```python
from google.colab import drive
drive.mount('/content/drive')
```

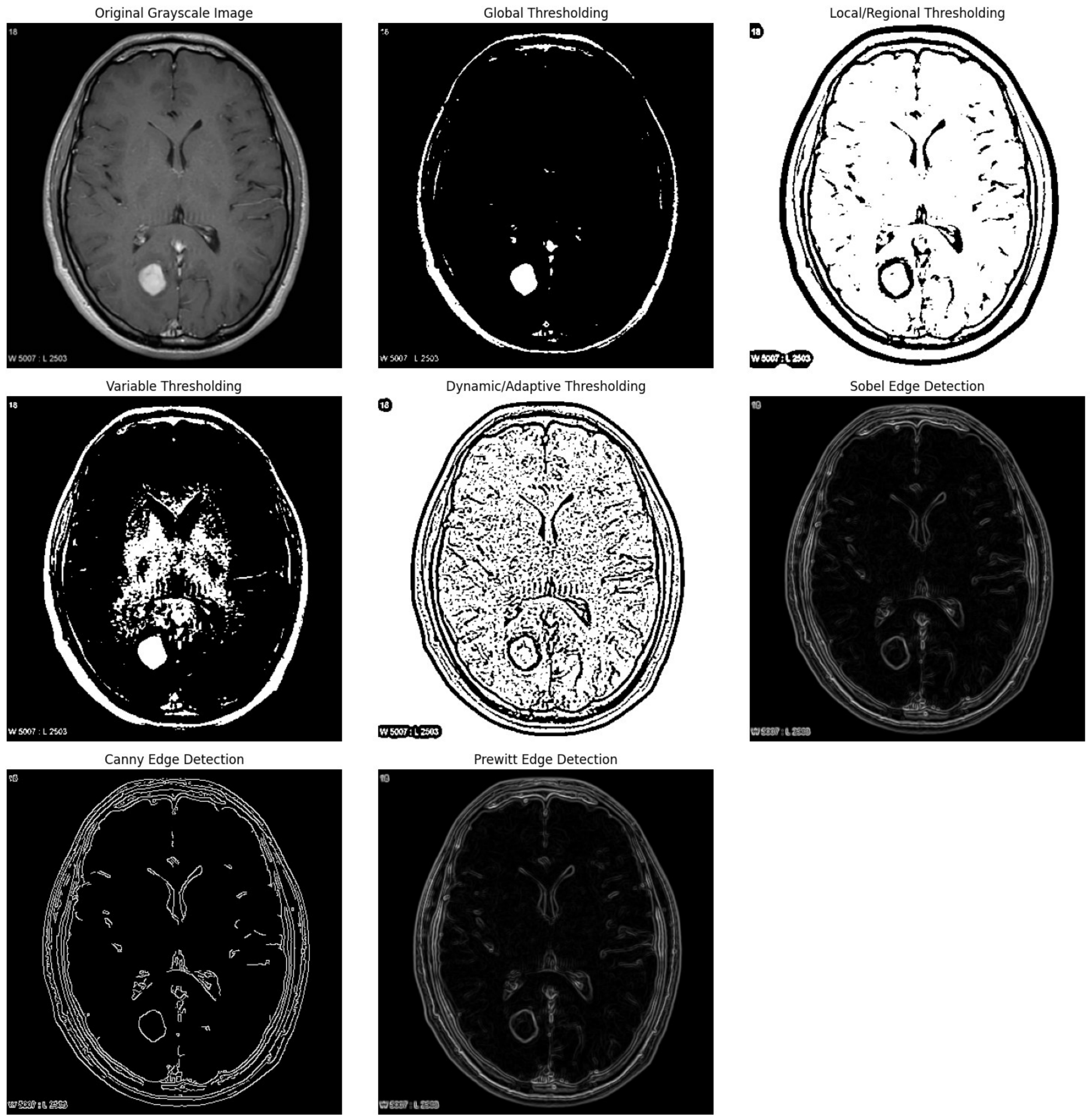
```
Mounted at /content/drive
```

```python
# Install necessary libraries (only once)
!pip install scikit-image

# Import required libraries
import cv2
import numpy as np
import matplotlib.pyplot as plt
from skimage.filters import threshold_local
from skimage import io, color, filters, feature

# Read the image
img = cv2.imread('Picture1.png')
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)  # Convert BGR to RGB
gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)  # Convert to
grayscale

# Create a figure for side-by-side display
plt.figure(figsize=(15, 15))

# 1. Original Grayscale Image
plt.subplot(3, 3, 1)
plt.imshow(gray_img, cmap='gray')
plt.title('Original Grayscale Image')
plt.axis('off')

# 2. Global Thresholding
_, global_thresh = cv2.threshold(gray_img, 120, 255,
cv2.THRESH_BINARY)
plt.subplot(3, 3, 2)
plt.imshow(global_thresh, cmap='gray')
plt.title('Global Thresholding')
plt.axis('off')

# 3. Local/Regional Thresholding
block_size = 35
local_thresh = threshold_local(gray_img, block_size, offset=10)
binary_local = gray_img > local_thresh
plt.subplot(3, 3, 3)
plt.imshow(binary_local, cmap='gray')
plt.title('Local/Regional Thresholding')
plt.axis('off')

# 4. Variable Thresholding
mean_val = np.mean(gray_img)
```

```python
std_val = np.std(gray_img)
variable_thresh = gray_img > (mean_val + std_val)
plt.subplot(3, 3, 4)
plt.imshow(variable_thresh, cmap='gray')
plt.title('Variable Thresholding')
plt.axis('off')

# 5. Dynamic/Adaptive Thresholding
adaptive_thresh = cv2.adaptiveThreshold(
    gray_img, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
    cv2.THRESH_BINARY, 11, 2)
plt.subplot(3, 3, 5)
