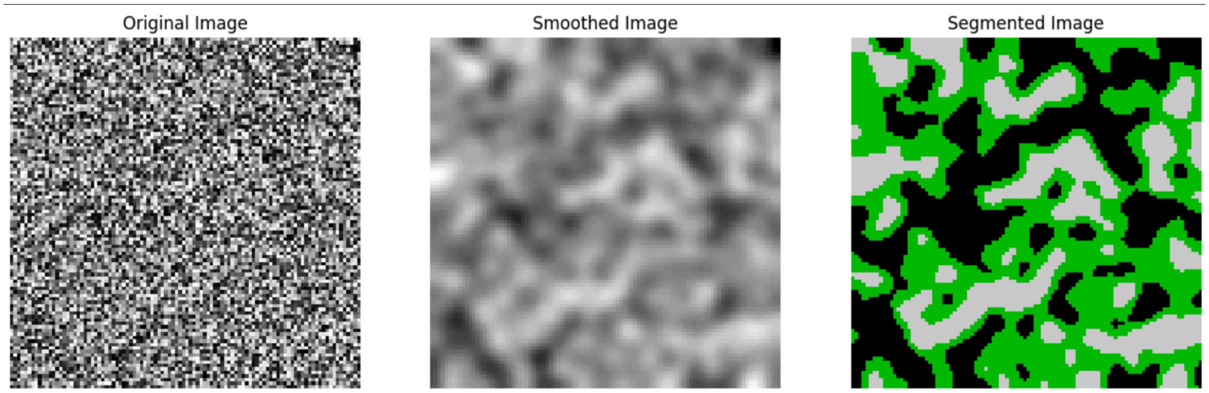
Code:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
def initialize_image(height, width):
    np.random.seed(42)  # For reproducibility
    return np.random.randint(0, 256, (height, width), dtype=np.uint8)

def segment_image(image, n_segments):
    flat_image = image.flatten().reshape(-1, 1)
    kmeans = KMeans(n_clusters=n_segments, random_state=42)
    kmeans.fit(flat_image)
    segmented_flat = kmeans.labels_
    segmented_image = segmented_flat.reshape(image.shape)
    return segmented_image

def parallel_cellular_algorithm(image, iterations):
    height, width = image.shape
    grid = image.copy()

    for iteration in range(iterations):
        new_grid = grid.copy()
        for x in range(height):
            for y in range(width):
                neighbors = get_neighbors(grid, x, y)
                new_grid[x, y] = evaluate_fitness(grid[x, y], neighbors)
        grid = new_grid

    return grid

def evaluate_fitness(cell, neighbors):
    return np.mean(neighbors)

def get_neighbors(grid, x, y):
    neighbors = []
    for dx in [-1, 0, 1]:
        for dy in [-1, 0, 1]:
            if dx == 0 and dy == 0:
                continue
            nx, ny = x + dx, y + dy
            if 0 <= nx < grid.shape[0] and 0 <= ny < grid.shape[1]:
                neighbors.append(grid[nx, ny])
    return neighbors

height, width = 100, 100  # Dimensions of the dummy image
iterations = 10  # Number of iterations for the algorithm
n_segments = 3  # Number of segments for image segmentation

image = initialize_image(height, width)

smoothed_image = parallel_cellular_algorithm(image, iterations)

segmented_image = segment_image(smoothed_image, n_segments)

plt.figure(figsize=(15, 5))
plt.subplot(1, 3, 1)
plt.title("Original Image")
plt.imshow(image, cmap='gray')
plt.axis('off')

plt.subplot(1, 3, 2)
plt.title("Smoothed Image")
plt.imshow(smoothed_image, cmap='gray')
plt.axis('off')
plt.subplot(1, 3, 3)
plt.title("Segmented Image")
plt.imshow(segmented_image, cmap='nipy_spectral')
plt.axis('off')

plt.show()
```

Output:


Original Image
Smoothed Image
Segmented Image