In [ ]:

```
import os

path = os.getcwd()
path
```

Out[ ]:

```
'h:\\Mine'
```

In [ ]:

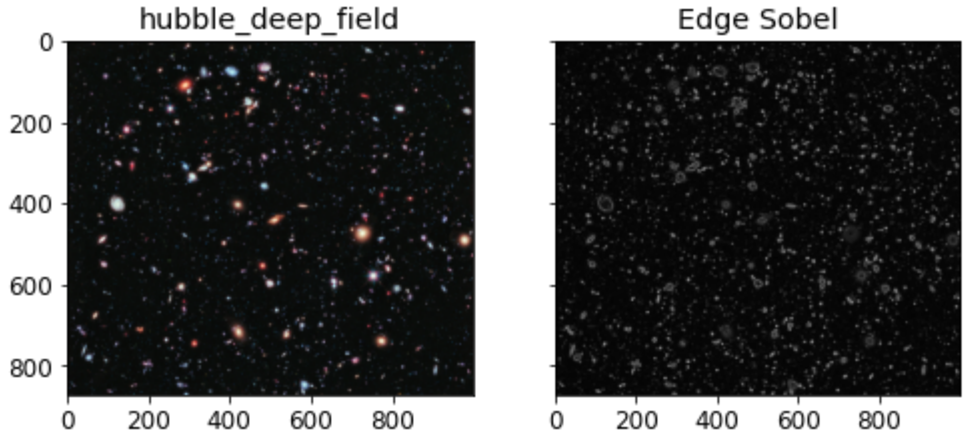
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
from skimage.io import imread, imshow
```

In [ ]:

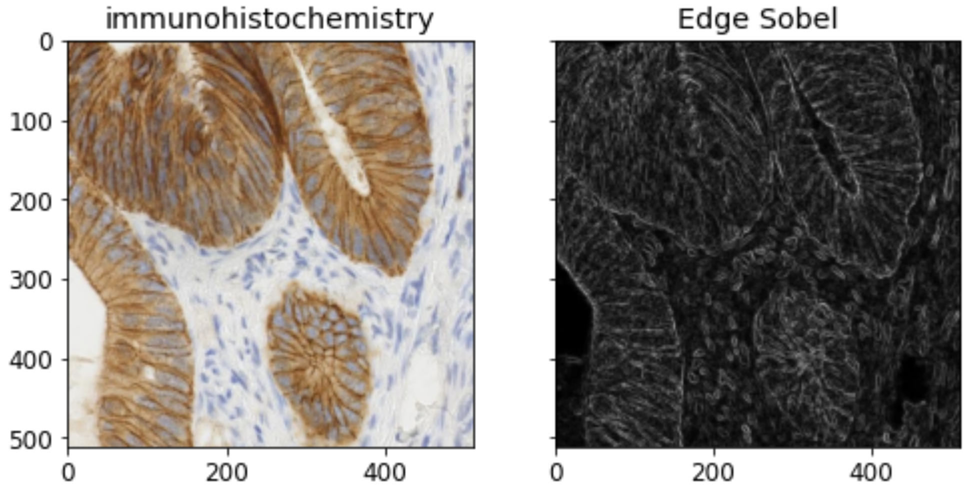
```
# Image Visualization - Sample Images

from skimage import data, filters
from skimage.color import rgb2gray
from skimage.util import compare_images
import matplotlib

matplotlib.rcParams['font.size'] =  12

images = ('hubble_deep_field', 'immunohistochemistry', 'cat', 'camera')

for name in images:
    caller = getattr(data, name)
    image = caller()
    plt.figure()
    #plt.title(name)
    print(name + str(image.shape))
    if image.ndim == 2:
        edge_sobel = filters.sobel(image)

        fig, axes = plt.subplots(ncols = 2, sharex=True,sharey=True,figsize=
        axes[0].imshow(image, cmap=plt.cm.gray)
        axes[0].set_title(name)

        axes[1].imshow(edge_sobel, cmap=plt.cm.gray)
        axes[1].set_title("Edge Sobel")
    else:
        grey = rgb2gray(image)
        edge_sobel = filters.sobel(grey)

        fig, axes = plt.subplots(ncols = 2, sharex=True,sharey=True,figsize=
        axes[0].imshow(image, cmap=plt.cm.gray)
        axes[0].set_title(name)

        axes[1].imshow(edge_sobel, cmap=plt.cm.gray)
        axes[1].set_title("Edge Sobel")

plt.show()
```

```
hubble_deep_field(872, 1000, 3)
immunohistochemistry(512, 512, 3)
cat(300, 451, 3)
camera(512, 512)
```

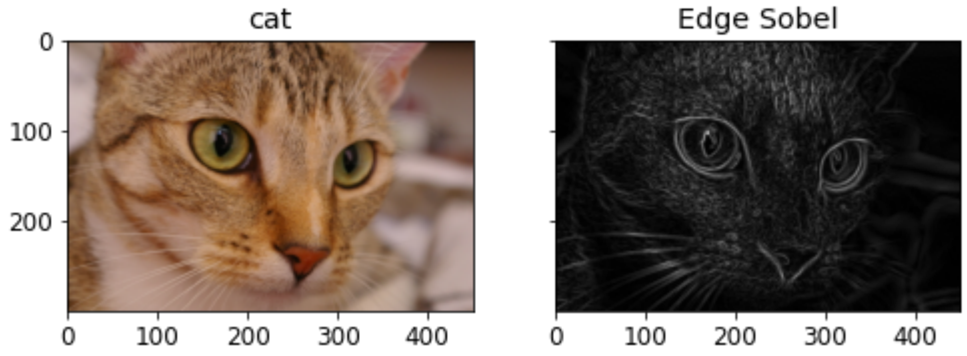
```
<Figure size 432x288 with 0 Axes>
```


hubble_deep_field
Edge Sobel
0
200
400
600
800
0
200
400
600
800
0
200
400
600
800

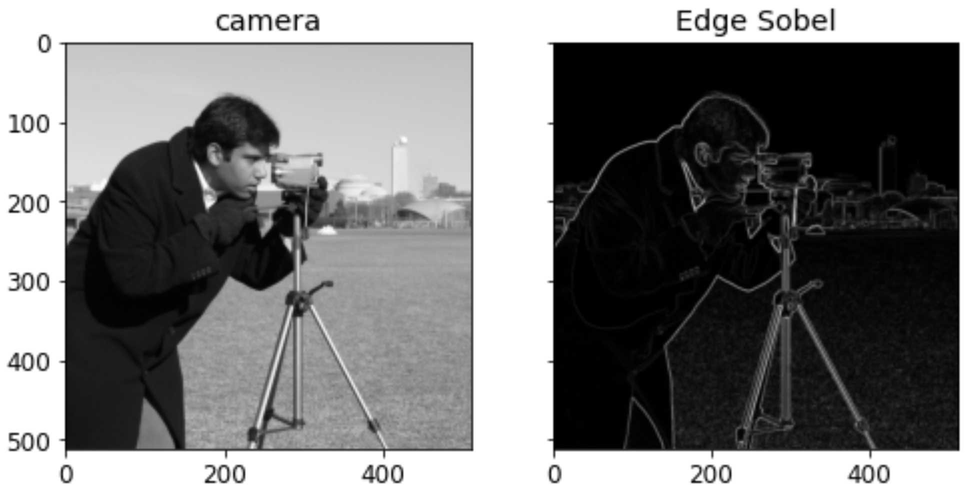
```
<Figure size 432x288 with 0 Axes>
```


immunohistochemistry
Edge Sobel
0
100
200
300
400
500
0
200
400
0
200
400

