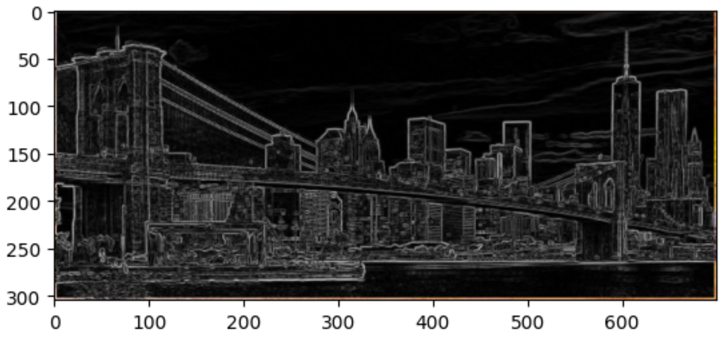
1st Edge Detection Techniques

```
import matplotlib.pyplot as plt
vertical_filter = [[-1,-2,-1], [0,0,0], [1,2,1]]
horizontal_filter = [[-1,0,1], [-2,0,2], [-1,0,1]]
img = plt.imread('bridge.png')
n,m,d = img.shape
edges_img = img.copy()
for row in range(3, n-2):
    for col in range(3, m-2):
        local_pixels = img[row-1:row+2, col-1:col+2, 0]
        vertical_transformed_pixels = vertical_filter*local_pixels
        vertical_score = vertical_transformed_pixels.sum()/4
        horizontal_transformed_pixels = horizontal_filter*local_pixels
        horizontal_score = horizontal_transformed_pixels.sum()/4
        edge_score = (vertical_score**2 + horizontal_score**2)**.5
        edges_img[row, col] = [edge_score]*3
edges_img = edges_img/edges_img.max()
plt.imshow(edges_img)
```

```
<matplotlib.image.AxesImage at 0x78b8eea6cd90>
```



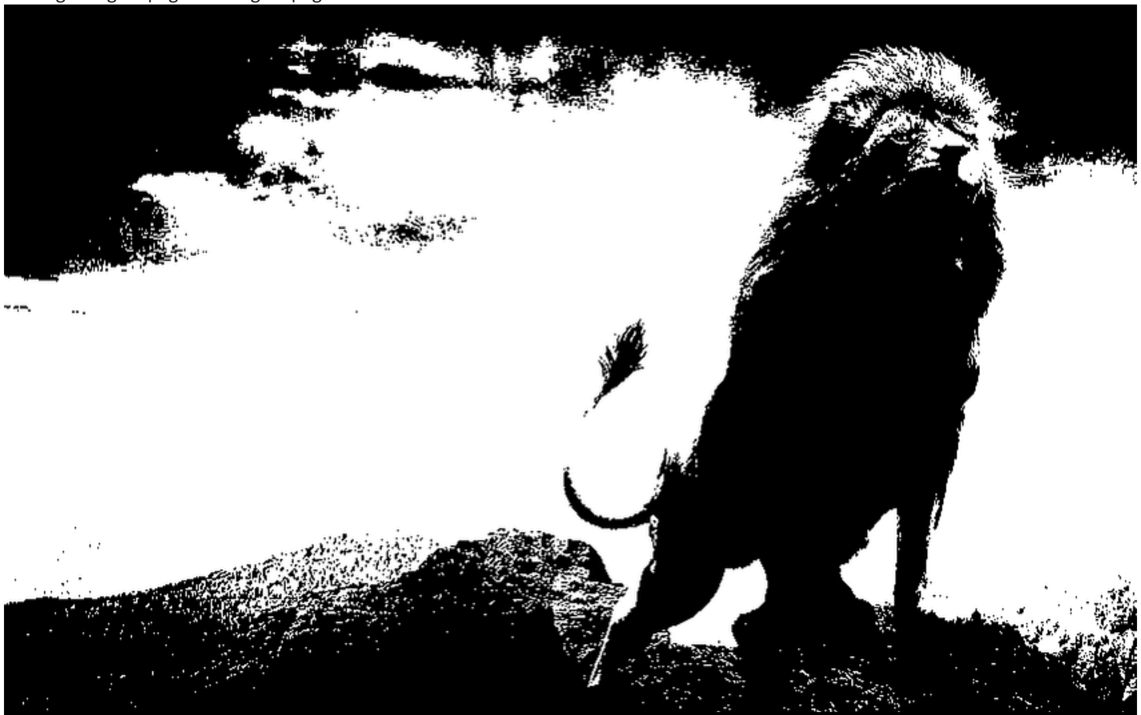
2nd Sobel Edge Detection:

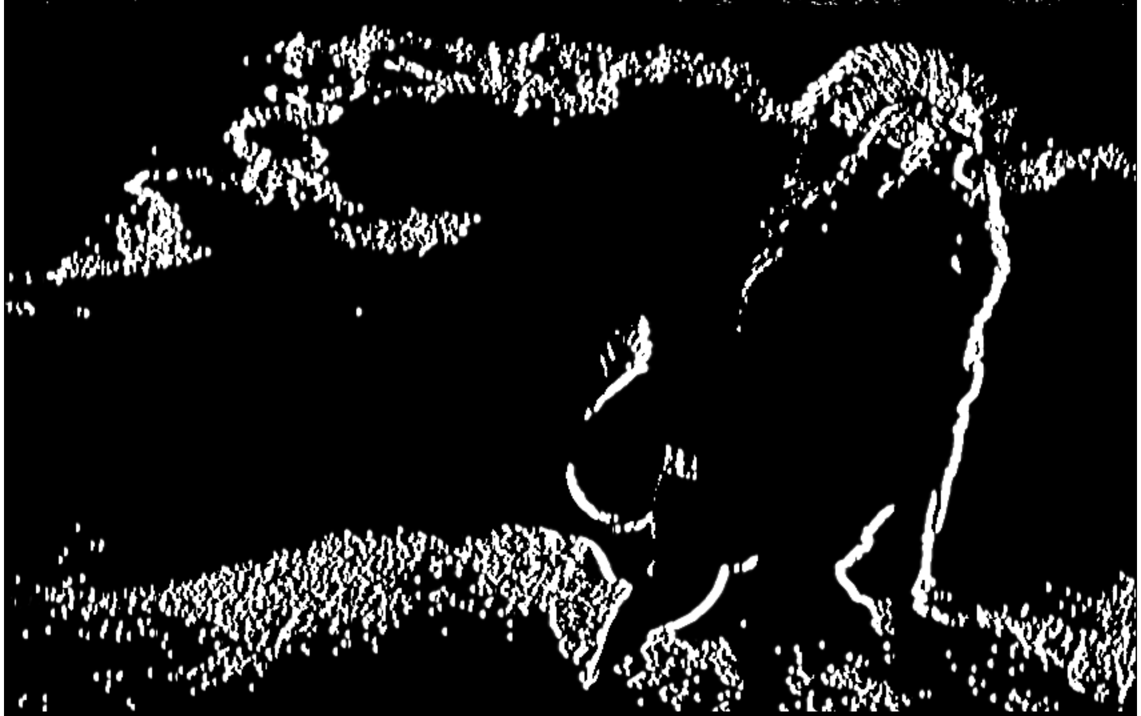
```
from google.colab import files
import cv2
import numpy as np
uploaded = files.upload()
if len(uploaded) > 0:
    file_name = list(uploaded.keys())[0]
    img = cv2.imdecode(np.frombuffer(uploaded[file_name], np.uint8), cv2.IMREAD_COLOR)
    from google.colab.patches import cv2_imshow
    cv2_imshow(img)
    img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    img_blur = cv2.GaussianBlur(img_gray, (3, 3), 0)
    sobelx = cv2.Sobel(src=img_blur, ddepth=cv2.CV_64F, dx=1, dy=0, ksize=5)
    sobely = cv2.Sobel(src=img_blur, ddepth=cv2.CV_64F, dx=0, dy=1, ksize=5)
    sobelxy = cv2.Sobel(src=img_blur, ddepth=cv2.CV_64F, dx=1, dy=1, ksize=5)
    cv2_imshow(sobelx)
    cv2_imshow(sobely)
    cv2_imshow(sobelxy)
    edges = cv2.Canny(img_blur, 50, 150)
    cv2_imshow(edges)
```

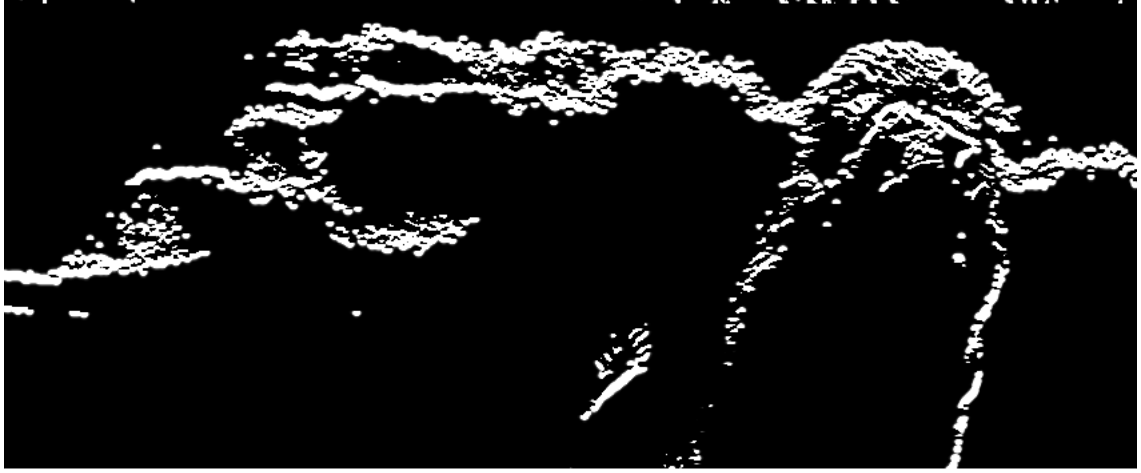
Choose Files Image2.png

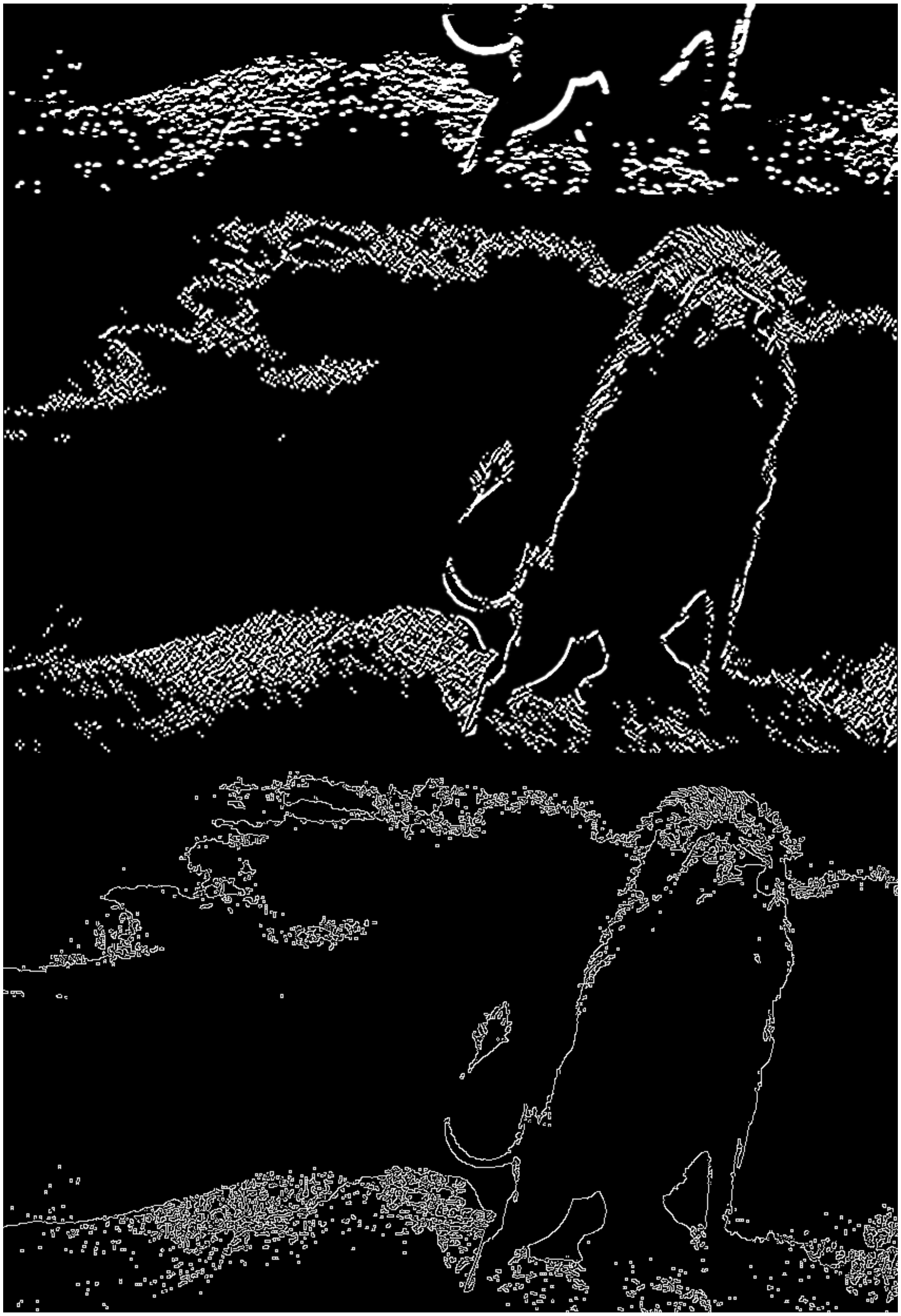
- **Image2.png**(image/png) - 110865 bytes, last modified: 1/22/2024 - 100% done

```
Saving Image2.png to Image2.png
```









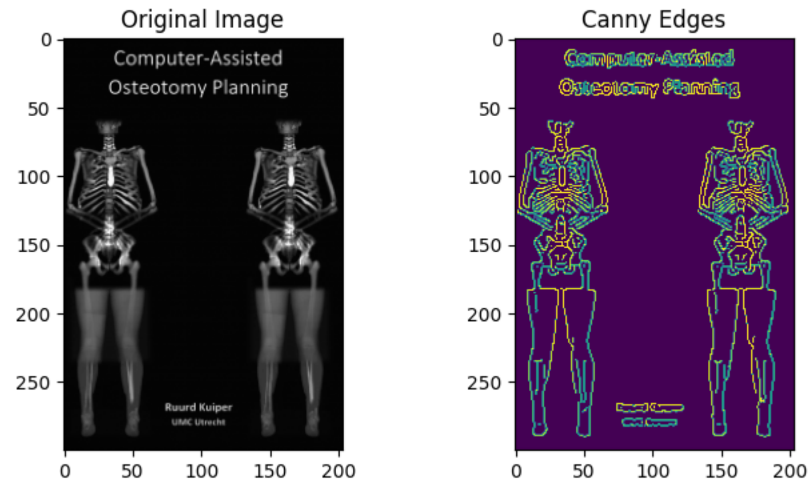
3rd Canny Edge Detection:

```
import matplotlib.pyplot as plt
from skimage import io, feature
from skimage.color import rgb2gray
from PIL import Image
from io import BytesIO
from google.colab import files
import imageio
uploaded = files.upload()
filename = next(iter(uploaded))
image = Image.open(BytesIO(uploaded[filename]))
if image.mode == 'RGBA':
    image = image.convert('RGB')
buffer = BytesIO()
image.save(buffer, format='PNG')
buffer.seek(0)
image_array = imageio.imread(buffer)
if image_array.ndim == 3 and image_array.shape[2] == 3:
    image_array = rgb2gray(image_array)
edges = feature.canny(image_array)
plt.figure(figsize=(8, 4))
plt.subplot(121)
plt.imshow(image_array, cmap='gray')
plt.title('Original Image')
plt.imshow(edges, cmap='viridis')
plt.title('Canny Edges')
plt.show()
```

Choose Files image3.png

- **image3.png**(image/png) - 64907 bytes, last modified: 1/22/2024 - 100% done

```
Saving image3.png to image3.png
<ipython-input-4-96986ec8b4b5>:35: DeprecationWarning: Starting with ImageIO v3 the behavior of this function will switch to that of ii
  image_array = imageio.imread(buffer)
```



