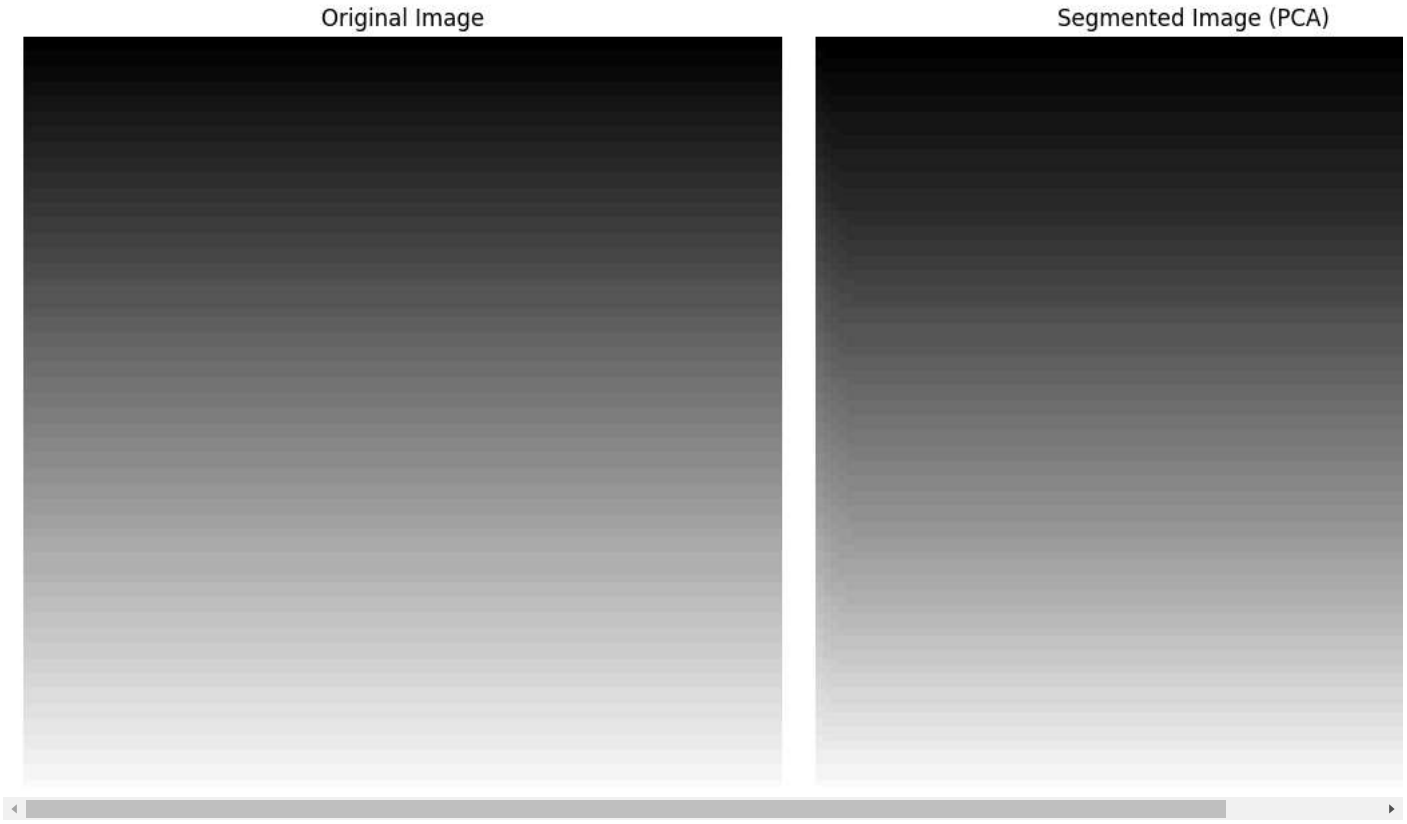
```
import numpy as np
import matplotlib.pyplot as plt
import cv2

# Function to create a simple gradient image for testing purposes
def create_gradient_image(size=(100, 100)):
    """Creates a simple gradient image for testing (values 0 to 255)."""
    return np.linspace(0, 255, size[0], dtype=np.uint8).reshape(-1, 1) * np.ones(size, dtype=np.uint8)

# Function to apply a simple "cellular automaton" update rule to the image
def cellular_automaton(image, max_iterations=30):
    # Copy the image to prevent modifying the original
    segmented_image = image.copy()

    for iteration in range(max_iterations):
        # Update the image based on neighboring pixel values
        new_image = segmented_image.copy()

        # Iterate over every pixel in the image
        for x in range(1, image.shape[0] - 1):  # Ignore borders
            for y in range(1, image.shape[1] - 1):  # Ignore borders
                # Take the average value of the neighbors (4-connected neighborhood)
                neighbors = [
                    segmented_image[x-1, y],  # Top
                    segmented_image[x+1, y],  # Bottom
                    segmented_image[x, y-1],  # Left
                    segmented_image[x, y+1]   # Right
                ]
                # The pixel updates to the average of its neighbors
                new_image[x, y] = int(np.mean(neighbors))

        # Update the image for the next iteration
        segmented_image = new_image

        # Optionally print progress every 5 iterations
        if iteration % 5 == 0:
            print(f"Iteration {iteration}/{max_iterations}")

    return segmented_image

# Load an image (or use a generated gradient image for testing)
# For testing, let's create a simple gradient image
image = create_gradient_image(size=(100, 100))

# Optionally, you can load your own image using OpenCV:
# image = cv2.imread('your_image_path.jpg', cv2.IMREAD_GRAYSCALE)

# Apply the Parallel Cellular Algorithm to segment the image
segmented_image = cellular_automaton(image, max_iterations=30)

# Plot the original and segmented images side by side
plt.figure(figsize=(12, 6))

# Plot the original image
plt.subplot(1, 2, 1)
plt.imshow(image, cmap='gray')
plt.title("Original Image")
plt.axis('off')

# Plot the segmented image
plt.subplot(1, 2, 2)
plt.imshow(segmented_image, cmap='gray')
plt.title("Segmented Image (PCA)")
plt.axis('off')

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

```
Iteration 0/30
Iteration 5/30
Iteration 10/30
Iteration 15/30
Iteration 20/30
Iteration 25/30
```



Start coding or generate with AI.