```
<Figure size 432x288 with 0 Axes>
```


cat
Edge Sobel
0
100
200
0
100
200
300
400
0
100
200
300
400

```
<Figure size 432x288 with 0 Axes>
```

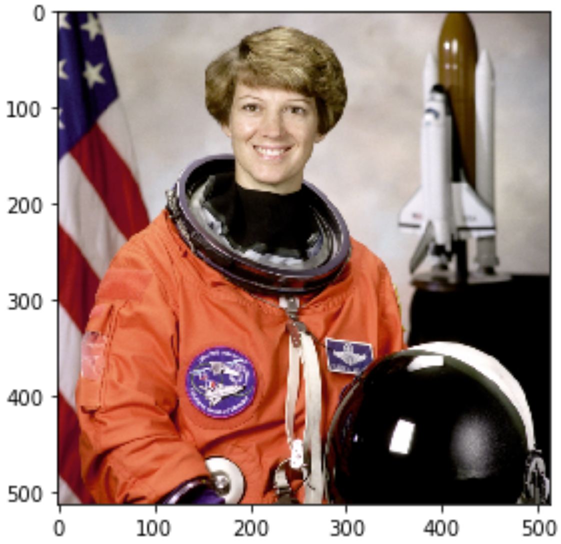


In [ ]:

```
from skimage.io import imread, imshow
from skimage import data

image = data.astronaut()
imshow(image)
```

Out[ ]:

```
<matplotlib.image.AxesImage at 0x199da8b7d90>
```



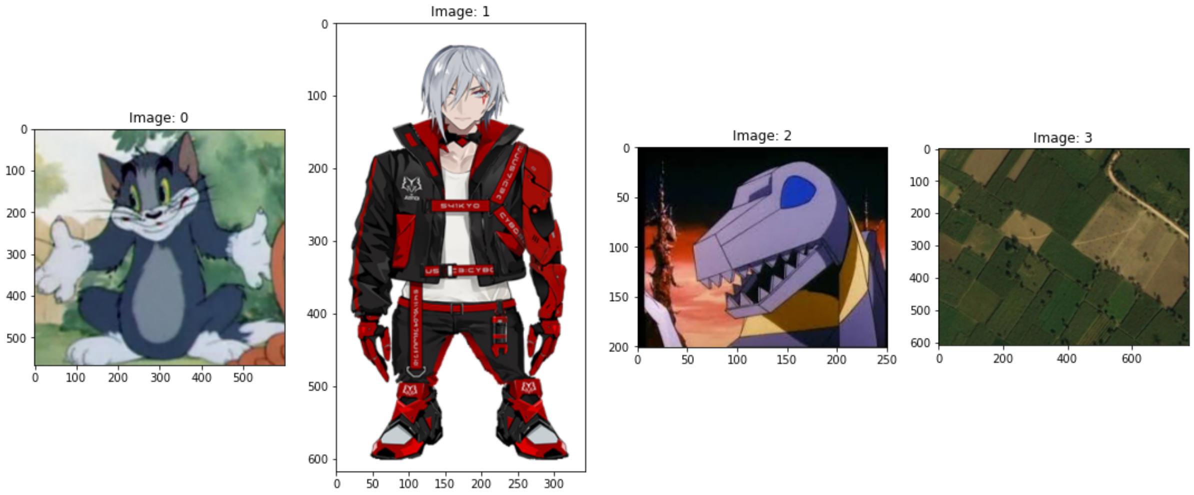
In [ ]:

```
from skimage.color import rgb2gray
from skimage import data

image = data.logo()
print(image.shape)
```

```
(500, 500, 4)
```

In [ ]:

```
# Image Visualization - Outside Images

img1 = imread("632.jpg")
img2 = imread("FMK8vSzWQAM1KIV.png")
img3 = imread("bozo.jpg")
img4 = imread("crop_field.jpg")

image_set = [img1, img2, img3, img4]

fig, axes = plt.subplots(figsize=(15,10),nrows=1, ncols=len(image_set), shar

for i in range(len(image_set)):
    axes.flat[i].imshow(image_set[i], cmap=plt.cm.gray)
    axes.flat[i].set_title("Image: " + str(i))

fig.tight_layout()
plt.show()
```

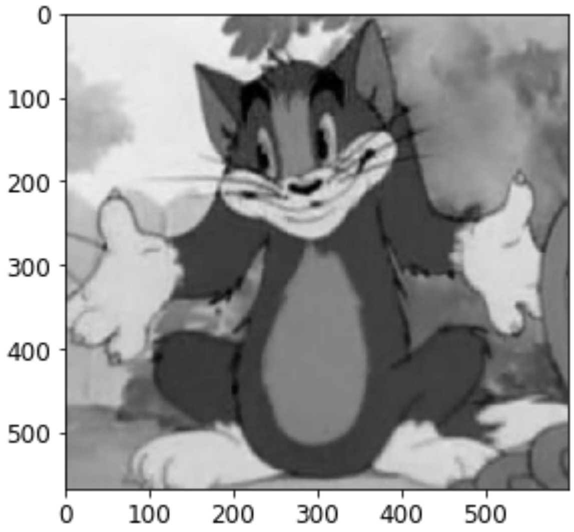


In [ ]:

```
image2 = imread("632.jpg", as_gray = True)
imshow(image2)
```

Out[ ]:

```
<matplotlib.image.AxesImage at 0x1e2951e97f0>
```



In [ ]:

```
image2 = imread("632.jpg", as_gray = True)
image1 = imread("632.jpg", as_gray = False)

print(image1.shape) # First image has three channels (RGB)
print(image2.shape) # One channel

print(image1.size)  # Channels also determine image size
print(image2.size)
```

```
(567, 599, 3)
(567, 599)
1018899
339633
```

In [ ]:

```
image2
```

Out[ ]:

```
array([[0.93108627, 0.93108627, 0.93108627, ..., 0.71704706, 0.70920392,
        0.70528235],
       [0.93108627, 0.93108627, 0.92716471, ..., 0.71704706, 0.70920392,
        0.70528235],
       [0.93108627, 0.92716471, 0.92716471, ..., 0.71704706, 0.71312549,
        0.70528235],
       ...,
       [0.59835843, 0.60228   , 0.60425216, ..., 0.44138196, 0.44138196,
        0.44138196],
       [0.60228   , 0.60228   , 0.60817373, ..., 0.43746039, 0.43746039,
        0.44138196],
       [0.60620157, 0.60620157, 0.60817373, ..., 0.43353882, 0.43353882,
        0.43746039]])
```

In [ ]:

```
image1
```

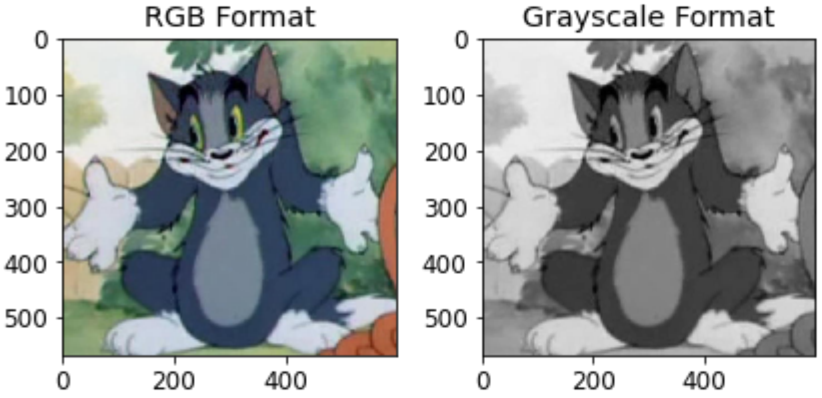
Out[ ]:

```
array([[[237, 238, 233],
        [237, 238, 233],
        [237, 238, 233],
        ...,
        [175, 190, 135],
        [173, 188, 133],
        [172, 187, 132]],

       [[237, 238, 233],
        [237, 238, 233],
        [236, 237, 232],
        ...,
        [175, 190, 135],
        [173, 188, 133],
        [172, 187, 132]],

       [[237, 238, 233],
        [236, 237, 232],
        [236, 237, 232],
        ...,
        [175, 190, 135],
        [174, 189, 134],
        [172, 187, 132]],

       ...,

       [[139, 160, 119],
        [140, 161, 120],
        [139, 162, 120],
        ...,
        [177,  98,  67],
        [177,  98,  67],
        [177,  98,  67]],

       [[140, 161, 120],
        [140, 161, 120],
        [140, 163, 121],
        ...,
        [176,  97,  66],
        [176,  97,  66],
        [177,  98,  67]],

       [[141, 162, 121],
        [141, 162, 121],
        [140, 163, 121],
        ...,
        [175,  96,  65],
        [175,  96,  65],
        [176,  97,  66]]], dtype=uint8)
```

In [ ]:

```
from skimage.color import rgb2gray

im = imread("632.jpg")
img_new = rgb2gray(im)

plt.subplot(1,2,1), imshow(im)
plt.title("RGB Format")

plt.subplot(1,2,2), imshow(img_new)
plt.title("Grayscale Format")

print(img_new.shape)
plt.show()
```

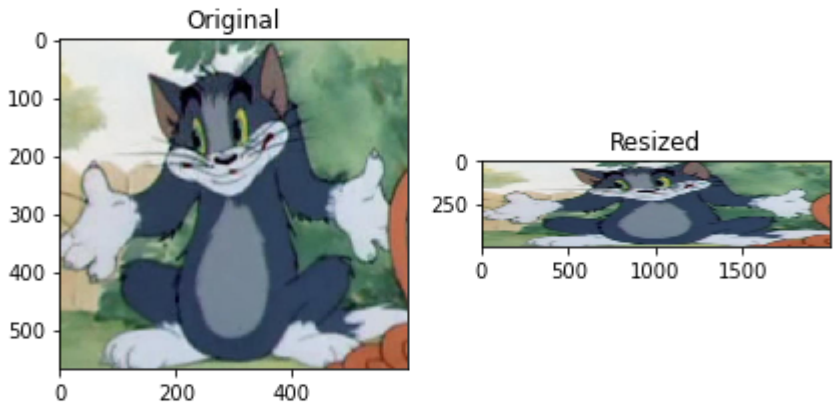
```
(567, 599)
```



In [ ]:

```
from skimage.transform import resize

img = imread("632.jpg")
img_resized = resize(img, (500,2000))
plt.subplot(121), imshow(img)
plt.title("Original")

plt.subplot(122), imshow(img_resized)
plt.title("Resized")

print("Original")
print(img.shape)
print(img.size)

print("Rescaled")
print(img_resized.shape)
print(img_resized.size)

plt.show()
```

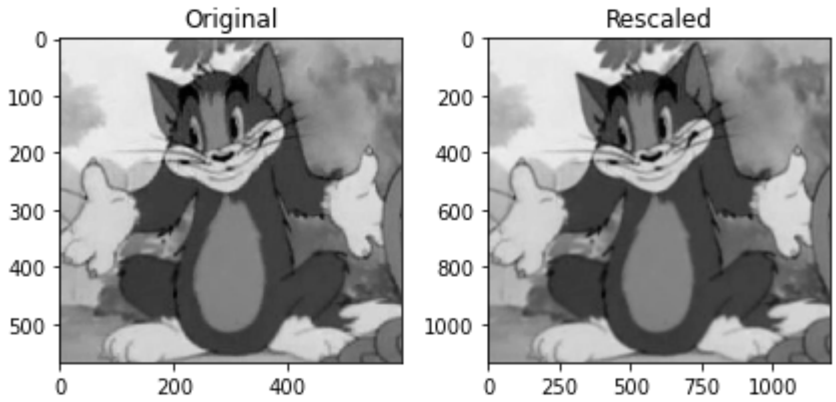
```
Original
(567, 599, 3)
1018899
Rescaled
(500, 2000, 3)
3000000
```



In [ ]:

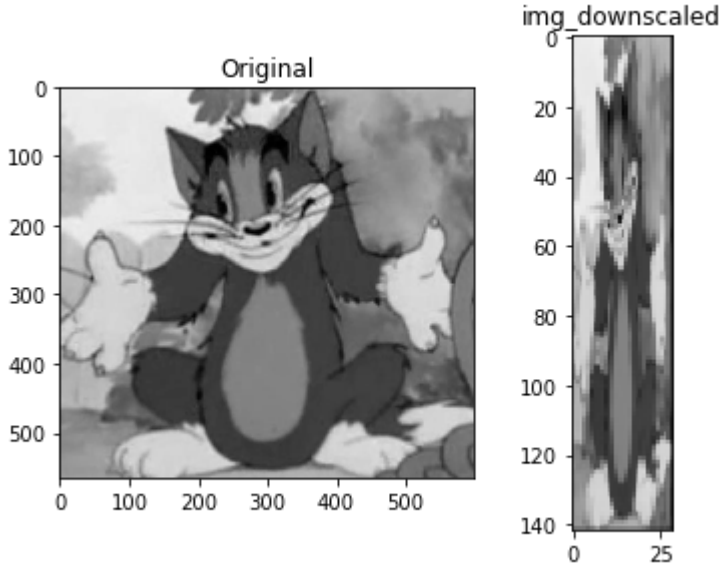
```
from skimage.transform import rescale

img = imread("632.jpg", as_gray = True)
img_rescaled = rescale(img, 2, anti_aliasing=True, anti_aliasing_sigma=0.6)

plt.subplot(121), imshow(img)
plt.title("Original")

plt.subplot(122), imshow(img_rescaled)
plt.title("Rescaled")

print("Original")
print(img.shape)
print(img.size)

print("Rescaled")
print(img_rescaled.shape)
print(img_rescaled.size)

plt.show()
```

```
D:\ImagePross\lib\site-packages\skimage\transform\_warps.py:341: UserWarnin
g: Anti-aliasing standard deviation greater than zero but not down-sampling
along all axes
  return resize(image, output_shape, order=order, mode=mode, cval=cval,
Original
(567, 599)
339633
Rescaled
(1134, 1198)
1358532
```



In [ ]:

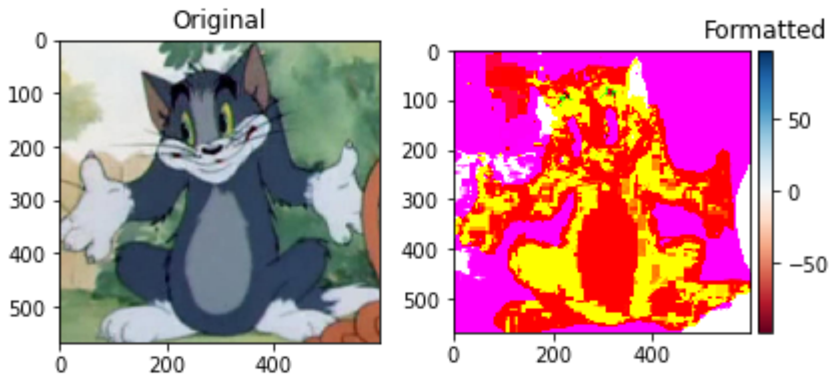
```
from skimage.transform import downscale_local_mean