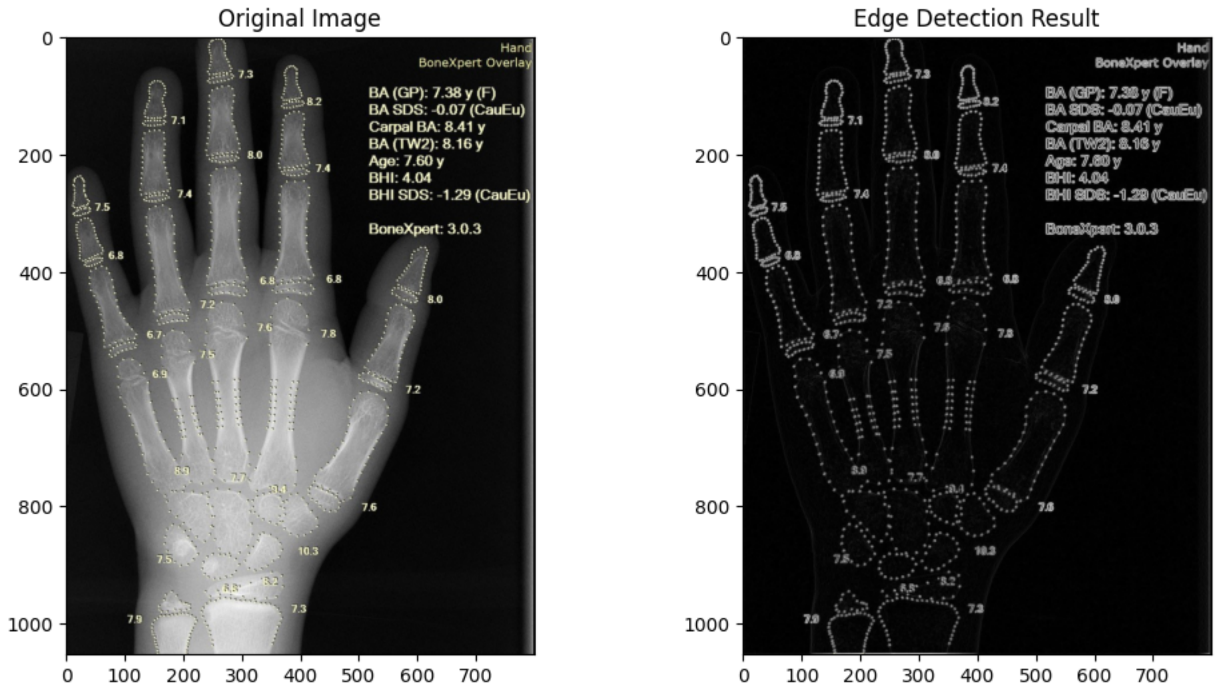
4th Edge Detection using Sobel and Gaussian Filters:

```
import numpy as np
import matplotlib.pyplot as plt
from google.colab import files
uploaded = files.upload()
if len(uploaded) > 0:
    file_name = list(uploaded.keys())[0]
    img = plt.imread(file_name)
    edges_img = img.copy()
    vertical_filter = np.array([[-1, -2, -1], [0, 0, 0], [1, 2, 1]])
    horizontal_filter = np.array([[-1, 0, 1], [-2, 0, 2], [-1, 0, 1]])
    for row in range(3, n-2):
        for col in range(3, m-2):
            local_pixels = img[row-1:row+2, col-1:col+2, 0]
            vertical_transformed_pixels = vertical_filter * local_pixels
            vertical_score = vertical_transformed_pixels.sum() / 4
            horizontal_score = horizontal_transformed_pixels.sum() / 4
            edge_score = (vertical_score**2 + horizontal_score**2)**0.5
            edges_img[row, col] = [edge_score] * 3
    edges_img = edges_img / edges_img.max()
    plt.figure(figsize=(12, 6))
    plt.subplot(1, 2, 1)
    plt.imshow(img)
    plt.title('Original Image')
    plt.subplot(1, 2, 2)
    plt.imshow(edges_img)
    plt.title('Edge Detection Result')
    plt.show()
```

Choose Files image4.jpg

- **image4.jpg**(image/jpeg) - 185755 bytes, last modified: 1/22/2024 - 100% done

```
Saving image4.jpg to image4.jpg
```



5th Gaussian Filter Detection:

```
import skimage
import skimage.io
import skimage.filters
import matplotlib.pyplot as plt
from ipywidgets import interact, FloatSlider
filename = 'bridge.png'
image = skimage.io.imread(fname=filename, as_gray=True)
def filter_function(sigma, threshold):
    masked = image.copy()
    masked[skimage.filters.gaussian(image, sigma=sigma) <= threshold] = 0
    plt.imshow(masked, cmap='gray')
    plt.axis('off')
    plt.show()
interact(filter_function, sigma=FloatSlider(min=0.0, max=7.0, step=0.1, value=1.0),
         threshold=FloatSlider(min=0.0, max=1.0, step=0.01, value=0.5))
```

sigma 2.50

threshold 0.50