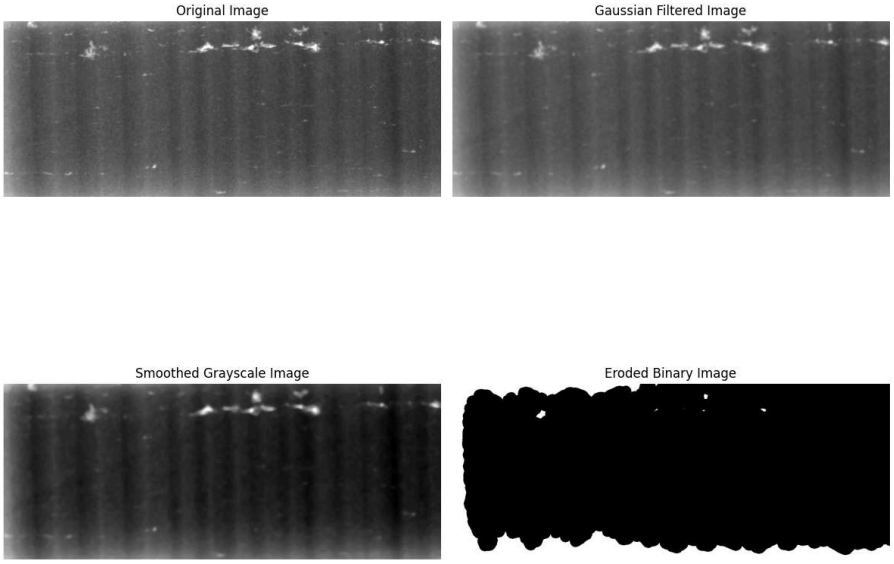
```
import numpy as np
import matplotlib.pyplot as plt
from skimage import io, color, filters
from skimage.morphology import disk, binary_erosion
from google.colab import files

# Upload the image
uploaded = files.upload()

# Assume only one image is uploaded
filename = list(uploaded.keys())[0]

# Read the image
I = io.imread(filename)

# Apply Gaussian filter
Iblur = filters.gaussian(I, sigma=2, multichannel=True)

# Convert to grayscale
gray = color.rgb2gray(Iblur)

# Smooth out the image to remove irregular patches
gray_smooth = filters.median(gray, selem=np.ones((10, 10)))

# Binarize the smoothed image
threshold = filters.threshold_otsu(gray_smooth)
BW = gray_smooth > threshold

# Erode the binary image to remove patches
BW_eroded = binary_erosion(BW, disk(10))

# Display the original and processed images
fig, axes = plt.subplots(2, 2, figsize=(12, 12))
axes[0, 0].imshow(I)
axes[0, 0].set_title('Original Image')
axes[0, 0].axis('off')
axes[0, 1].imshow(Iblur)
axes[0, 1].set_title('Gaussian Filtered Image')
axes[0, 1].axis('off')
axes[1, 0].imshow(gray_smooth, cmap='gray')
axes[1, 0].set_title('Smoothed Grayscale Image')
axes[1, 0].axis('off')
axes[1, 1].imshow(BW_eroded, cmap='gray')
axes[1, 1].set_title('Eroded Binary Image')
axes[1, 1].axis('off')
plt.tight_layout()
plt.show()
```

Choose Files testImage1.jpg

- **testImage1.jpg**(image/jpeg) - 96766 bytes, last modified: 5/10/2024 - 100% done

```
Saving testImage1.jpg to testImage1.jpg
<ipython-input-1-0ad534d862f3>:17: FutureWarning: `multichannel` is a deprecated argumer
  Iblur = filters.gaussian(I, sigma=2, multichannel=True)
<ipython-input-1-0ad534d862f3>:23: FutureWarning: `selem` is a deprecated argument name
  gray_smooth = filters.median(gray, selem=np.ones((10, 10)))
```



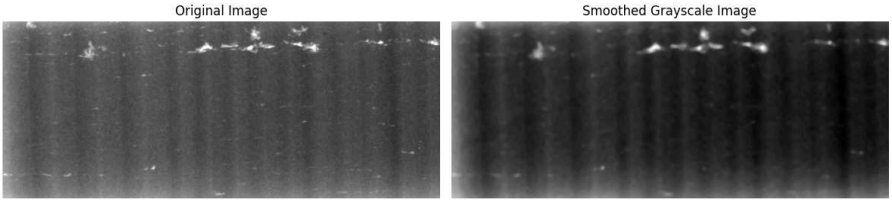

```
import numpy as np
import matplotlib.pyplot as plt
from skimage import io, color, filters
from google.colab import files

# Upload the image
uploaded = files.upload()

# Assume only one image is uploaded
filename = list(uploaded.keys())[0]

# Read the image
I = io.imread(filename)

# Apply Gaussian filter
Iblur = filters.gaussian(I, sigma=2, multichannel=True)

# Convert to grayscale
gray = color.rgb2gray(Iblur)

# Smooth out the image to remove irregular patches
gray_smooth = filters.median(gray, selem=np.ones((10, 10)))

# Display the original and processed images
fig, axes = plt.subplots(1, 2, figsize=(12, 6))
axes[0].imshow(I)
axes[0].set_title('Original Image')
axes[0].axis('off')
axes[1].imshow(gray_smooth, cmap='gray')
```

```
axes[1].set_title('Smoothed Grayscale Image')
axes[1].axis('off')
plt.tight_layout()
plt.show()
```

Choose Files testImage1.jpg

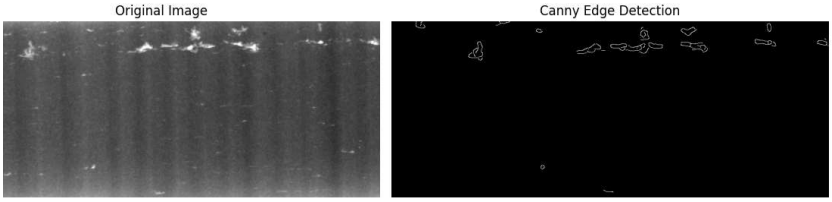
- **testImage1.jpg**(image/jpeg) - 96766 bytes, last modified: 5/10/2024 - 100% done

```
Saving testImage1.jpg to testImage1 (1).jpg
<ipython-input-2-9629882cec4c>:16: FutureWarning: `multichannel` is a deprecated argumer
  Iblur = filters.gaussian(I, sigma=2, multichannel=True)
<ipython-input-2-9629882cec4c>:22: FutureWarning: `selem` is a deprecated argument name
  gray_smooth = filters.median(gray, selem=np.ones((10, 10)))
```




```
import numpy as np
import matplotlib.pyplot as plt
from skimage import io, color, filters, feature
from google.colab import files

# Upload the image
uploaded = files.upload()

# Assume only one image is uploaded
filename = list(uploaded.keys())[0]

# Read the image
I = io.imread(filename)

# Apply Gaussian filter
Iblur = filters.gaussian(I, sigma=2, multichannel=True)

# Convert to grayscale
gray = color.rgb2gray(Iblur)

# Smooth out the image to remove irregular patches
gray_smooth = filters.median(gray, selem=np.ones((10, 10)))

# Perform Canny edge detection
edges = feature.canny(gray_smooth, sigma=1)

# Display the original and processed images
fig, axes = plt.subplots(1, 2, figsize=(12, 6))
axes[0].imshow(I)
axes[0].set_title('Original Image')
axes[0].axis('off')
axes[1].imshow(edges, cmap='gray')
axes[1].set_title('Canny Edge Detection')
axes[1].axis('off')
plt.tight_layout()
plt.show()
```

```
Files testImage1.jpg
nage1.jpg(image/jpeg) - 96766 bytes, last modified: 5/10/2024 - 100% done
 testImage1.jpg to testImage1 (2).jpg
on-input-3-547bac9ffb00>:16: FutureWarning: `multichannel` is a deprecated argument name
 = filters.gaussian(I, sigma=2, multichannel=True)
on-input-3-547bac9ffb00>:22: FutureWarning: `selem` is a deprecated argument name for `me
_smooth = filters.median(gray, selem=np.ones((10, 10)))
```

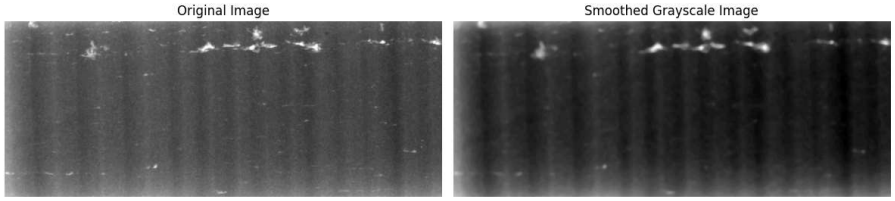


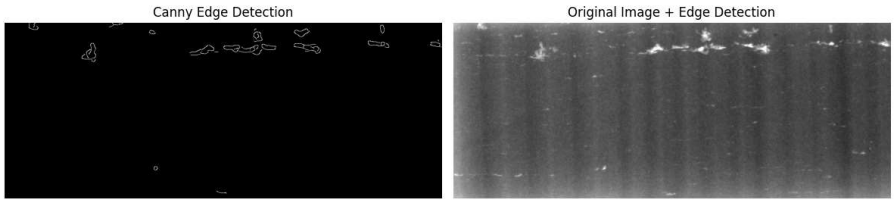

```
import numpy as np
import matplotlib.pyplot as plt
from skimage import io, color, filters, feature
from google.colab import files

# Upload the image
uploaded = files.upload()

# Assume only one image is uploaded
filename = list(uploaded.keys())[0]

# Read the image
I = io.imread(filename)

# Apply Gaussian filter
Iblur = filters.gaussian(I, sigma=2, multichannel=True)

# Convert to grayscale
gray = color.rgb2gray(Iblur)

# Smooth out the image to remove irregular patches
gray_smooth = filters.median(gray, selem=np.ones((10, 10)))

# Perform Canny edge detection
edges = feature.canny(gray_smooth, sigma=1)

# Create a composite image with original image overlaid on edge detection result
composite_image = np.zeros_like(I)
composite_image[:,:,0] = np.maximum(I[:,:,0], edges)
composite_image[:,:,1] = np.maximum(I[:,:,1], edges)
composite_image[:,:,2] = np.maximum(I[:,:,2], edges)

# Display the original, smoothed grayscale, edge detection, and composite images
fig, axes = plt.subplots(2, 2, figsize=(12, 12))
axes[0, 0].imshow(I)
axes[0, 0].set_title('Original Image')
axes[0, 0].axis('off')
axes[0, 1].imshow(gray_smooth, cmap='gray')
axes[0, 1].set_title('Smoothed Grayscale Image')
axes[0, 1].axis('off')
axes[1, 0].imshow(edges, cmap='gray')
axes[1, 0].set_title('Canny Edge Detection')
axes[1, 0].axis('off')
axes[1, 1].imshow(composite_image)
axes[1, 1].set_title('Original Image + Edge Detection')
axes[1, 1].axis('off')
plt.tight_layout()
plt.show()
```

Choose Files testImage1.jpg

- **testImage1.jpg**(image/jpeg) - 96766 bytes, last modified: 5/10/2024 - 100% done

```
Saving testImage1.jpg to testImage1 (3).jpg
<ipython-input-4-e7c9371fc07c>:16: FutureWarning: `multichannel` is a deprecated argumer
  Iblur = filters.gaussian(I, sigma=2, multichannel=True)
<ipython-input-4-e7c9371fc07c>:22: FutureWarning: `selem` is a deprecated argument name
  gray_smooth = filters.median(gray, selem=np.ones((10, 10)))
```






```
import numpy as np
import matplotlib.pyplot as plt
from skimage import io, color, filters, feature, morphology
from google.colab import files

# Upload the image
uploaded = files.upload()

# Assume only one image is uploaded
filename = list(uploaded.keys())[0]

# Read the image
I = io.imread(filename)

# Apply Gaussian filter
Iblur = filters.gaussian(I, sigma=2, multichannel=True)

# Convert to grayscale
gray = color.rgb2gray(Iblur)

# Smooth out the image to remove irregular patches
gray_smooth = filters.median(gray, selem=np.ones((10, 10)))

# Perform Canny edge detection
edges = feature.canny(gray_smooth, sigma=1)

# Dilate the edges to capture smaller white spots
edges_dilated = morphology.binary_dilation(edges, morphology.disk(2))
```

```
# Display the original and processed images
fig, axes = plt.subplots(1, 2, figsize=(12, 6))
axes[0].imshow(I)
axes[0].set_title('Original Image')
axes[0].axis('off')
axes[1].imshow(edges_dilated, cmap='gray')
axes[1].set_title('Canny Edge Detection with Dilation')
axes[1].axis('off')
plt.tight_layout()
plt.show()
```

Choose Files testImage1.jpg

- **testImage1.jpg**(image/jpeg) - 96766 bytes, last modified: 5/10/2024 - 100% done

```
Saving testImage1.jpg to testImage1 (4).jpg
<ipython-input-5-0e38fa0b38ed>:16: FutureWarning: `multichannel` is a deprecated argument name for `gaussian`. It will be removed in ver
```