img = imread("632.jpg", as_gray = True)

img_downscaled = downscale_local_mean(img, (4,21))

plt.subplot(121), imshow(img)
plt.title("Original")

plt.subplot(122), imshow(img_downscaled)
plt.title("img_downscaled")

plt.show()
```



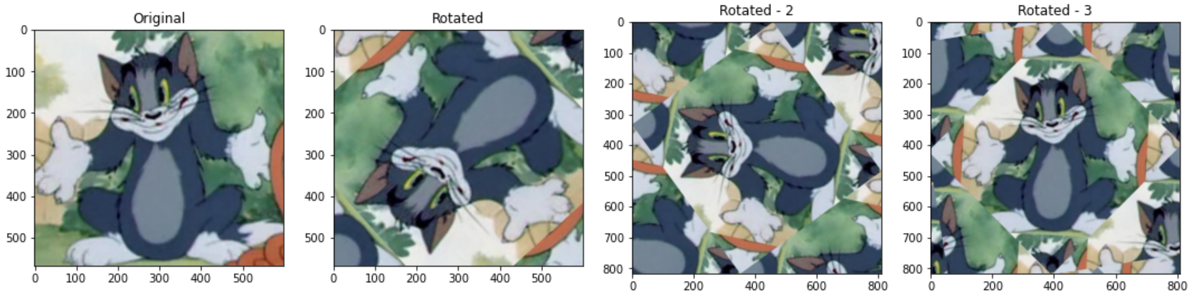
In [ ]:

```
from skimage.color import rgb2lab
img = imread("632.jpg")
im_format = rgb2lab(img)

plt.subplot(121), imshow(img)
plt.title("Original")

plt.subplot(122), imshow(im_format)
plt.title("Formatted")

print(im_format.shape)
print(im_format.size)

plt.show()
```

```
Clipping input data to the valid range for imshow with RGB data ([0..1] for
floats or [0..255] for integers).
(567, 599, 3)
1018899
```



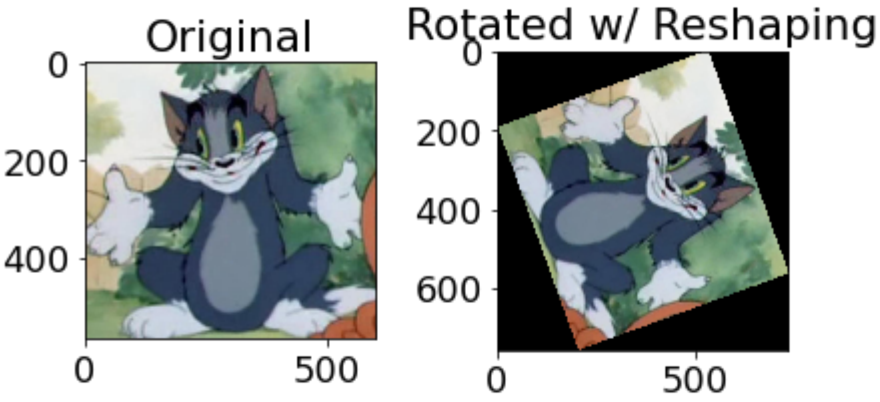
```
In [ ]: from skimage.transform import rotate

        angle = 132

        img = imread("632.jpg")
        rotated = rotate(img,angle,resize=False, mode='wrap')
        rotated2 = rotate(rotated,-54,resize=True, mode='wrap')
        rotated3 = rotate(rotated2,-81,resize=False, mode='wrap')


        fig, ax = plt.subplots(ncols=4, sharex=False, sharey=False, figsize=(15,4))

        ax[0].imshow(img,cmap=plt.cm.gray)
        ax[0].set_title("Original")

        ax[1].imshow(rotated,cmap=plt.cm.gray)
        ax[1].set_title("Rotated")

        ax[2].imshow(rotated2,cmap=plt.cm.gray)
        ax[2].set_title("Rotated - 2")

        ax[3].imshow(rotated3,cmap=plt.cm.gray)
        ax[3].set_title("Rotated - 3")

        plt.tight_layout()
        plt.show()
```



In [ ]:

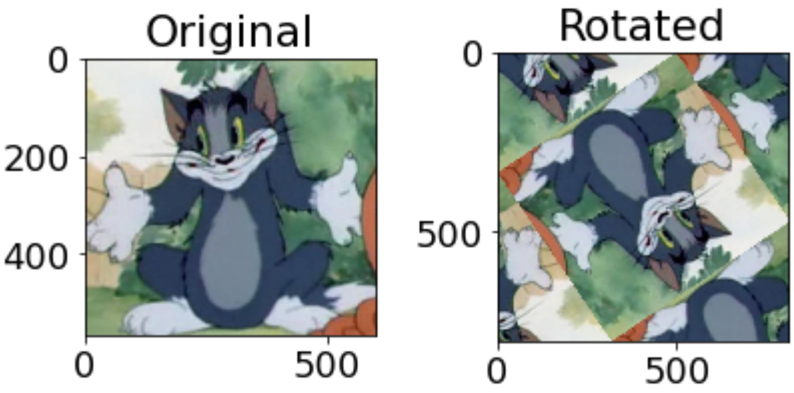
```
from scipy.ndimage.interpolation import rotate

img = imread("632.jpg")
rotated = rotate(img, -70, reshape=True)

plt.subplot(121), imshow(img)
plt.title("Original")
plt.subplot(122), imshow(rotated)
plt.title("Rotated w/ Reshaping")

plt.show()
```

```
C:\Users\ibtid\AppData\Local\Temp\ipykernel_6924\3470409349.py:1: Deprecatio
nWarning: Please use `rotate` from the `scipy.ndimage` namespace, the `scip
y.ndimage.interpolation` namespace is deprecated.
  from scipy.ndimage.interpolation import rotate
```



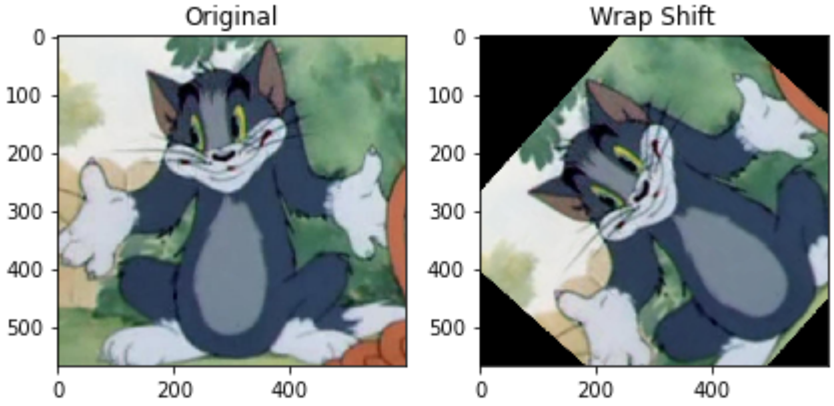
In [ ]:

```
from skimage.transform import rotate

angle = 214

img = imread("632.jpg")
rotated = rotate(img,angle,resize=True, mode='wrap')

plt.subplot(121), imshow(img)
plt.title("Original")

plt.subplot(122), imshow(rotated)
plt.title("Rotated")

plt.show()
```



In [ ]:

```
# Image shifting can add shift-invariance to images via changing the positio
# Being a geometric transformation, it maps the object to a new set of (x,y)
# x' = x + dx
# y' = y + dy

from skimage.transform import rotate, AffineTransform, warp

img = imread("632.jpg")

transform = AffineTransform(translation=(300,-180),rotation = 0.84)  # Rotati
wrapshift = warp(img,transform,mode='constant')
plt.subplot(121), imshow(img)
plt.title("Original")

plt.subplot(122), imshow(wrapshift)
plt.title("Wrap Shift")

plt.show()
```



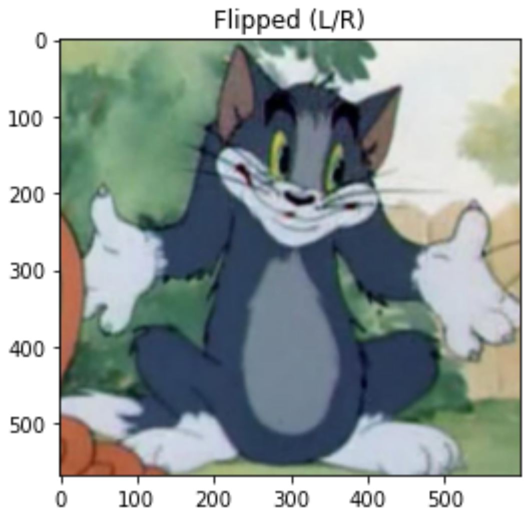
In [ ]:

```
# Fliplr mirrors the pixel values of each image.

import numpy as np

img = imread("632.jpg")

flipLR = np.fliplr(img)

print(np.reshape(img, (567*599*3)))
print(np.reshape(flipLR, (567*599*3))) # Each pixel feature should be differ

print(img.size) # But sizes remain constant
print(flipLR.size)

plt.imshow(flipLR)
plt.title("Flipped (L/R)")
```

```
[237 238 233 ... 176  97  66]
[172 187 132 ... 141 162 121]
1018899
1018899
```

Out[ ]:

```
Text(0.5, 1.0, 'Flipped (L/R)')
```



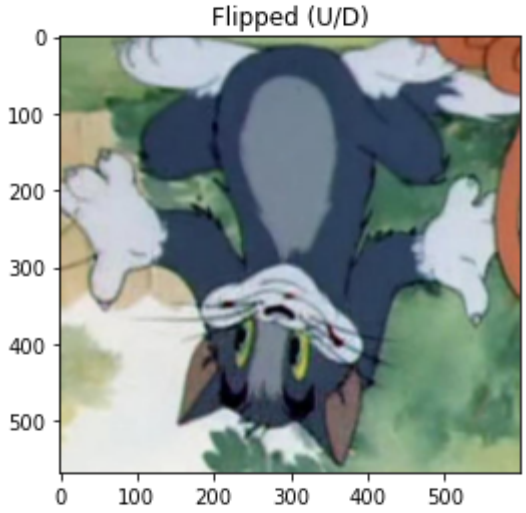
In [ ]:

```
import numpy as np

img = imread("632.jpg")
flipUD = np.flipud(img)

print(np.reshape(flipUD, (567*599*3)))

plt.imshow(flipUD)
plt.title("Flipped (U/D)")
```

```
[141 162 121 ... 172 187 132]
```

Out[ ]:

```
Text(0.5, 1.0, 'Flipped (U/D)')
```



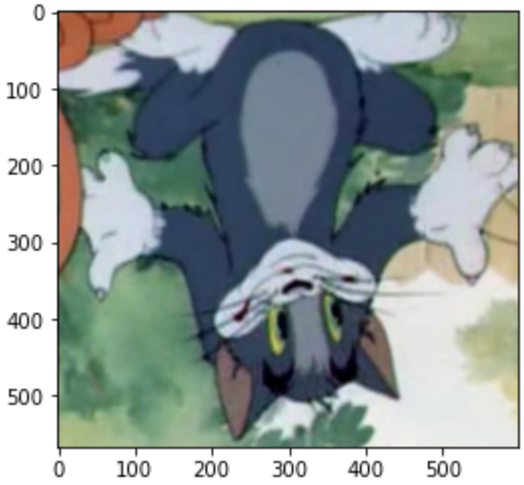
In [ ]:

```
flipUD = np.fliplr(flipUD)
print(np.reshape(flipUD, (567*599*3)))
plt.imshow(flipUD)
```

```
[176  97  66 ... 237 238 233]
```

Out[ ]:

```
<matplotlib.image.AxesImage at 0x199d4bf9880>
```



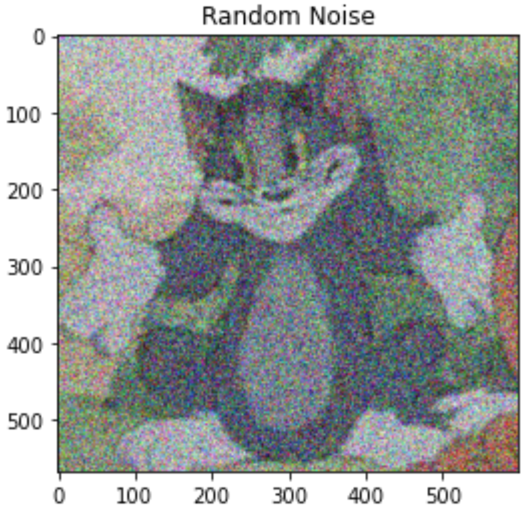
In [ ]:

```
from skimage.util import random_noise

img = imread("632.jpg")
sigma = 0.674 # Standard Deviation
noisyrandom = random_noise(img,var=sigma**2)

print(np.reshape(noisyrandom, (567*599*3)))

plt.imshow(noisyrandom)
plt.title("Random Noise")
```

```
[1.         0.98885346 1.         ... 0.78177447 1.         0.43443474]
```

Out[ ]:

```
Text(0.5, 1.0, 'Random Noise')
```



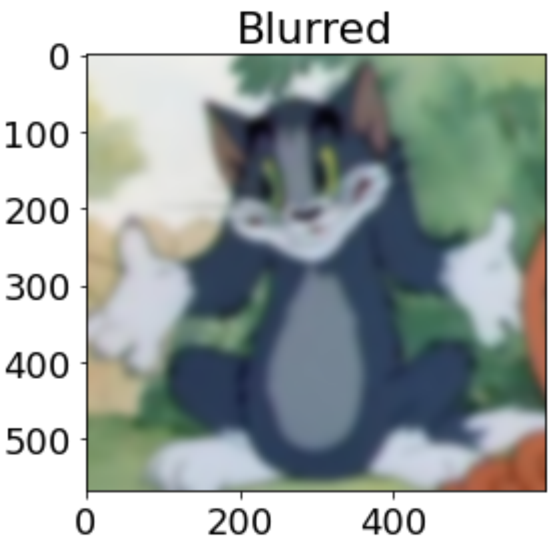
In [ ]:

```
from skimage.filters import gaussian

img = imread("632.jpg")
blurred = gaussian(img, sigma=6, multichannel=True)

plt.imshow(blurred)
plt.title("Blurred")
```

```
C:\Users\ibtid\AppData\Local\Temp\ipykernel_6924\2832857498.py:4: FutureWarn
ing: `multichannel` is a deprecated argument name for `gaussian`. It will be
removed in version 1.0. Please use `channel_axis` instead.
  blurred = gaussian(img, sigma=6, multichannel=True)
```

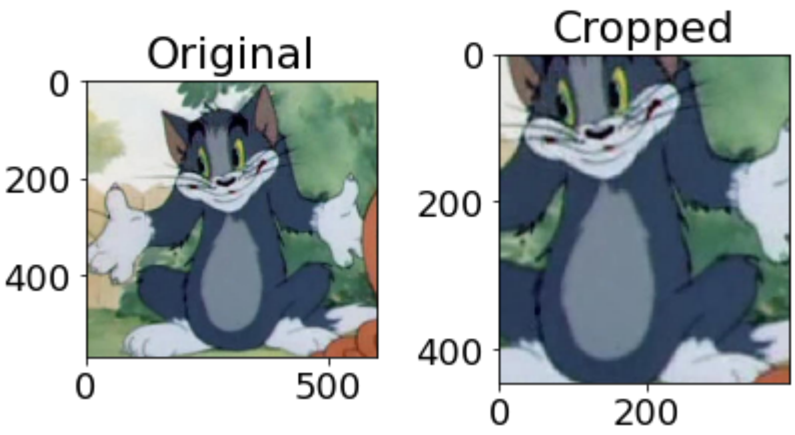
Out[ ]:

```
Text(0.5, 1.0, 'Blurred')
```



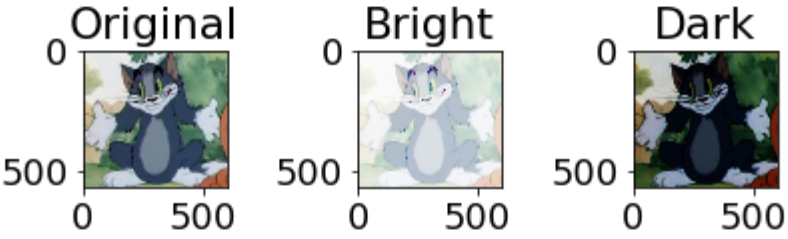
In [ ]:

```
img = imread("632.jpg")
cropped = img[100: (img.shape[0] - 21), 150:(img.shape[1] - 54)]

plt.subplot(121), imshow(img)
plt.title("Original")
plt.subplot(122), imshow(cropped)
plt.title("Cropped")

plt.show()
```



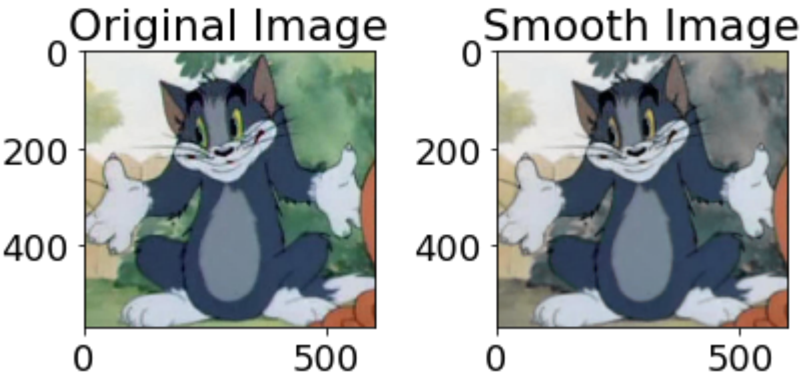
In [ ]:

```
from skimage import exposure

img = imread("632.jpg")

img_bright = exposure.adjust_gamma(img, gamma=0.2, gain=1) # < 1 gamma
img_dark = exposure.adjust_gamma(img, gamma=2.5, gain=1) # > 1 gamma

plt.subplot(131), imshow(img)
plt.title("Original")
plt.subplot(132), imshow(img_bright)
plt.title("Bright")
plt.subplot(133), imshow(img_dark)
plt.title("Dark")

plt.show()
```



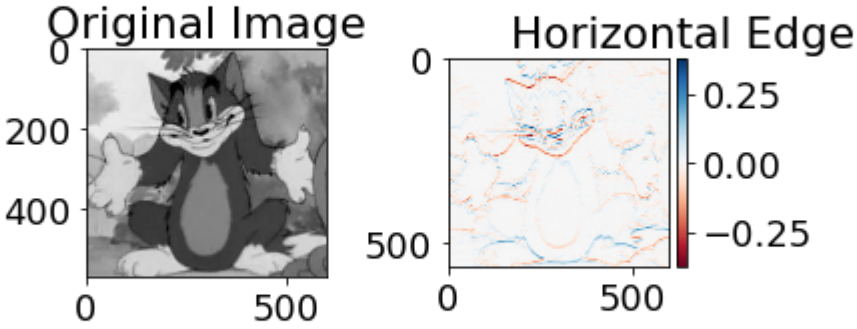
In [ ]:

```
from skimage.filters import median # Of neighboring pixels for each pixel on

img = imread("632.jpg")
img_median = median(img )

plt.subplot(121), imshow(img)
plt.title('Original Image')

plt.subplot(122),imshow(img_median)
plt.title('Smooth Image')

plt.show()
```



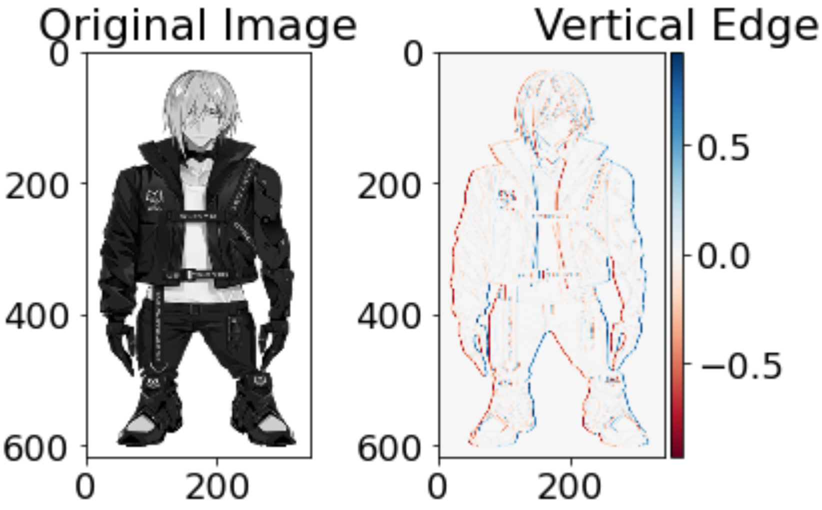
In [ ]:

```
from skimage.filters import sobel_h

img = imread("632.jpg", as_gray=True)
img_sobelh = sobel_h(img)

plt.subplot(121), imshow(img)
plt.title('Original Image')

plt.subplot(122),imshow(img_sobelh)
plt.title('Horizontal Edge')

plt.show()
```



In [ ]:

```
from skimage.filters import sobel_v

img = imread("FMK8vSzWQAM1KIV.png", as_gray=True)
img_sobelv = sobel_v(img)

plt.subplot(121), imshow(img)
plt.title('Original Image')

plt.subplot(122),imshow(img_sobelv)
plt.title('Vertical Edge')

plt.show()
```



In [ ]:

```
image = imread('FMK8vSzWQAM1KIV.png', as_gray=True)
print(image.shape)

pixel_gray = np.reshape(image, (617 * 344)) # Pixel Features
pixel_gray
```

```
(617, 344)
```

Out[ ]:

```
array([1., 1., 1., ..., 1., 1., 1.])
```

In [ ]:

```
image = imread('FMK8vSzWQAM1KIV.png')
print(image.shape)

pixel_rgb = np.reshape(image, (617 * 344 * 4)) # Pixel Features
pixel_rgb
```

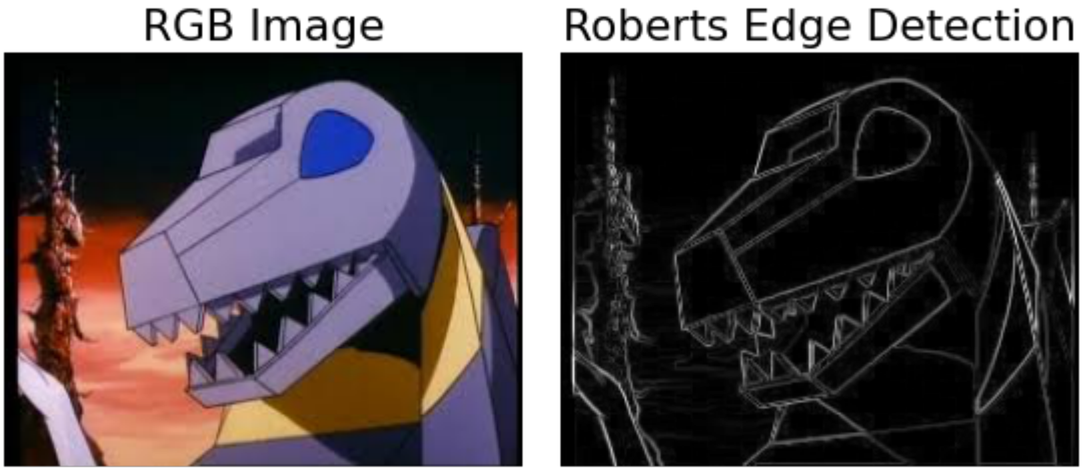
```
(617, 344, 4)
```

Out[ ]:

```
array([255, 255, 255, ..., 255, 255, 255], dtype=uint8)
```

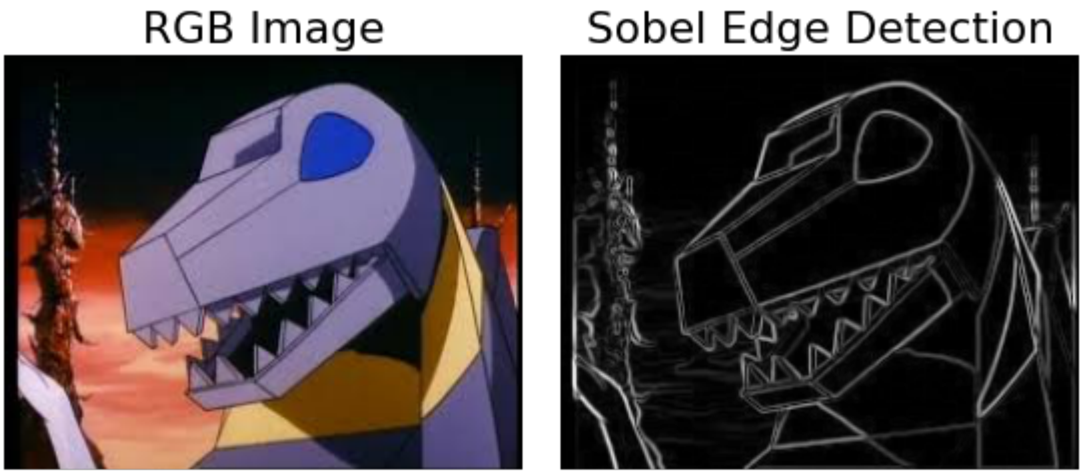
In [ ]:

```
import numpy as np
import matplotlib.pyplot as plt

from skimage import filters
from skimage.util import compare_images

image_field = imread('bozo.jpg', as_gray=True)
image_field_rgb = imread('bozo.jpg', as_gray=False)

edge_roberts = filters.roberts(image_field) # MUST BE a Two-Dimensional imag

fig, axes = plt.subplots(ncols=2,sharex=True,sharey=True,figsize=(8,4))

# Column 1
axes[0].imshow(image_field_rgb, cmap=plt.cm.gray)
axes[0].set_title("RGB Image")

# Column 2
axes[1].imshow(edge_roberts, cmap=plt.cm.gray)
axes[1].set_title("Roberts Edge Detection")

for ax in axes:
    ax.axis('off')

plt.tight_layout()
plt.show()
```



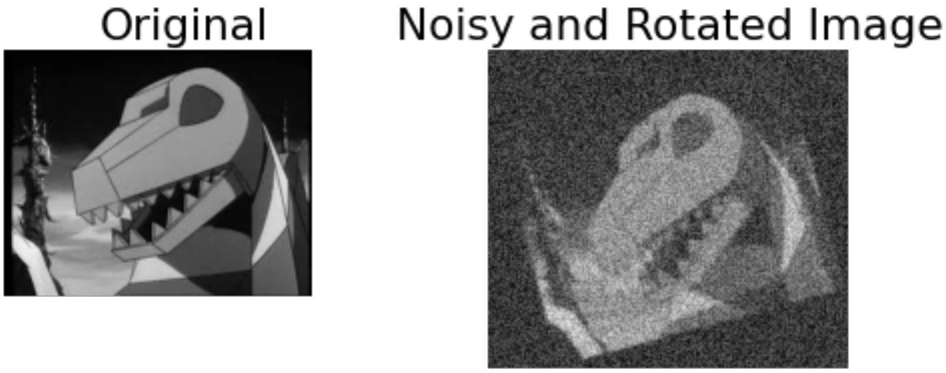
In [ ]:

```
import numpy as np
import matplotlib.pyplot as plt

from skimage import filters
from skimage.util import compare_images

image_field = imread('bozo.jpg', as_gray=True)
image_field_rgb = imread('bozo.jpg', as_gray=False)

edge_sobel = filters.sobel(image_field)  # MUST BE a Two-Dimensional image ar

fig, axes = plt.subplots(ncols=2,sharex=True,sharey=True,figsize=(8,4))

# Column 1
axes[0].imshow(image_field_rgb, cmap=plt.cm.gray)
axes[0].set_title("RGB Image")

# Column 2
axes[1].imshow(edge_sobel, cmap=plt.cm.gray)
axes[1].set_title("Sobel Edge Detection")

for ax in axes:
    ax.axis('off')

fig.tight_layout()
plt.show()
```



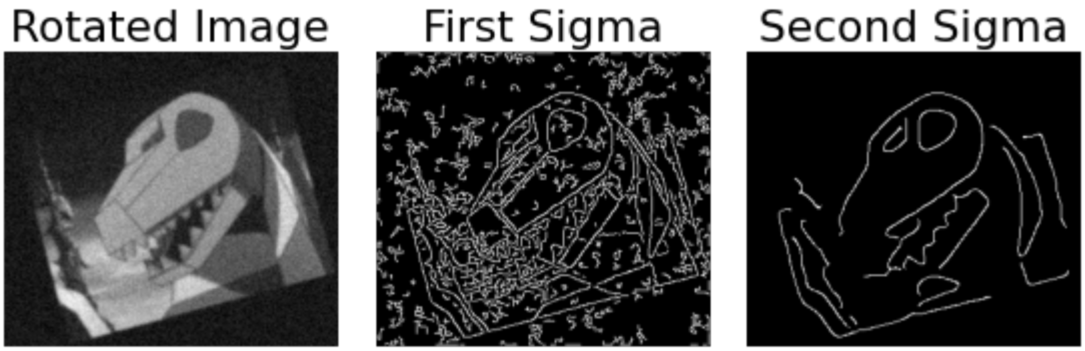
In [ ]:

```
# Canny Edge Detector


import numpy as np
import matplotlib.pyplot as plt
from scipy import ndimage as ndi

from skimage import feature

img = imread("bozo.jpg", as_gray=True)

img_noisy = ndi.rotate(img, 15, mode='constant')
img_noisy = ndi.gaussian_filter(img_noisy, 1)
img_noisy += 0.7 * np.random.random(img_noisy.shape)

fig, (ax1, ax2) = plt.subplots(nrows=1, ncols=2, figsize=(7, 3),sharex=True,

ax1.imshow(img, cmap=plt.cm.gray)
ax1.axis('off')
ax1.set_title("Original")

ax2.imshow(img_noisy, cmap=plt.cm.gray)
ax2.axis("off")
ax2.set_title("Noisy and Rotated Image")

fig.tight_layout()
plt.show()
```



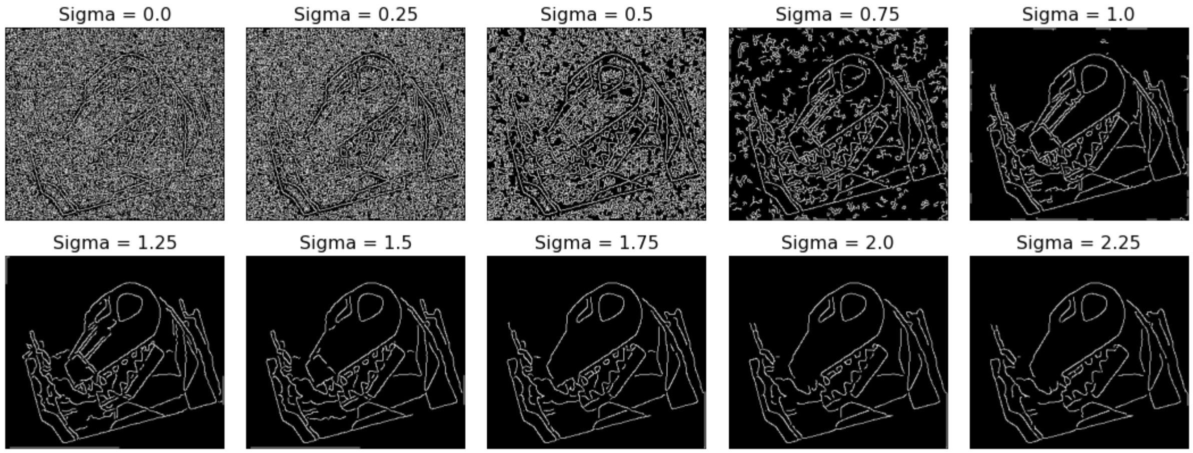
In [ ]:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy import ndimage as ndi

from skimage import feature

img = imread("bozo.jpg", as_gray=True)

img_noisy = ndi.rotate(img, 15, mode='constant')
img_noisy = ndi.gaussian_filter(img_noisy, 1)
img_noisy += 0.2 * np.random.random(img_noisy.shape)

edges1 = feature.canny(img_noisy, sigma=1)
edges2 = feature.canny(img_noisy, sigma=4)

fig, (ax1, ax2, ax3) = plt.subplots(nrows=1, ncols=3, figsize=(8, 3),sharex=

ax1.imshow(img_noisy, cmap=plt.cm.gray)
ax1.axis('off')
ax1.set_title("Rotated Image")

ax2.imshow(edges1, cmap=plt.cm.gray)
ax2.axis("off")
ax2.set_title("First Sigma")

ax3.imshow(edges2, cmap=plt.cm.gray)
ax3.axis("off")
ax3.set_title("Second Sigma")

fig.tight_layout()
plt.show()
```



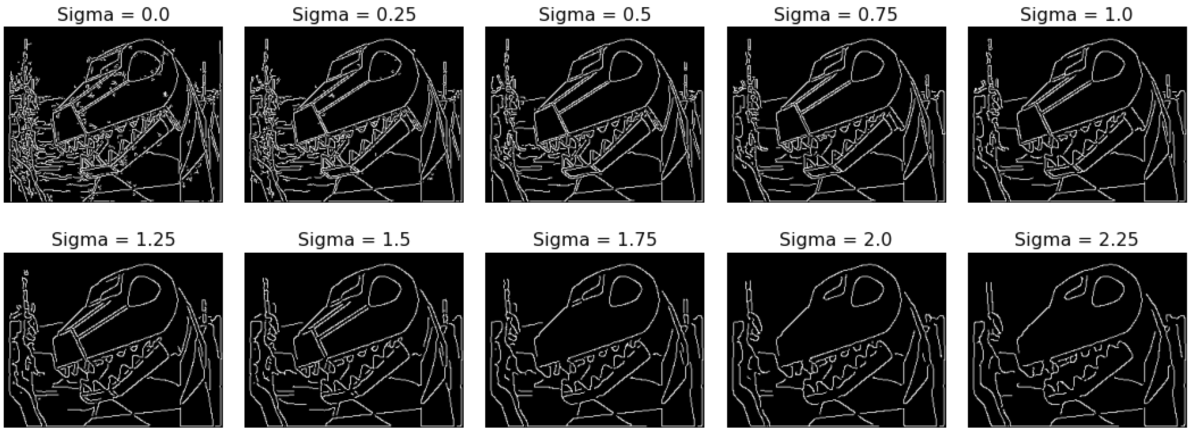
In [ ]:

```
# Canny Filter Set by Increasing Sigma - Noisy Image

import numpy as np
import matplotlib.pyplot as plt
from scipy import ndimage as ndi

from skimage import feature

img = imread("bozo.jpg", as_gray=True)

img_noisy = ndi.rotate(img, 15, mode='constant')
img_noisy = ndi.gaussian_filter(img_noisy, 1)
img_noisy += 0.2 * np.random.random(img_noisy.shape)

edges = []

for i in [0.25*j for j in range(1,11)]:
    edges.append(feature.canny(img_noisy, sigma=i))

fig, axes = plt.subplots(figsize=(15,6),nrows=2, ncols=5, sharex=True, share

for i in range(10):
    axes.flat[i].imshow(edges[i],cmap=plt.cm.gray)
    axes.flat[i].set_axis_off()
    axes.flat[i].set_title("Sigma = {}".format(0.25*i),fontsize=16)
fig.tight_layout()
plt.show()
```



In [ ]:

```
# Canny Filter Set by Increasing Sigma - Original Image

import numpy as np
import matplotlib.pyplot as plt
from scipy import ndimage as ndi

from skimage import feature

img = imread("bozo.jpg", as_gray=True)

edges = []

for i in [0.25*j for j in range(1,11)]:
    edges.append(feature.canny(img, sigma=i))

fig, axes = plt.subplots(figsize=(15,6),nrows=2, ncols=5, sharex=True, share

for i in range(10):
    axes.flat[i].imshow(edges[i],cmap=plt.cm.gray)
    axes.flat[i].set_axis_off()
    axes.flat[i].set_title("Sigma = {}".format(0.25*i),fontsize=16)
fig.tight_layout()
plt.show()
```



In [ ]: