Created By - Piyush Dwivedi

Connect me - https://www.linkedin.com/in/piyush-dwivedi-1909a5213/

Email Address- dwivedipiyush9754@gmail.com

Library used in project and library features

#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

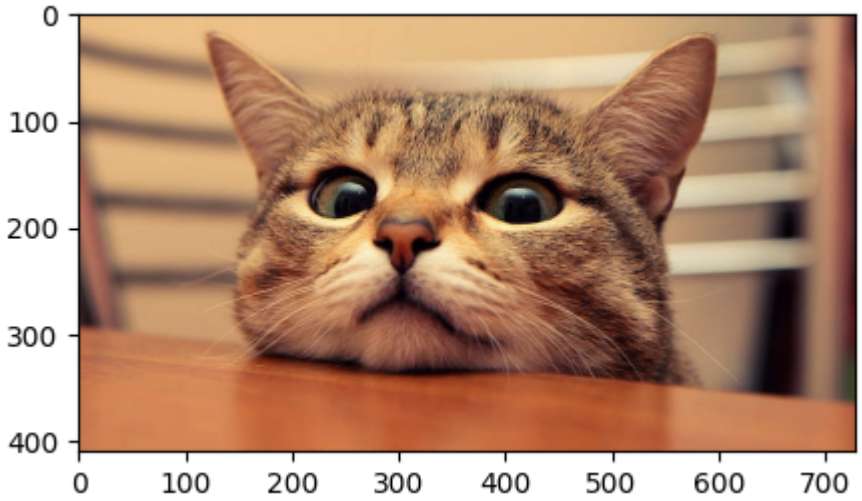
Image processing

#Hog

Histogram of Oriented Gradients

In [3]:
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
import warnings
from skimage.feature import hog
warnings.filterwarnings("ignore")
%matplotlib inline
```

In [4]:
```
plt.figure(figsize=(5,3))
img = plt.imread('cute.jpg')
plt.imshow(img)
plt.grid(False)
plt.show()
```



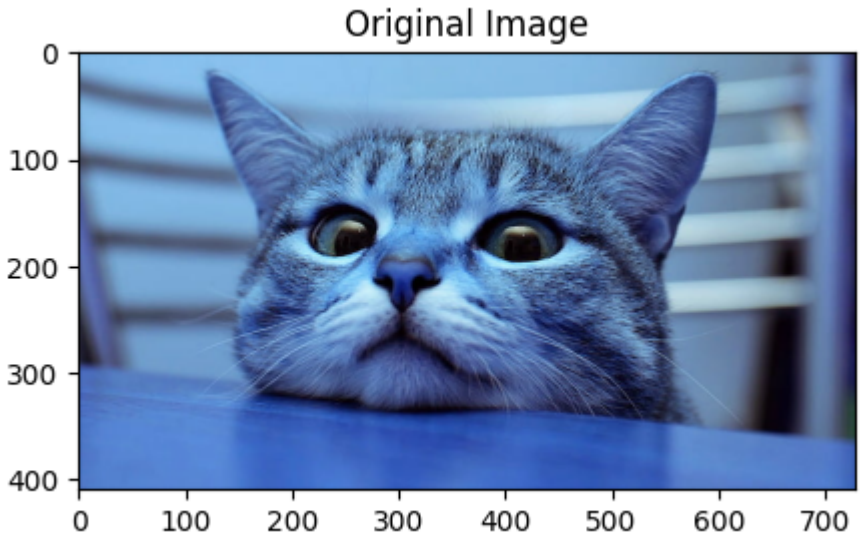
In [5]:
```
len(img.shape)
```

Out[5]:
```
3
```

In [6]:
```
img.shape
```

Out[6]:
```
(410, 728, 3)
```

In [7]:
```
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)
```



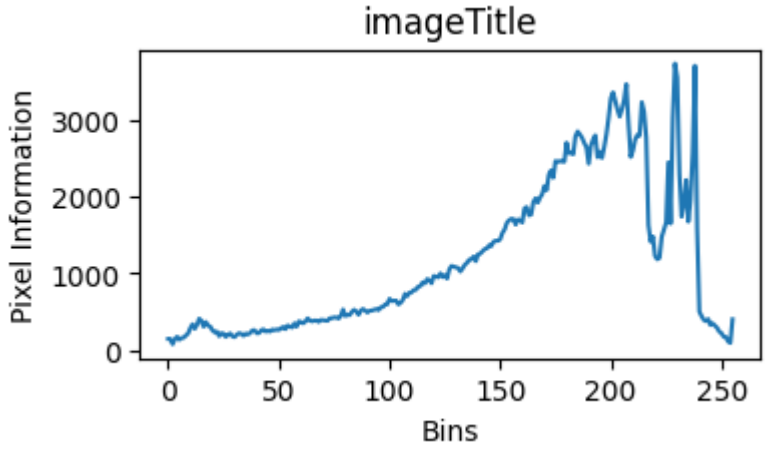
In [8]:
```
cv2.split(img)
```

Out[8]:
```
(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))
```

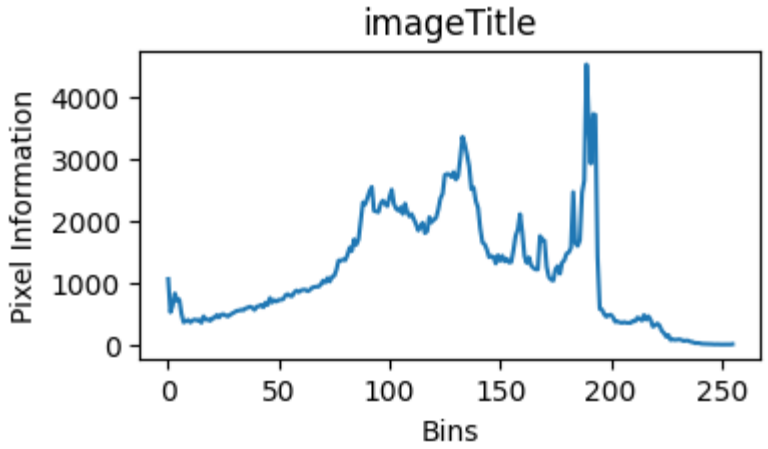
In [9]:
```
def imageHistogram(image, imageTitle, mask = None):
    color_Channel=cv2.split(img)
    color_com = ('b', 'g', 'r')
    plt.figure(figsize = (4, 2))
    plt.title(imageTitle)
    plt.xlabel("Bins")
    plt.ylabel("Pixel Information")
    for (color_Channel, color_com) in zip(color_Channel, 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 [10]:
```
image_argument = {"Image":"cute.jpg"}
```

```
image = cv2.imread(image_argument["Image"])
image= cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
smaplehisto = 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(smaplehisto)
plt.show()
```

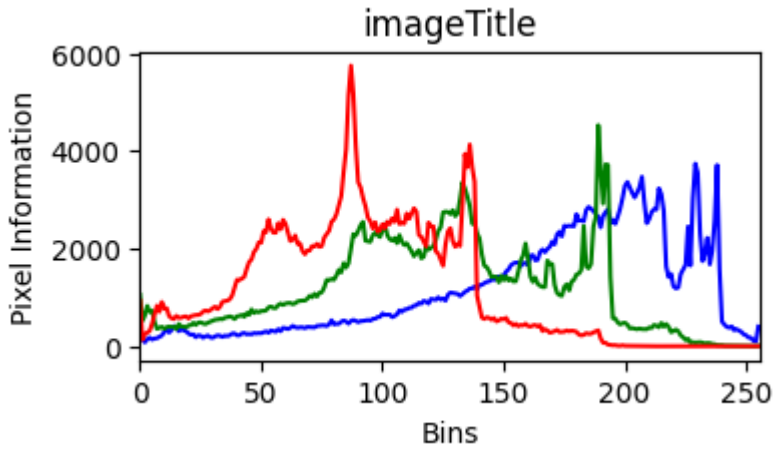


In [11]:
```
image_argument = {"Image":"cute.jpg"}
image = cv2.imread(image_argument["Image"])
image= cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
smaplehisto = cv2.calcHist([image],[1], None, [256], [0, 256])
plt.figure(figsize = (4, 2))
plt.title("imageTitle")
plt.xlabel("Bins")
plt.ylabel("Pixel Information")
plt.plot(smaplehisto)#Line
plt.show()
```



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

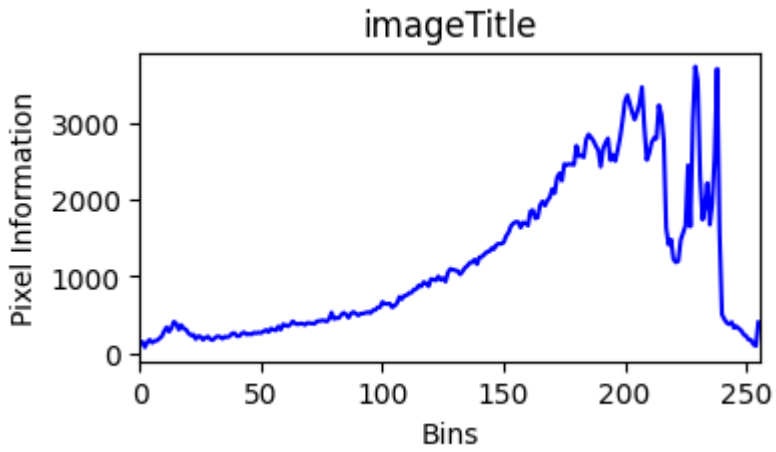


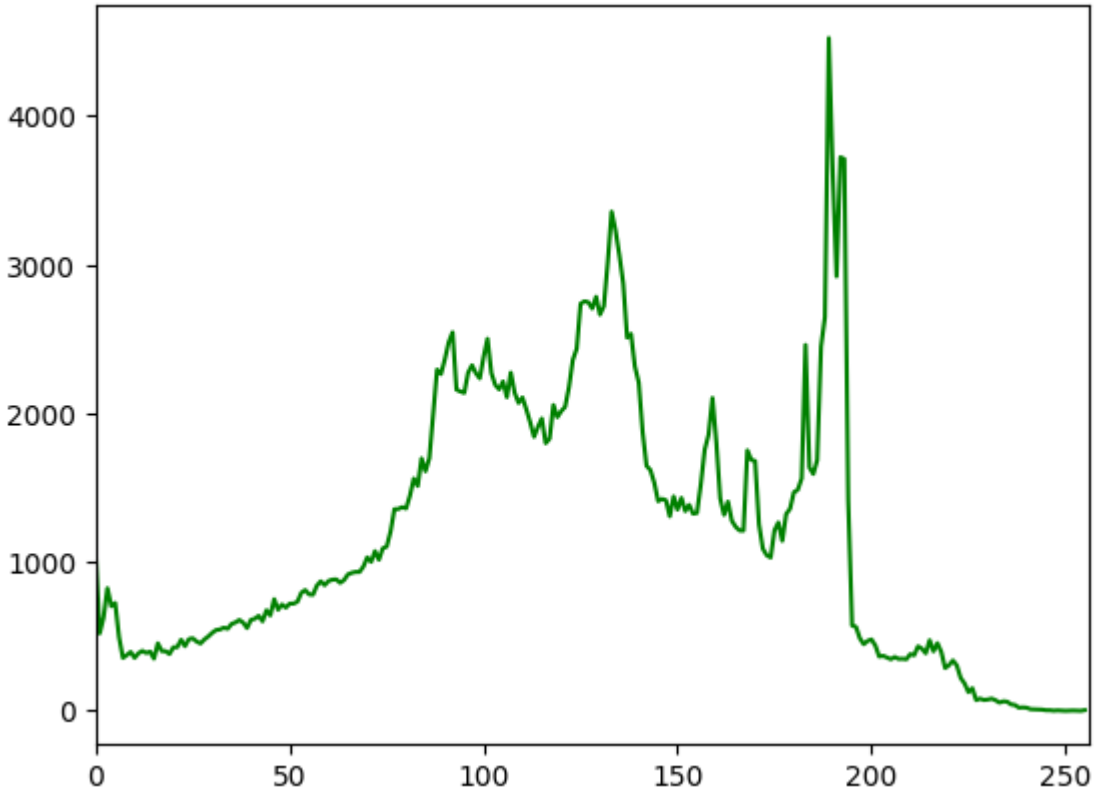
In [13]:
```
cv2.split(image)
```

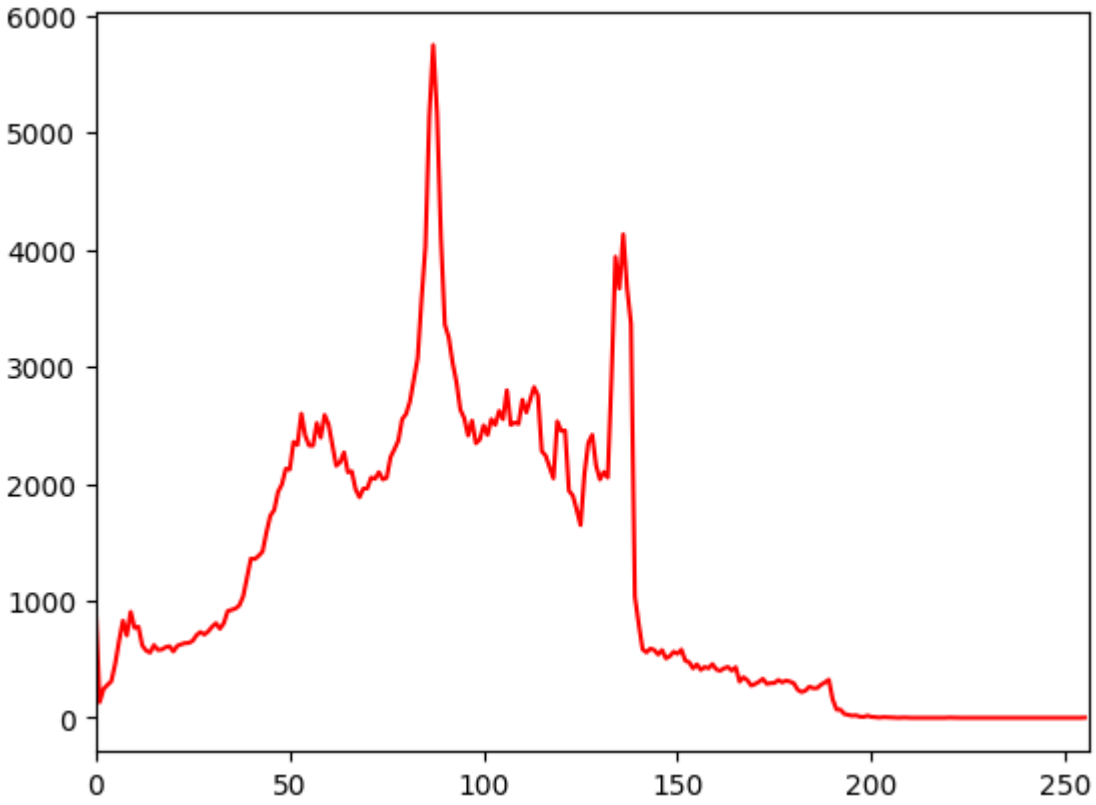
Out[13]:
```
(array([[234, 234, 234, ...,  98,  87,  80],
        [234, 234, 234, ...,  98,  87,  80],
        [234, 234, 234, ...,  98,  88,  80],
        ...,
        [199, 199, 199, ..., 185, 185, 184],
        [198, 198, 200, ..., 186, 186, 185],
        [198, 198, 200, ..., 186, 186, 185]], dtype=uint8),
 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 [14]:
```
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)
```

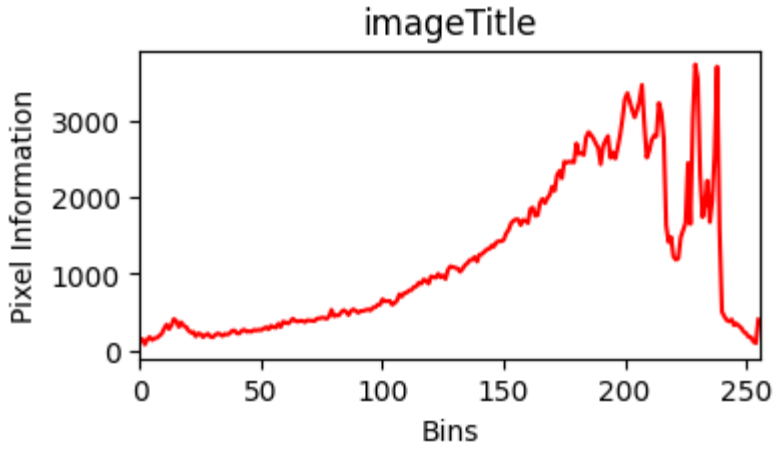
```
    plt.xlim([0, 256])
    plt.show()
```

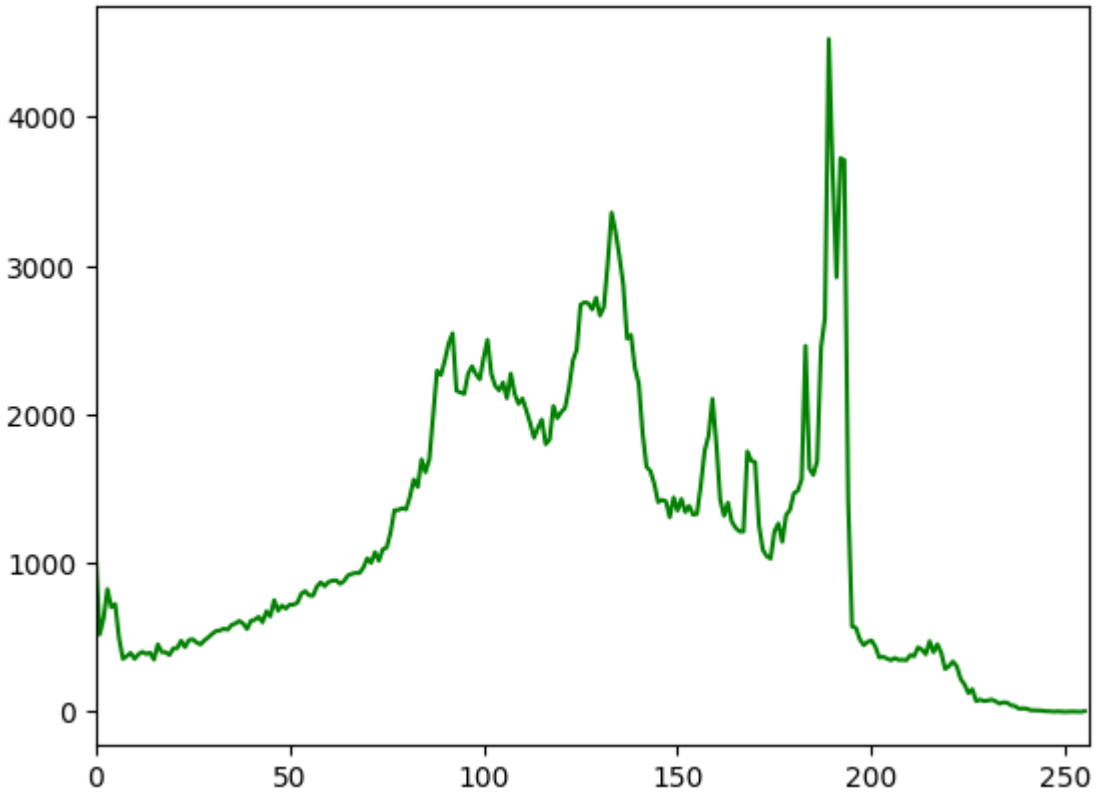


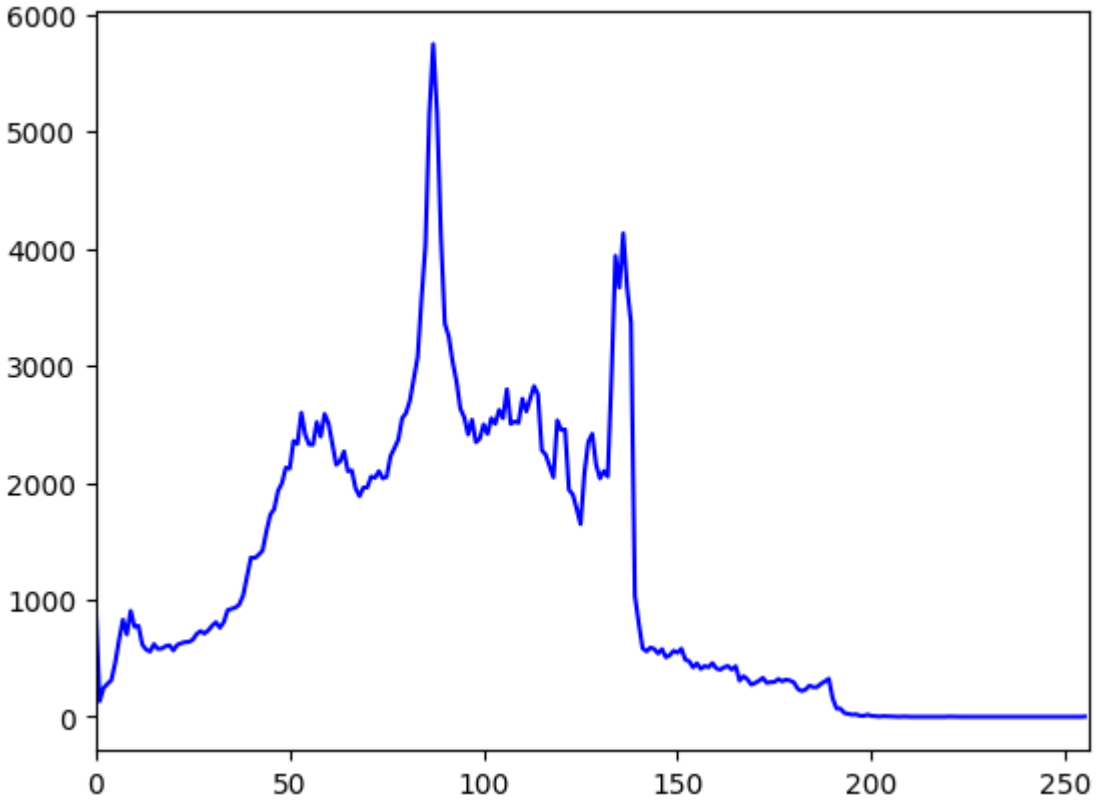




In [15]:
```
color_Channel = cv2.split(image)
color_com = ('r', 'g', 'b')
plt.figure(figsize = (4, 2))
plt.title("imageTitle")
plt.xlabel("Bins")
plt.ylabel("Pixel Information")
for (color_Channel, color_com) in zip(color_Channel, color_com):
    histogram = cv2.calcHist([color_Channel], [0],None, [256], [0, 256])
    plt.plot(histogram, color = color_com)
    plt.xlim([0, 256])
    plt.show()
```

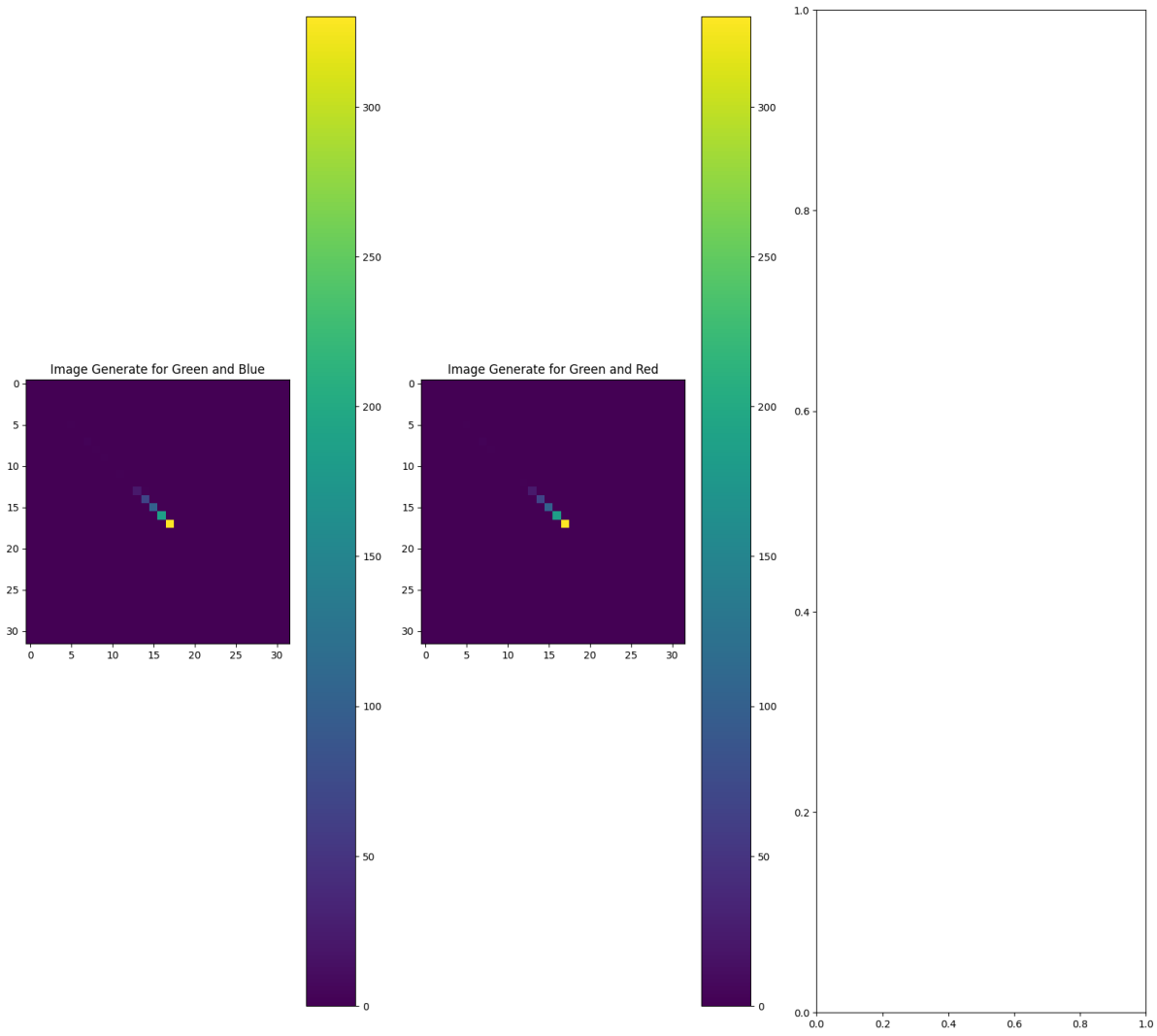






In [16]:
```
color = ["B", "G", "R"]
fig = plt.figure(figsize=(20, 18))
ax = fig.add_subplot(131)
hist = cv2.calcHist([color_Channel[1], color_Channel[0]], [0, 1], None, [32,
32], [0, 256, 0, 256])
p = ax.imshow(hist, interpolation = "nearest")
```

```
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 [17]:

```
image = cv2.imread("cute.jpg")
imageHistogram(image, "Image Title")
catImageshow("orginal image", image)
```

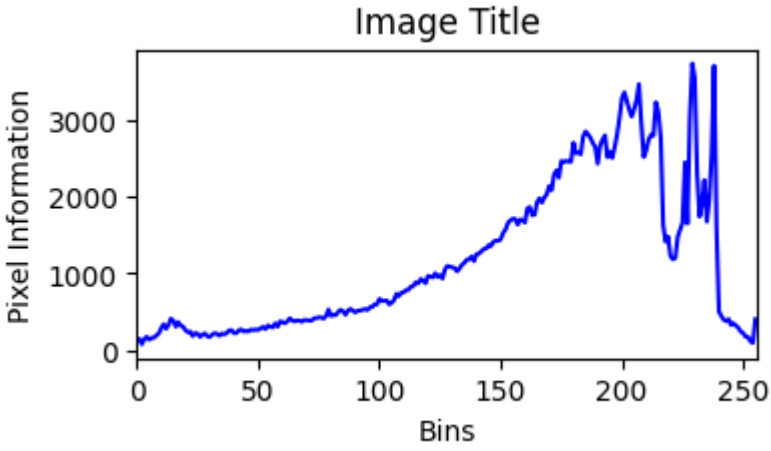
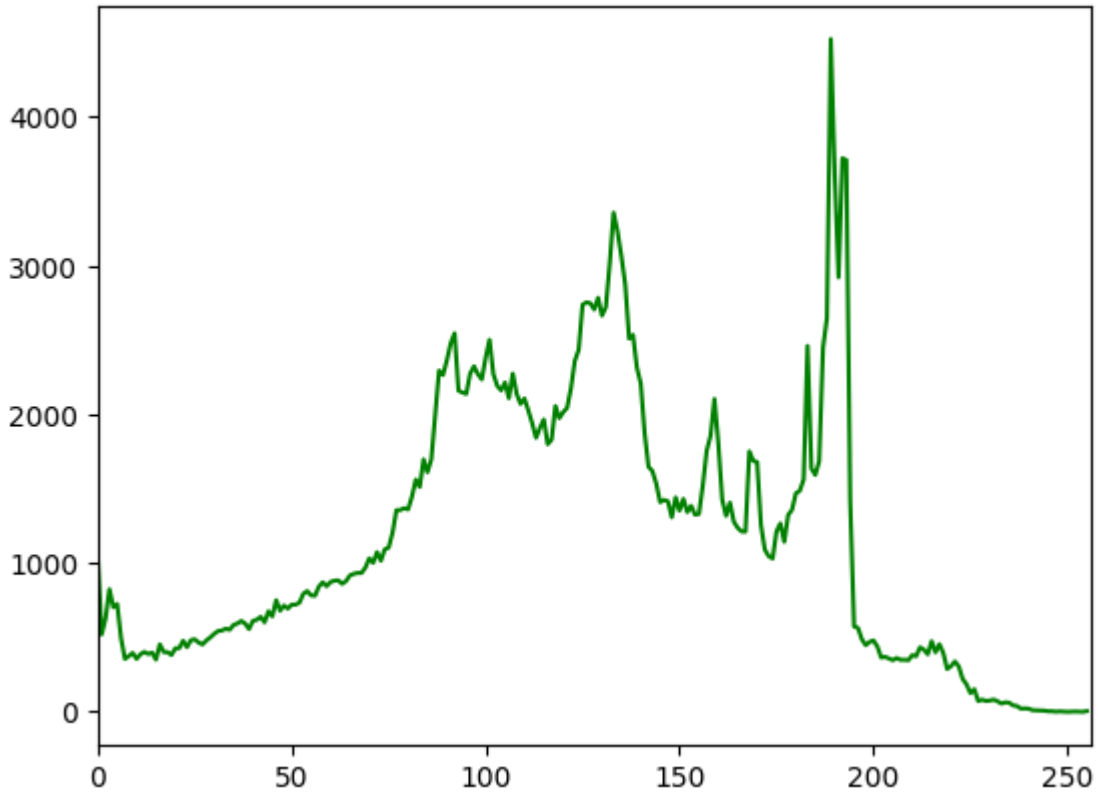
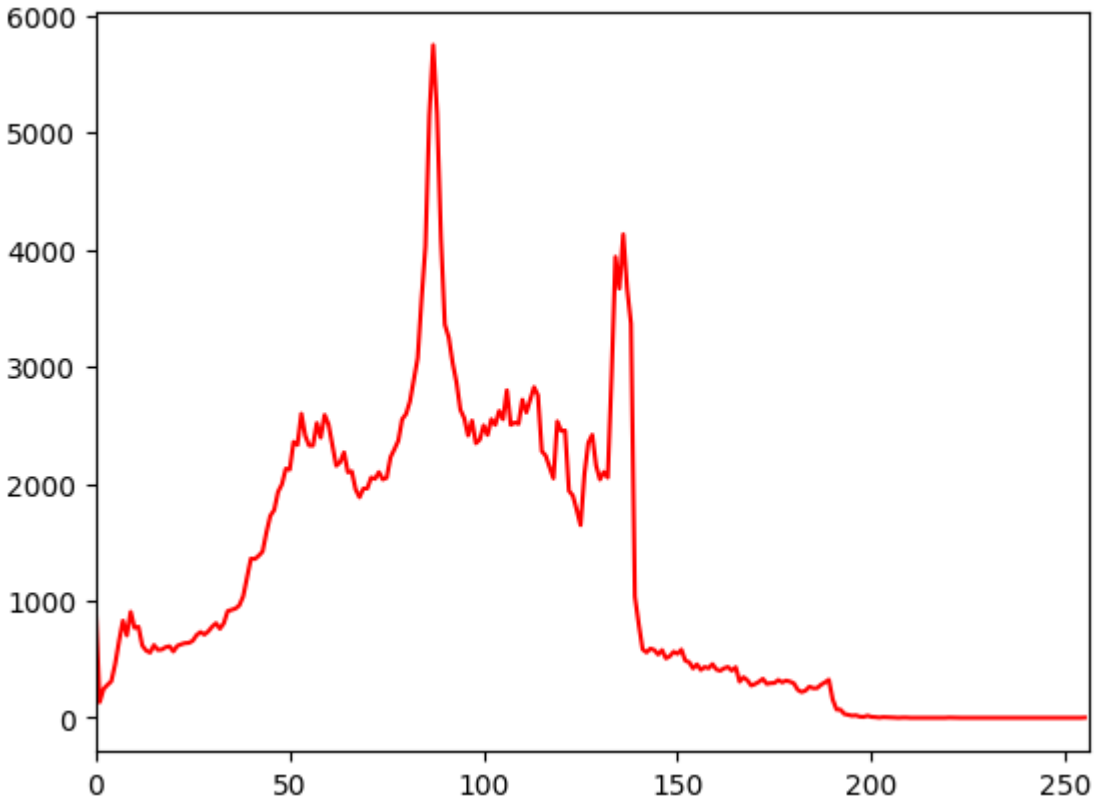
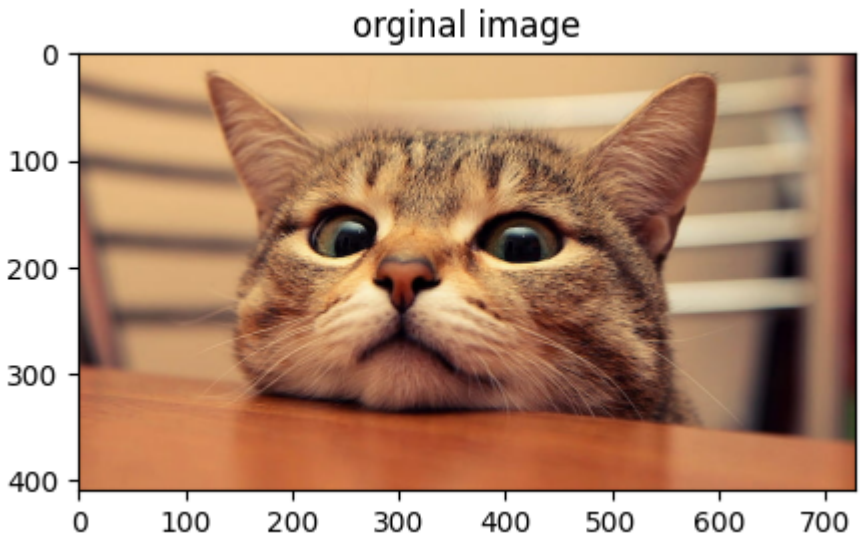


Image Title
Pixel Information
3000
2000
1000
0
0
50
100
150
200
250
Bins


4000
3000
2000
1000
0
0
50
100
150
200
250





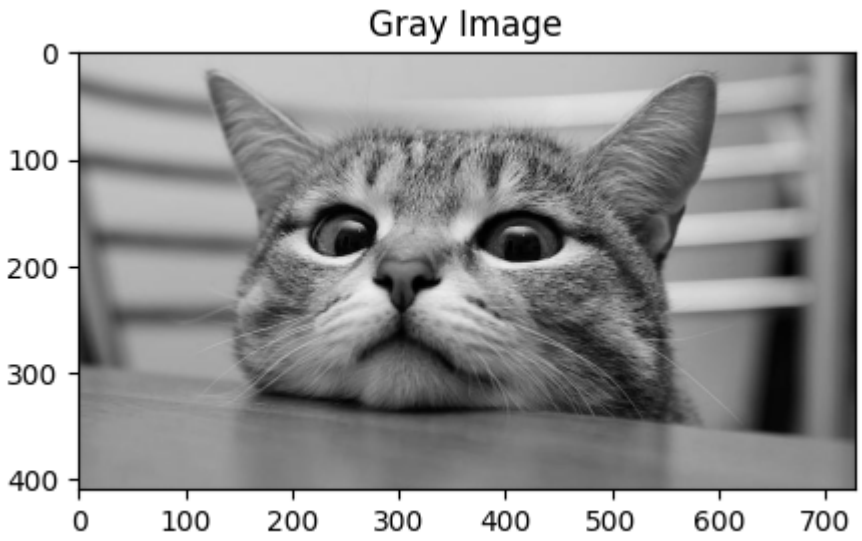
Gray Scale

In [18]:
```
gray = cv2.cvtColor(img,cv2.COLOR_RGB2GRAY)
```

In [19]:
```
gray.shape
```

Out[19]: (410, 728)

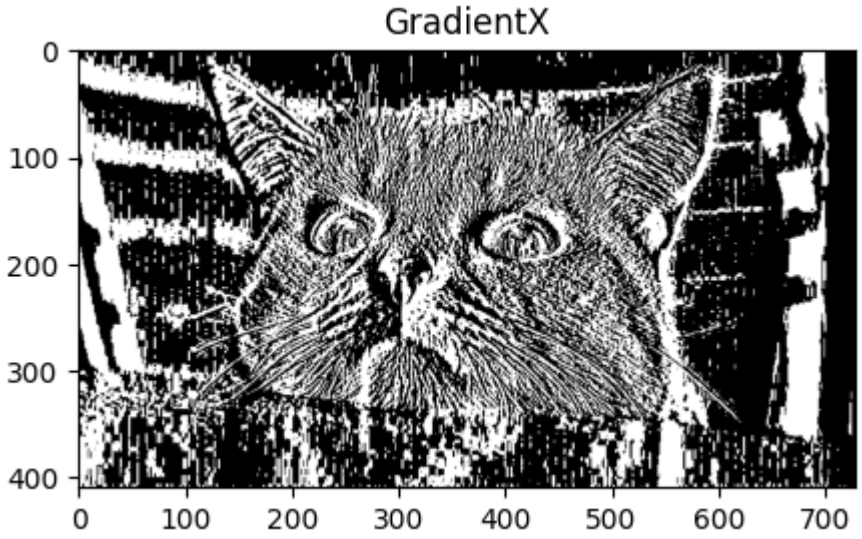
In [20]:
```
catImageshow("Gray Image",gray)
```



In [21]:
```
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)
```

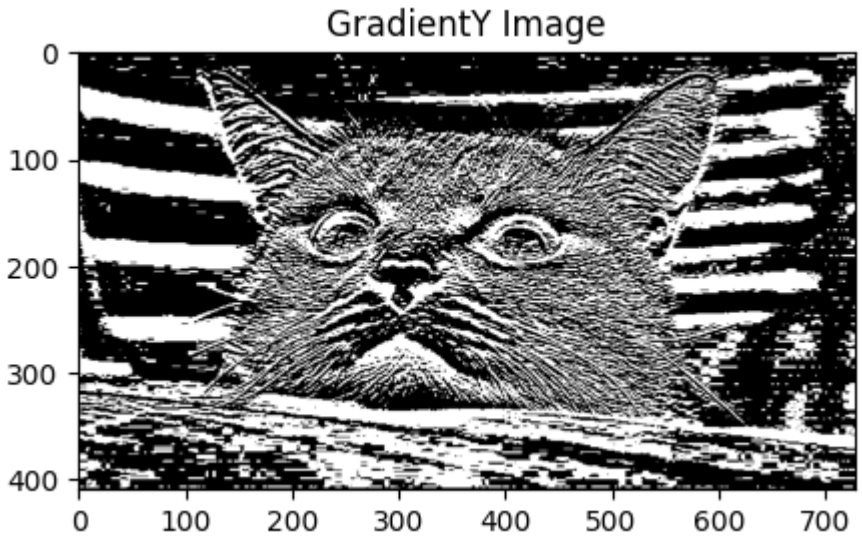
In [22]:
```
catImageshow("GradientX",gradientX)
```

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



In [23]:
```
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 [24]:
```
img.shape
```

Out[24]:
```
(410, 728, 3)
```

In [25]:
```
img.shape[:2]
```

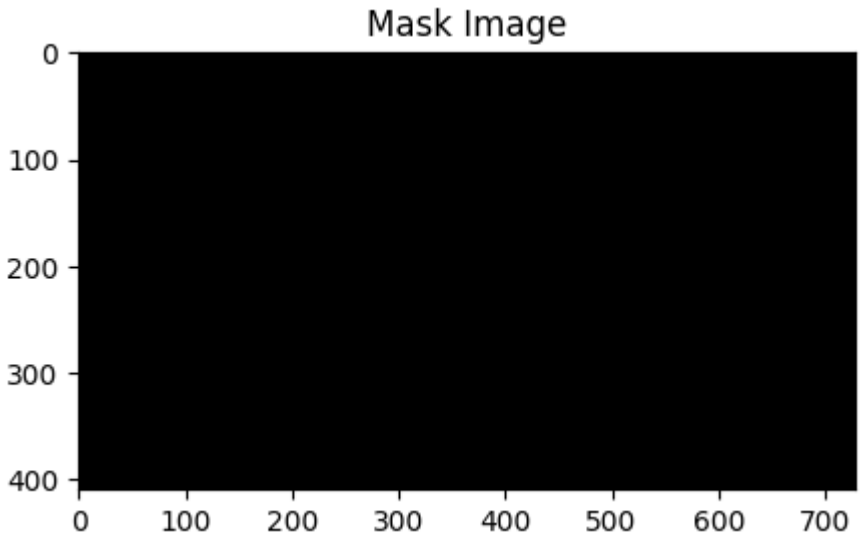
Out[25]:
```
(410, 728)
```

In [26]:
```
mask = np.zeros(img.shape[:2],dtype="uint8")
```

In [27]:
```
mask
```

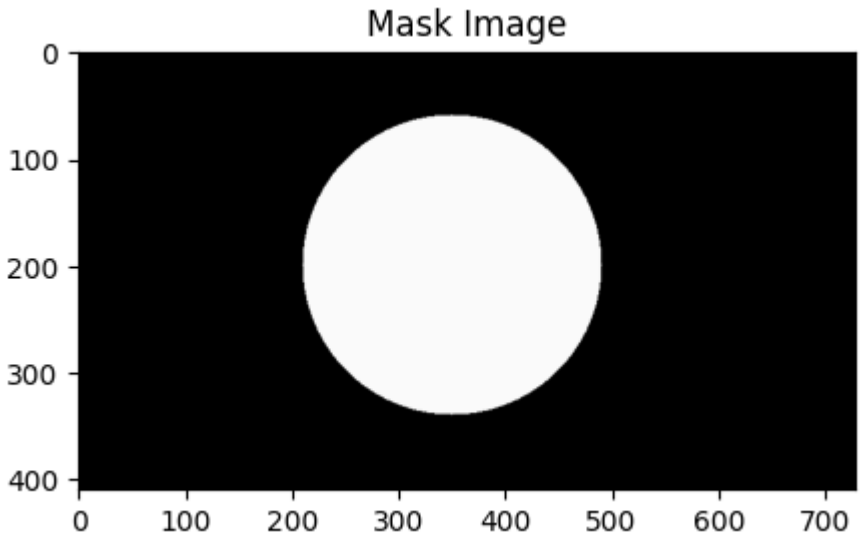
Out[27]:
```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]], dtype=uint8)
```

In [28]:
```
cv2.rectangle(mask,(100, 500), (150, 100), 0)
catImageshow("Mask Image",mask)
```



Bit_Mask

In [29]:
```
cv2.circle(mask, (350, 200), 140, 250, -1)
bit_mask = cv2.bitwise_and(img,img,mask = mask)
catImageshow("Mask Image",mask)
catImageshow("Bit_mask Image",bit_mask)
```



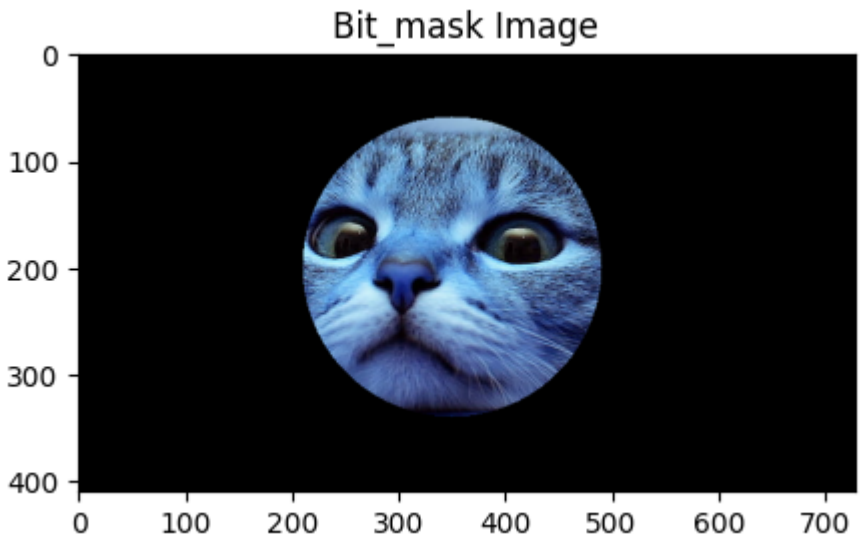


Image Scaling

In [30]:
```
img.shape
```

Out[30]: (410, 728, 3)

In [31]:
```
img/255
```

Out[31]:
```
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 [32]:
```
img.shape
```

Out[32]:
```
(410, 728, 3)
```

Resize Image

In [33]:
```
customsizeH = 120/img.shape[0]
```

In [34]:
```
customsizeW = 120/img.shape[1]
```

In [35]:
```
customsizeW
```

Out[35]:
```
0.16483516483516483
```

In [36]:
```
customsizeH
```

Out[36]:
```
0.2926829268292683
```

In [37]:
```
imgDimension = (120,int(img.shape[0]*customsizeW))
```

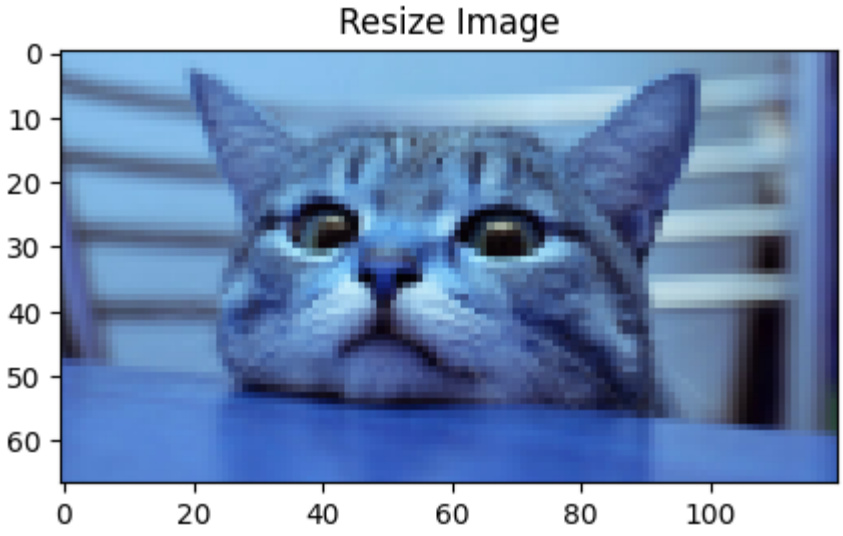
In [38]:
```
imgDimension
```

Out[38]:
```
(120, 67)
```

In [39]:
```
img.shape
```

Out[39]:
```
(410, 728, 3)
```

In [40]:
```
resizeImage = cv2.resize(img,imgDimension,interpolation = cv2.INTER_AREA)
catImageshow("Resize Image",resizeImage)
```



In [41]:
```
resizeImage.shape
```

Out[41]:
```
(67, 120, 3)
```

Rotate Image

In [42]:
```
(imageH,imageW) = img.shape[:2]
```

In [43]:
```
imageH
```

Out[43]:
```
410
```

In [44]:
```
imageW
```

Out[44]:
```
728
```

In [45]:
```
(centerX,centerY) = (imageH//2,imageW//2)
```

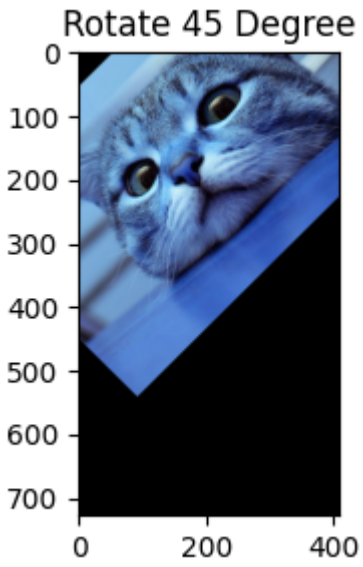
In [46]:
```
centerX
```

Out[46]:
```
205
```

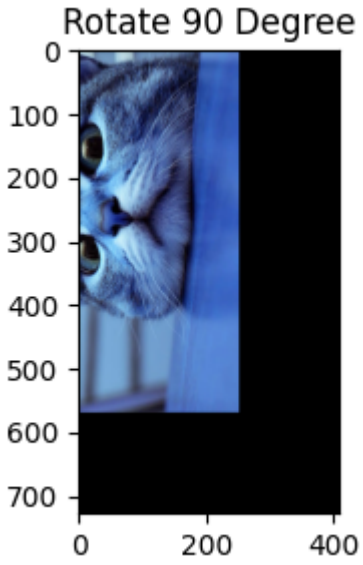
In [47]:
```
centerY
```

Out[47]:
```
364
```

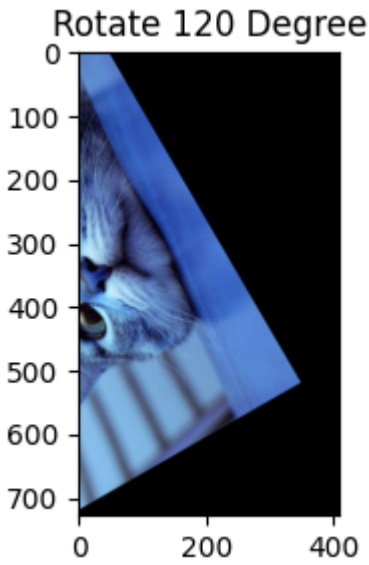
In [48]:
```
imageRorate = cv2.getRotationMatrix2D((centerX,centerY),45,1.0)
rotateNow = cv2.warpAffine(img,imageRorate,(imageH,imageW))
catImageshow("Rotate 45 Degree",rotateNow)
```



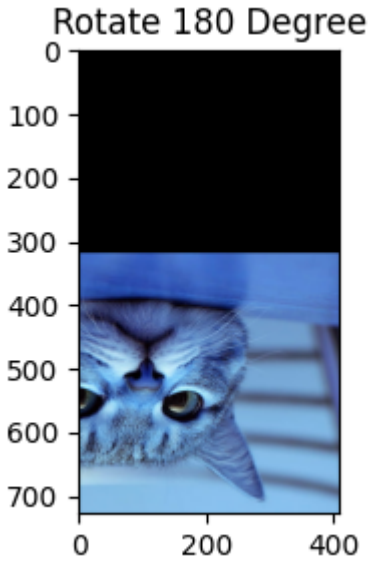
In [49]:
```
imageRorate = cv2.getRotationMatrix2D((centerX,centerY),90,1.0)
rotateNow = cv2.warpAffine(img,imageRorate,(imageH,imageW))
catImageshow("Rotate 90 Degree",rotateNow)
```



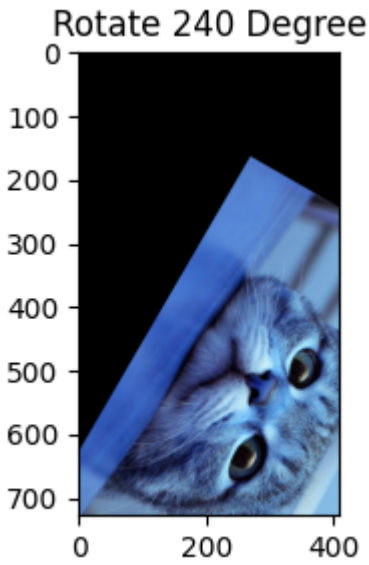
In [50]:
```
imageRorate = cv2.getRotationMatrix2D((centerX,centerY),120,1.0)
rotateNow = cv2.warpAffine(img,imageRorate,(imageH,imageW))
catImageshow("Rotate 120 Degree",rotateNow)
```



In [51]:
```
imageRorate = cv2.getRotationMatrix2D((centerX,centerY),180,1.0)
rotateNow = cv2.warpAffine(img,imageRorate,(imageH,imageW))
catImageshow("Rotate 180 Degree",rotateNow)
```



In [52]:
```
imageRorate = cv2.getRotationMatrix2D((centerX,centerY),240,1.0)
rotateNow = cv2.warpAffine(img,imageRorate,(imageH,imageW))
catImageshow("Rotate 240 Degree",rotateNow)
```



In [53]:
```
imageRorate = cv2.getRotationMatrix2D((centerX,centerY),360,1.0)
rotateNow = cv2.warpAffine(img,imageRorate,(imageH,imageW))
catImageshow("Rotate again 360 Degree",rotateNow)
```

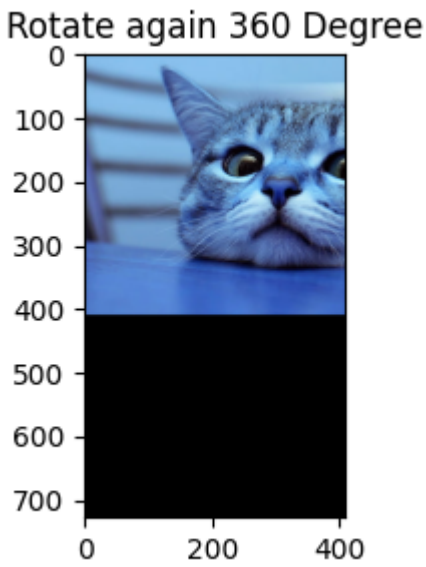
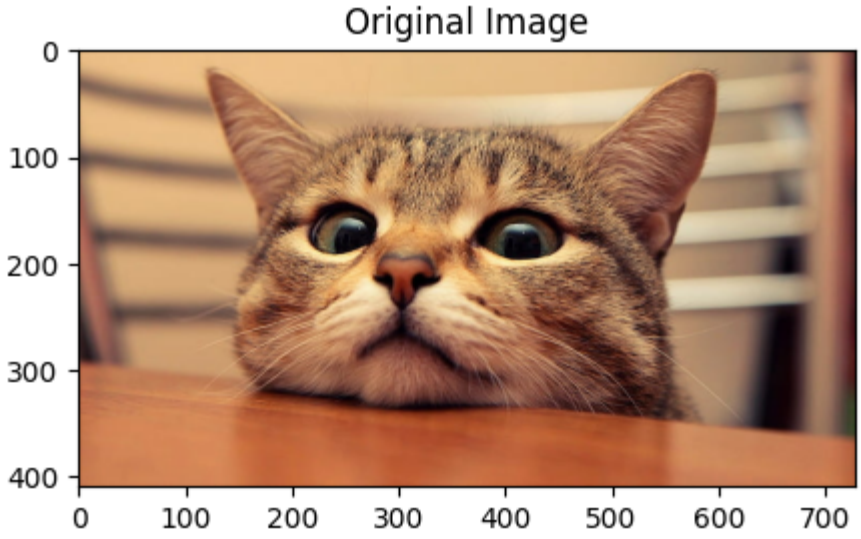
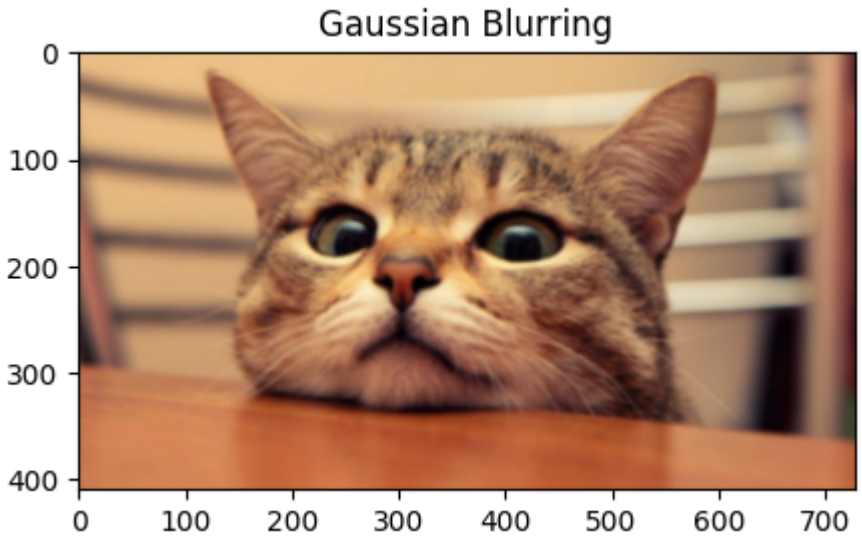
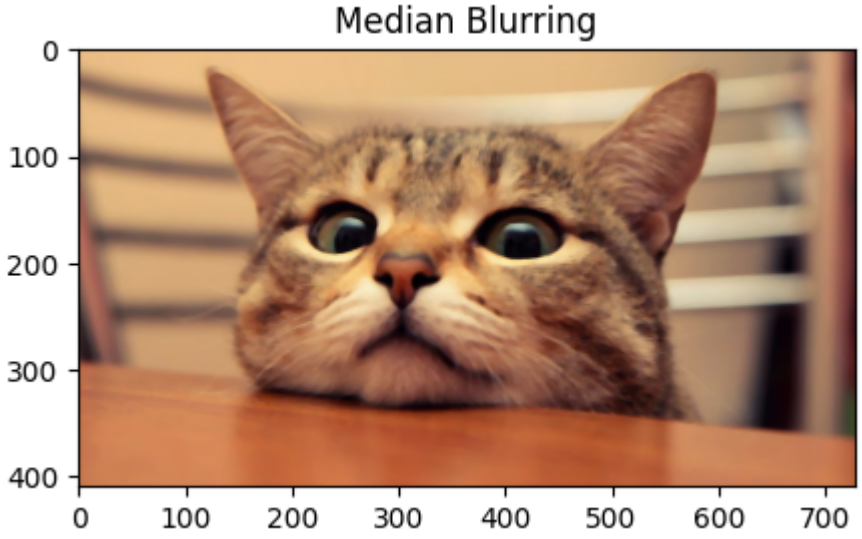


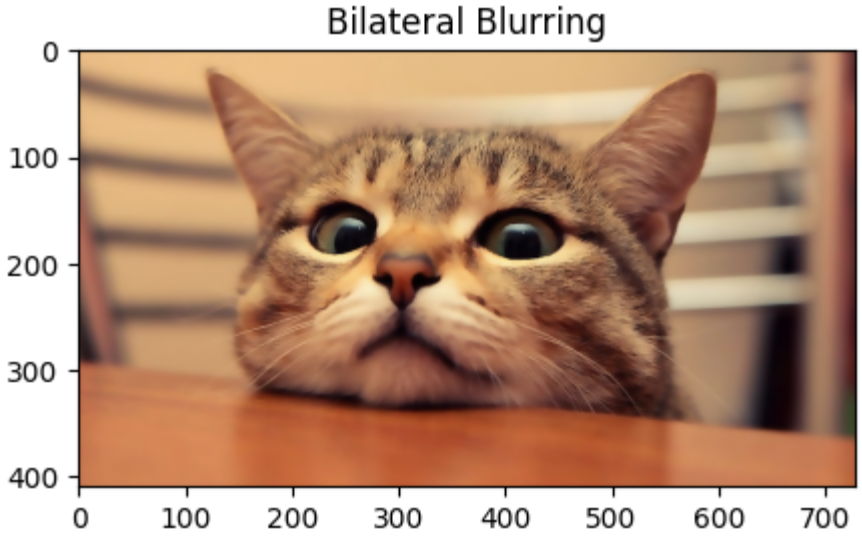
Image blurring

In [54]:
```
image = cv2.imread('cute.jpg')

catImageshow("Original Image",image)


# Gaussian Blur
Gaussian = cv2.GaussianBlur(image, (7, 7), 0)
catImageshow("Gaussian Blurring",Gaussian)

# Median Blur
median = cv2.medianBlur(image, 5)
catImageshow("Median Blurring",median)

# Bilateral Blur
bilateral = cv2.bilateralFilter(image, 9, 75, 75)
catImageshow("Bilateral Blurring",bilateral)
```









In [55]:
```
image = cv2.imread('cute.jpg')
# original image
a=0
cv2.imshow('Original Image', image)
```

```
a=cv2.waitKey(0)

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

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

# Bilateral Blur
bilateral = cv2.bilateralFilter(image, 9, 75, 75)

d=0
cv2.imshow('Bilateral Blurring', bilateral)
d=cv2.waitKey(0)
cv2.destroyAllWindows()
```

Scaling

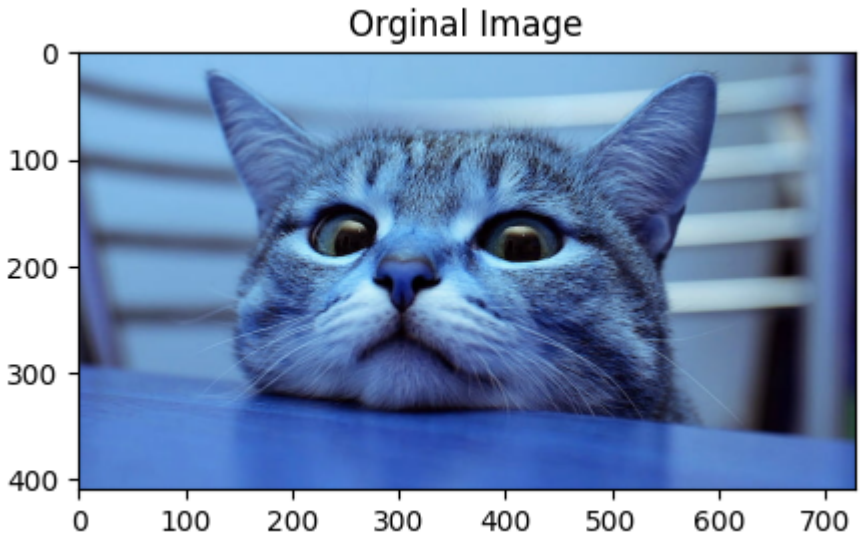
In [56]:

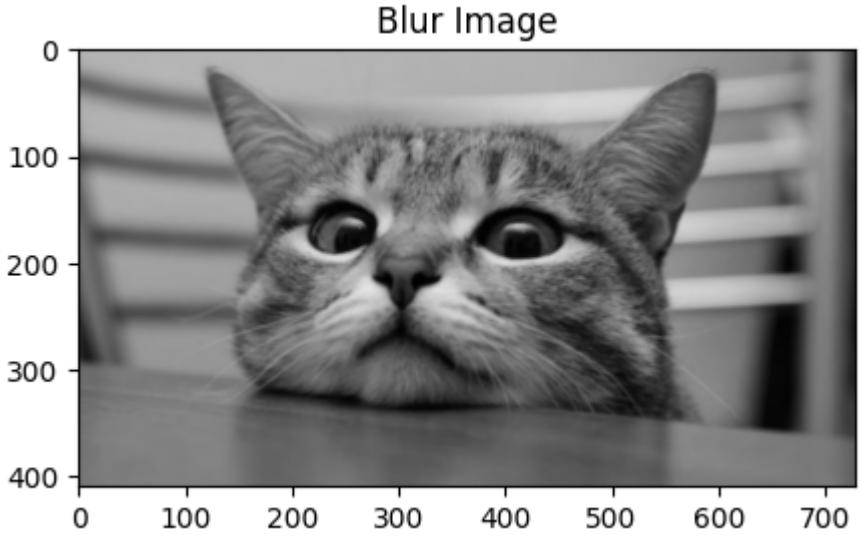
```
maxScaleUp = 100
scaleFactor = 1
#
windowName = "Resize Image"
trackbarValue = "Scale"

# 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()
```

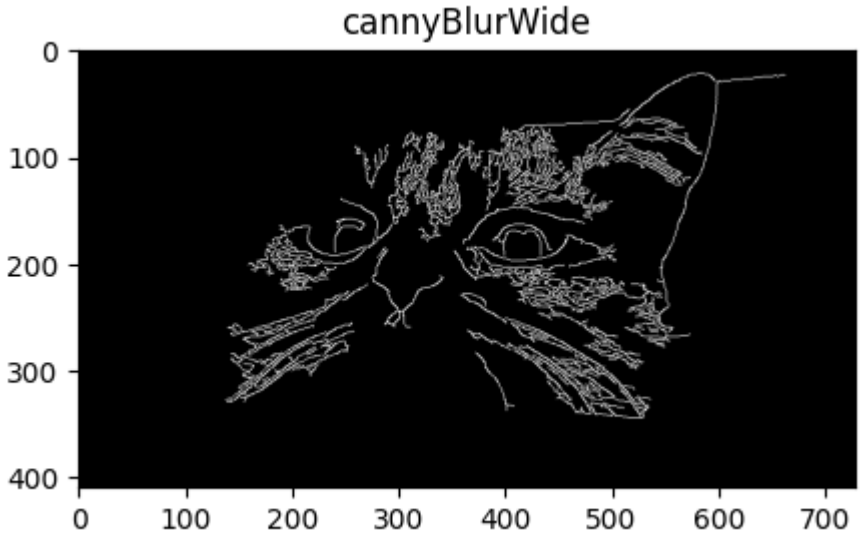
Canny Edge Detection

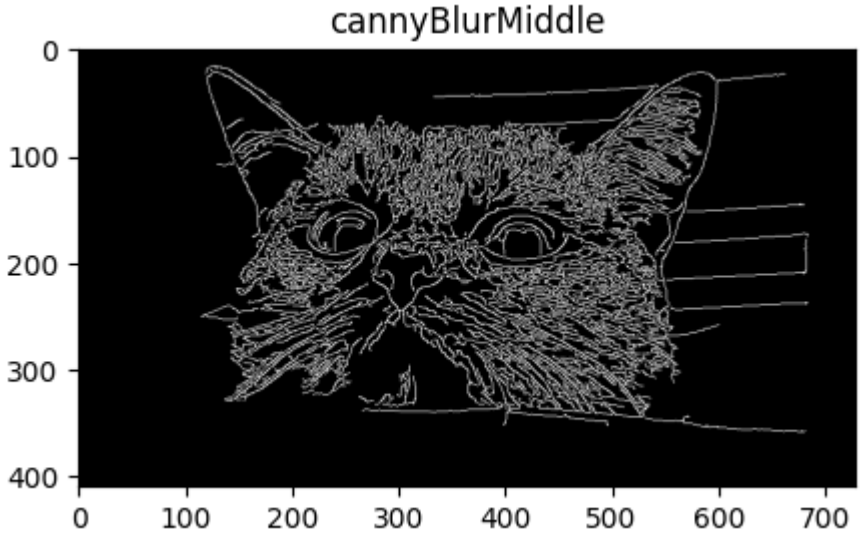
In [57]:
```
image=cv2.imread('cute.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
blurImage = cv2.GaussianBlur(gray, (5, 5), 0)
catImageshow("Orginal Image", img)
catImageshow("Blur Image", blurImage)
img_blur = cv2.GaussianBlur(img, (5,5), 0)
edges = cv2.Canny(image=img_blur, threshold1=100, threshold2=200)
```

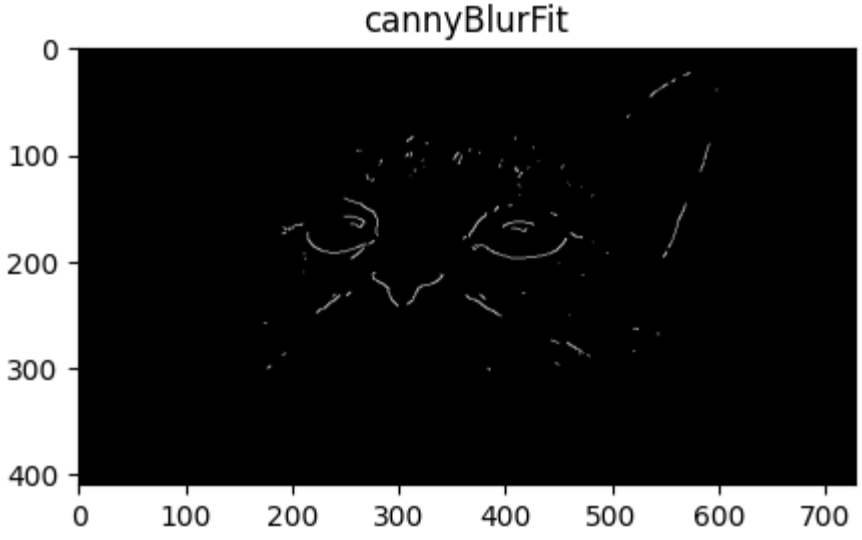




In [58]:
```
cannyBlurWide = cv2.Canny(blurImage, 20, 250)
cannyBlurMiddle = cv2.Canny(blurImage, 15, 120)
cannyBlurFit = cv2.Canny(blurImage, 240, 250)
catImageshow("cannyBlurWide",cannyBlurWide)
catImageshow("cannyBlurMiddle", cannyBlurMiddle)
catImageshow("cannyBlurFit", cannyBlurFit)
```







Sobel Edge Detection

In [59]:
```
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()
```

Created By - Piyush Dwivedi

Connect me - https://www.linkedin.com/in/piyush-dwivedi-1909a5213/

Email Address- dwivedipiyush9754@gmail.com