plt.imshow(adaptive_thresh, cmap='gray')
plt.title('Dynamic/Adaptive Thresholding')
plt.axis('off')

# 6. Sobel Edge Detection
sobel_edges = filters.sobel(gray_img)
plt.subplot(3, 3, 6)
plt.imshow(sobel_edges, cmap='gray')
plt.title('Sobel Edge Detection')
plt.axis('off')

# 7. Canny Edge Detection
edges_canny = cv2.Canny(gray_img, 100, 200)
plt.subplot(3, 3, 7)
plt.imshow(edges_canny, cmap='gray')
plt.title('Canny Edge Detection')
plt.axis('off')

# 8. Prewitt Edge Detection
prewitt_edges = filters.prewitt(gray_img)
plt.subplot(3, 3, 8)
plt.imshow(prewitt_edges, cmap='gray')
plt.title('Prewitt Edge Detection')
plt.axis('off')

# Show all plots
plt.tight_layout()
plt.show()
```

```
Requirement already satisfied: scikit-image in
/usr/local/lib/python3.12/dist-packages (0.25.2)
Requirement already satisfied: numpy>=1.24 in
/usr/local/lib/python3.12/dist-packages (from scikit-image) (2.0.2)
Requirement already satisfied: scipy>=1.11.4 in
/usr/local/lib/python3.12/dist-packages (from scikit-image) (1.16.1)
Requirement already satisfied: networkx>=3.0 in
/usr/local/lib/python3.12/dist-packages (from scikit-image) (3.5)
Requirement already satisfied: pillow>=10.1 in
```

```
/usr/local/lib/python3.12/dist-packages (from scikit-image) (11.3.0)
Requirement already satisfied: imageio!=2.35.0,>=2.33 in
/usr/local/lib/python3.12/dist-packages (from scikit-image) (2.37.0)
Requirement already satisfied: tifffile>=2022.8.12 in
/usr/local/lib/python3.12/dist-packages (from scikit-image)
(2025.6.11)
Requirement already satisfied: packaging>=21 in
/usr/local/lib/python3.12/dist-packages (from scikit-image) (25.0)
Requirement already satisfied: lazy-loader>=0.4 in
/usr/local/lib/python3.12/dist-packages (from scikit-image) (0.4)
```

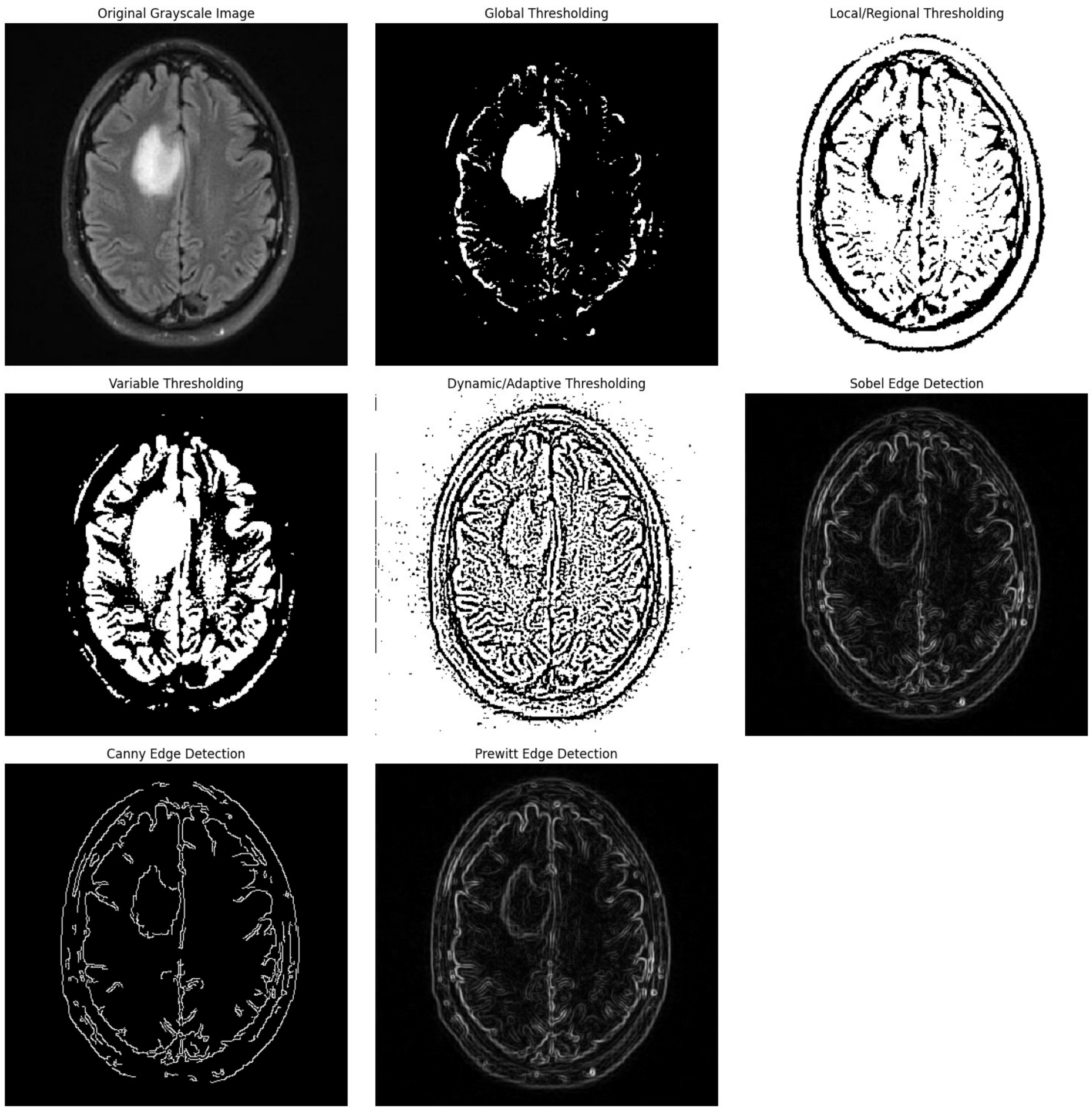


```python
# Install necessary libraries (only once)
!pip install scikit-image

# Import required libraries
import cv2
import numpy as np
import matplotlib.pyplot as plt
from skimage.filters import threshold_local
from skimage import io, color, filters, feature

# Read the image
img = cv2.imread('Picture2.png')
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)  # Convert BGR to RGB
gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)  # Convert to
grayscale

# Create a figure for side-by-side display
plt.figure(figsize=(15, 15))

# 1. Original Grayscale Image
plt.subplot(3, 3, 1)
plt.imshow(gray_img, cmap='gray')
plt.title('Original Grayscale Image')
plt.axis('off')

# 2. Global Thresholding
_, global_thresh = cv2.threshold(gray_img, 120, 255,
cv2.THRESH_BINARY)
plt.subplot(3, 3, 2)
plt.imshow(global_thresh, cmap='gray')
plt.title('Global Thresholding')
plt.axis('off')

# 3. Local/Regional Thresholding
block_size = 35
local_thresh = threshold_local(gray_img, block_size, offset=10)
binary_local = gray_img > local_thresh
plt.subplot(3, 3, 3)
plt.imshow(binary_local, cmap='gray')
plt.title('Local/Regional Thresholding')
plt.axis('off')

# 4. Variable Thresholding
mean_val = np.mean(gray_img)
std_val = np.std(gray_img)
variable_thresh = gray_img > (mean_val + std_val)
plt.subplot(3, 3, 4)
plt.imshow(variable_thresh, cmap='gray')
plt.title('Variable Thresholding')
plt.axis('off')
```

```
# 5. Dynamic/Adaptive Thresholding
adaptive_thresh = cv2.adaptiveThreshold(
    gray_img, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
    cv2.THRESH_BINARY, 11, 2)
plt.subplot(3, 3, 5)
plt.imshow(adaptive_thresh, cmap='gray')
plt.title('Dynamic/Adaptive Thresholding')
plt.axis('off')

# 6. Sobel Edge Detection
sobel_edges = filters.sobel(gray_img)
plt.subplot(3, 3, 6)
plt.imshow(sobel_edges, cmap='gray')
plt.title('Sobel Edge Detection')
plt.axis('off')

# 7. Canny Edge Detection
edges_canny = cv2.Canny(gray_img, 100, 200)
plt.subplot(3, 3, 7)
plt.imshow(edges_canny, cmap='gray')
plt.title('Canny Edge Detection')
plt.axis('off')

# 8. Prewitt Edge Detection
prewitt_edges = filters.prewitt(gray_img)
plt.subplot(3, 3, 8)
plt.imshow(prewitt_edges, cmap='gray')
plt.title('Prewitt Edge Detection')
plt.axis('off')

# Show all plots
plt.tight_layout()
plt.show()
```

```
Requirement already satisfied: scikit-image in
/usr/local/lib/python3.12/dist-packages (0.25.2)
Requirement already satisfied: numpy>=1.24 in
/usr/local/lib/python3.12/dist-packages (from scikit-image) (2.0.2)
Requirement already satisfied: scipy>=1.11.4 in
/usr/local/lib/python3.12/dist-packages (from scikit-image) (1.16.1)
Requirement already satisfied: networkx>=3.0 in
/usr/local/lib/python3.12/dist-packages (from scikit-image) (3.5)
Requirement already satisfied: pillow>=10.1 in
/usr/local/lib/python3.12/dist-packages (from scikit-image) (11.3.0)
Requirement already satisfied: imageio!=2.35.0,>=2.33 in
/usr/local/lib/python3.12/dist-packages (from scikit-image) (2.37.0)
Requirement already satisfied: tifffile>=2022.8.12 in
/usr/local/lib/python3.12/dist-packages (from scikit-image)
(2025.6.11)
```

```
Requirement already satisfied: packaging>=21 in
/usr/local/lib/python3.12/dist-packages (from scikit-image) (25.0)
Requirement already satisfied: lazy-loader>=0.4 in
/usr/local/lib/python3.12/dist-packages (from scikit-image) (0.4)
```



```
display(df)
```

