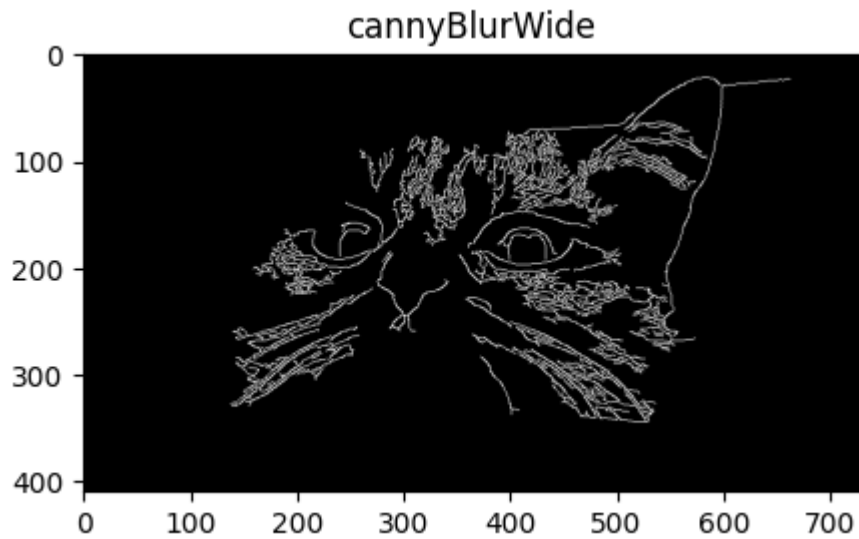
```

Canny Edge Detection

```
In [505... image=cv2.imread('Cute.jpg')
image = cv2.imread("CuteCat.jpg")
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
blurImage = cv2.GaussianBlur(gray, (5, 5), 0)
catImageshow("Original Image", img)
catImageshow("Blur Image", blurImage)
```



```
In [506... cannyBlurWide = cv2.Canny(blurImage, 20, 250)
cannyBlurMiddle = cv2.Canny(blurImage, 15, 120)
cannyBlurFit = cv2.Canny(blurImage, 240, 250)
catImageshow("cannyBlurWide",cannyBlurWide)
catImageshow("cannyBlurMiddle", cannyBlurMiddle)
catImageshow("cannyBlurFit", cannyBlurFit)
```

```
In [507... image=cv2.imread('Cute.jpg')
img_blur = cv2.GaussianBlur(image, (5,5), 0)
edges = cv2.Canny(image=img_blur, threshold1=100, threshold2=200)
k = 0
```

```
# Display Canny Edge Detection Image
cv2.imshow('Canny Edge Detection', edges)

k=cv2.waitKey(0)
cv2.destroyAllWindows
```

Out[507]: <function destroyAllWindows>

Sobel Edge Detection

```
In [508... image=cv2.imread('Cute.jpg')
img_blur = cv2.GaussianBlur(image, (5,5), 0)

sobelx = cv2.Sobel(src=img_blur, ddepth=cv2.CV_64F, dx=1, dy=0, ksize=5) # Sobel Ed
sobely = cv2.Sobel(src=img_blur, ddepth=cv2.CV_64F, dx=0, dy=1, ksize=5) # Sobel Ed
sobelxy = cv2.Sobel(src=img_blur, ddepth=cv2.CV_64F, dx=1, dy=1, ksize=5) # Combine
k=0
d=0
c=0
# Display Sobel Edge Detection Images
cv2.imshow('Sobel X', sobelx)
k=cv2.waitKey(0)

cv2.imshow('Sobel Y', sobely)
d=cv2.waitKey(0)

cv2.imshow('Sobel X Y using Sobel() function', sobelxy)
c=cv2.waitKey(0)
cv2.destroyAllWindows()
```