{"summary":"{\n \"name\": \"df\",\n \"rows\": 8,\n \"fields\": [\n
{\n   \"column\": \"Feature/Aspect\",\n   \"properties\": {\n
\"dtype\": \"string\",\n   \"num_unique_values\": 8,\n
\"samples\": [\n   \"Use Case\",\n   \"Sensitivity\",\n

```
\"Principle\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Similarity Technique\",\n      \"properties\": {\n        \"dtype\":
\"string\",\n        \"num_unique_values\": 8,\n        \"samples\":
[\n          \"Grouping or segmenting areas with uniform
properties\",\n          \"Sensitive to noise within regions, may
merge different tissues\",\n          \"Detects regions with similar
intensity or texture\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Discontinuity Technique\",\n      \"properties\": {\n
\"dtype\": \"string\",\n        \"num_unique_values\": 8,\n
\"samples\": [\n          \"Identifying edges, boundaries, and
contours\",\n          \"Sensitive to noise, but focuses on strong
gradients\",\n          \"Detects abrupt changes in intensity or
gradient\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"df"}
```

Tumor 1 Image Analysis:

Image ID / Description	Segmentation Method	Boundary Clarity (✔/✘)	Tumor Fully Captured? (✔/✘)	Noise Sensitivity (Low / Medium / High)	Comments / Observations
Tumor 1	Global Threshold (127)	✔	✔	Low	Tumor clearly segmented with strong contrast.
	Adaptive Mean (11)	✘	✘	Medium	Boundary not clear, partial tumor capture; some noise.
	Adaptive Gaussian (11)	✔	✔	Medium	Better capture than Adaptive Mean; decent boundary clarity.
	Otsu's Threshold	✔	✔	Low	Tumor well segmented, clear boundary, low noise.
	Sobel (ksize=3)	✘	✘	High	Edges noisy, tumor boundary fragmented.
	Prewitt	✘	✘	High	Noisy edges, poor tumor detection.
	Canny (100–200)	✔	✔	Medium	Edges clear, tumor fully captured, some noise present.

Tumor 2 Image Analysis:

Image ID / Description	**Segmentation Method**	**Boundary Clarity (✔/✘)**	**Tumor Fully Captured? (✔/✘)**	**Noise Sensitivity (Low / Medium / High)**	**Comments / Observations**
IMG-1 (e.g., high contrast tumor)	Global Threshold (127)	✔	✔	Low	Tumor and brain boundary clearly visible; minimal noise.
	Adaptive Mean (11)	✘	✘	Medium	Boundary unclear, some parts missing; noise affects results.
	Adaptive Gaussian (11)	✔	✔	Medium	Better than adaptive mean; boundary mostly clear.
	Otsu's Threshold	✔	✔	Low	Clear boundary and full tumor capture, minimal noise.
	Sobel (ksize=3)	✘	✘	High	Too much noise, boundaries unclear.
	Prewitt	✘	✘	High	Noisy edges, poor tumor capture.
	Canny (100–200)	✔	✔	Medium	Edges well-defined but some noise present.