Aim:
To write a program for region based segmentation.

Algorithm:

1.Import necessary libraries such as numpy, matplotlib.pyplot, skimage, and scipy.ndimage.

2.Load and Pre-process Image:

Load the sample rocket image using data.rocket(). Convert the RGB image to grayscale using rgb2gray().

3.Edge Detection:

Apply the Canny edge detection algorithm to the grayscale image using canny().

4.Region Filling:

Fill the regions enclosed by edges using binary_fill_holes() from scipy.ndimage.

5.Elevation Map Computation:

Compute the elevation map of the grayscale image using the Sobel filter with filters.sobel().

6.Marker Creation for Watershed Segmentation:

Create markers for the watershed algorithm. Markers are created based on pixel intensity values:

Mark pixels with intensity less than 30/255 as 1.

Mark pixels with intensity greater than 150/255 as 2.

7.Watershed Segmentation:

Perform watershed segmentation using the elevation map and the created markers with segmentation.watershed().

8.Hole Filling in Segmented Image:

Fill holes in the segmented image using binary_fill_holes() from scipy.ndimage. Label the filled regions using nd.label().

9.Overlay Segmented Labels on Original Image:

Overlay the labeled segments on the original grayscale image using label2rgb().

10.Visualization:

Create a figure with two subplots. The first subplot shows the original grayscale image. The second subplot shows the original image with labeled regions in different colors.

Program Code:

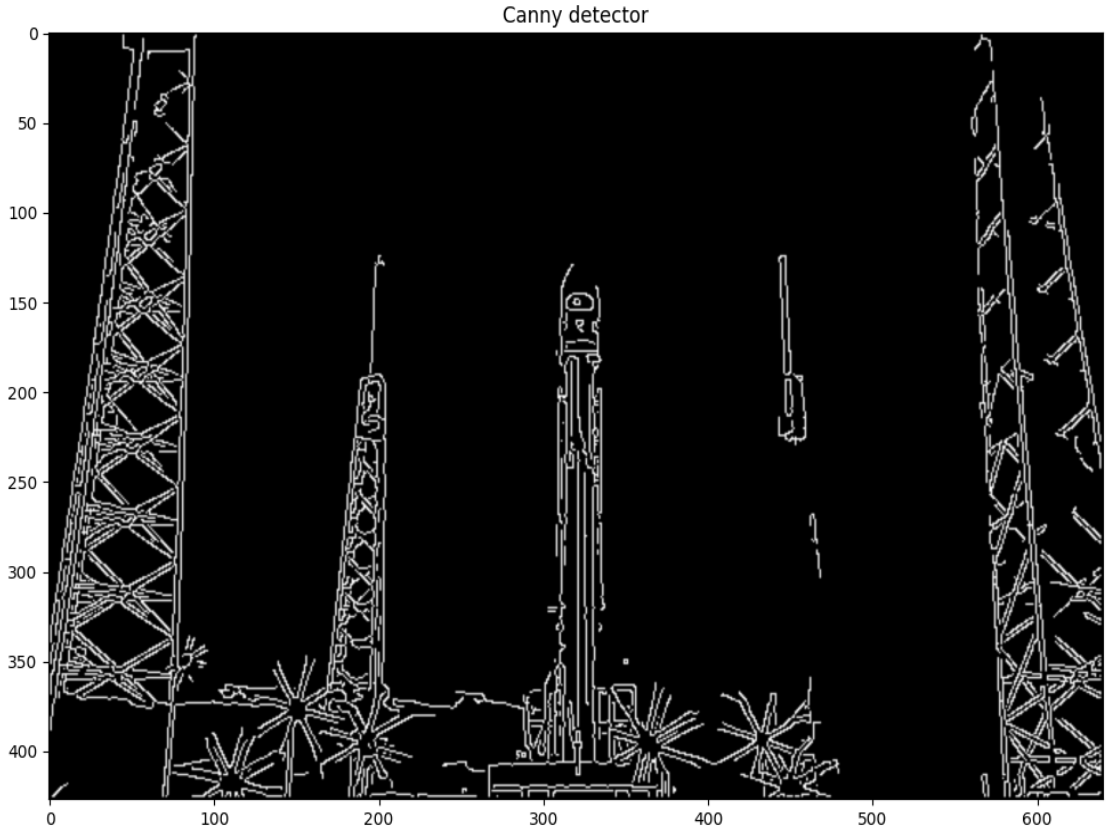
```
import numpy as np
import matplotlib.pyplot as plt
from skimage import data, segmentation, color, filters
from skimage.color import rgb2gray
from skimage.feature import canny
import scipy.ndimage as nd

plt.rcParams["figure.figsize"] = (12, 8)

# Load the rocket image and convert to grayscale
rocket = data.rocket()
rocket_gray = rgb2gray(rocket)

# Plot the original image
plt.figure()
plt.imshow(rocket_gray, cmap='gray')
plt.title('Original Image')
plt.axis('off')
plt.show()

# Canny edge detector
edges = canny(rocket_gray)
plt.figure()
```

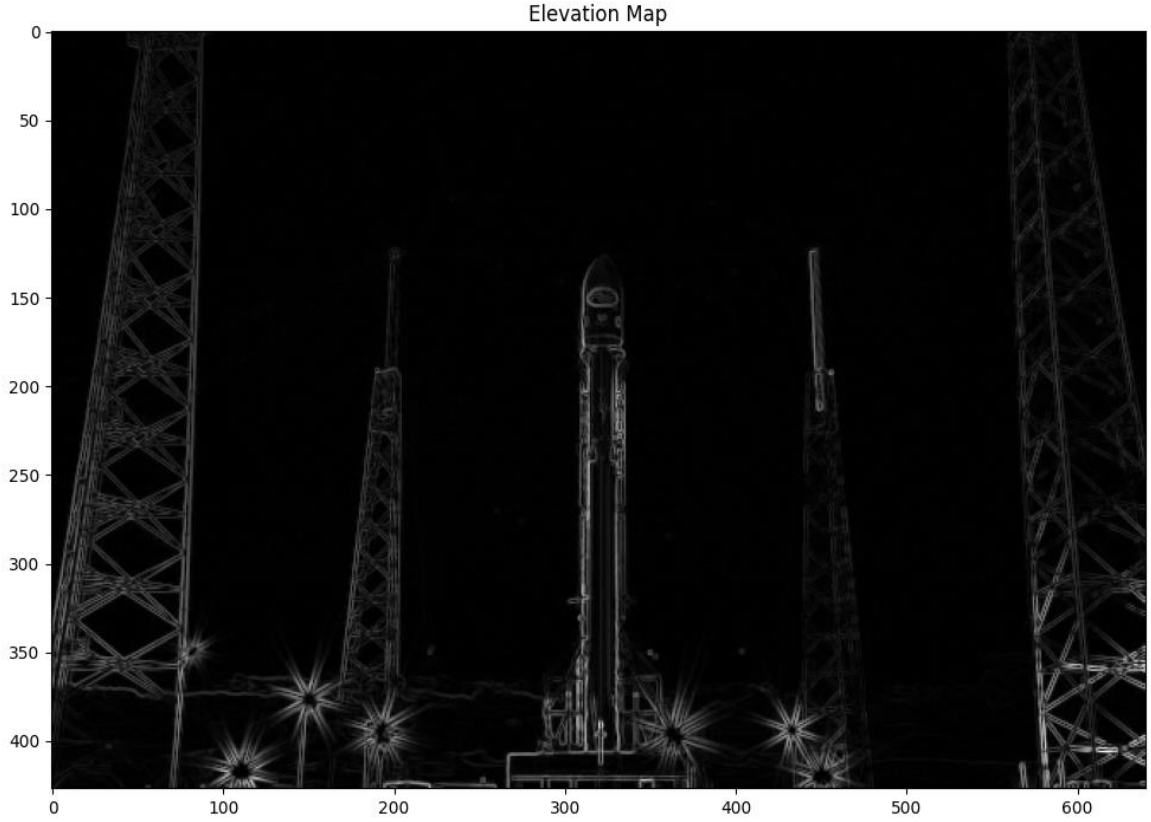
```
plt.imshow(edges, cmap='gray', interpolation='gaussian')
plt.title('Canny Detector')
plt.axis('off')
plt.show()

# Fill holes in the edges
fill_im = nd.binary_fill_holes(edges)
plt.figure()
plt.imshow(fill_im, cmap='gray')
plt.title('Region Filling')
plt.axis('off')
plt.show()

# Create an elevation map using the Sobel filter
elevation_map = filters.sobel(rocket_gray)
plt.figure()
plt.imshow(elevation_map, cmap='gray')
plt.title('Elevation Map')
plt.axis('off')
plt.show()

# Define markers for watershed
markers = np.zeros_like(rocket_gray, dtype=np.int32)
markers[rocket_gray < 0.4] = 1
markers[rocket_gray > 0.6] = 2

plt.figure()
plt.imshow(markers, cmap='gray')
plt.title('Markers')
plt.axis('off')
plt.show()
```

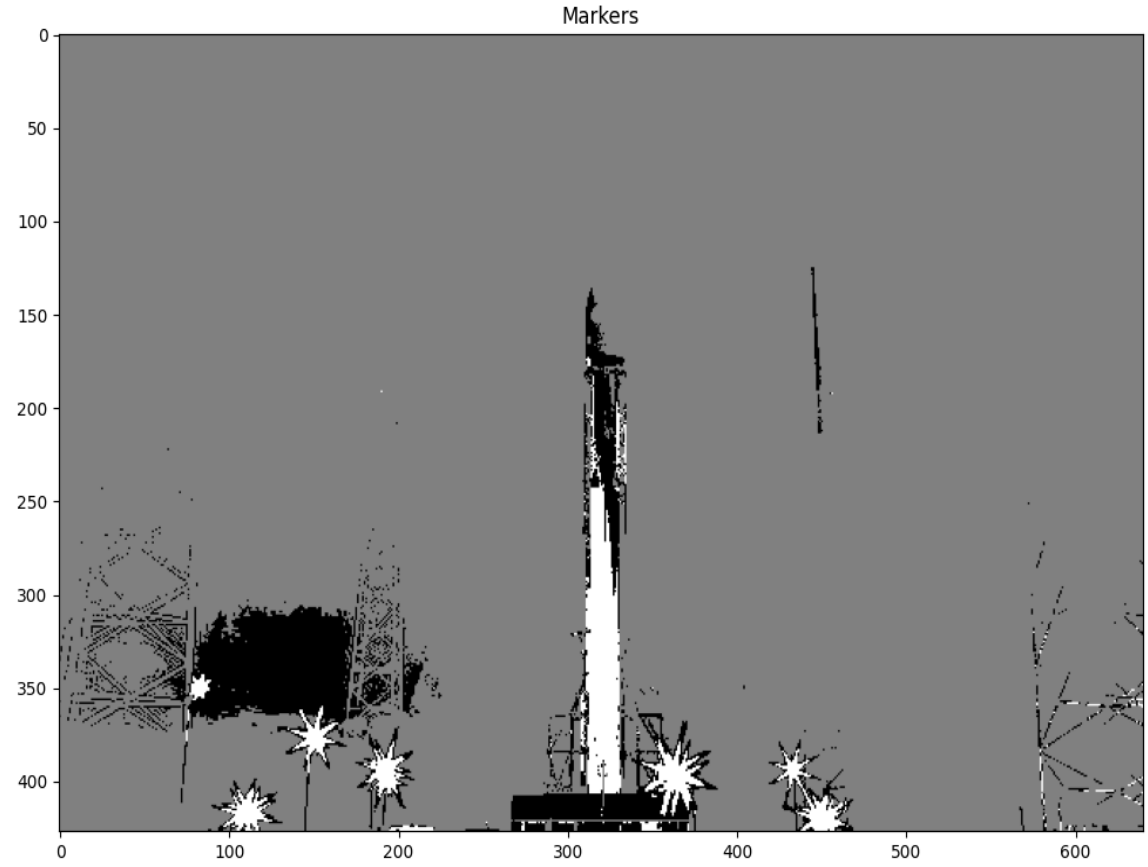
```
# Apply watershed segmentation
segmentation_result = segmentation.watershed(elevation_map, markers)

plt.figure()
plt.imshow(segmentation_result, cmap='gray')
plt.title('Watershed Segmentation')
plt.axis('off')
plt.show()

# Fill holes in the segmentation result
segmentation_filled = nd.binary_fill_holes(segmentation_result - 1)
label_rock, _ = nd.label(segmentation_filled)

# Create a label overlay
image_label_overlay = color.label2rgb(label_rock, image=rocket_gray, bg_label=0)

# Display the contour overlay and label overlay
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(24, 16), sharey=True)
ax1.imshow(rocket_gray, cmap='gray')
ax1.contour(segmentation_filled, [0.5], linewidths=1.8, colors='w')
ax1.set_title('Contour Overlay')
ax2.imshow(image_label_overlay)
ax2.set_title('Label Overlay')
plt.show()
```

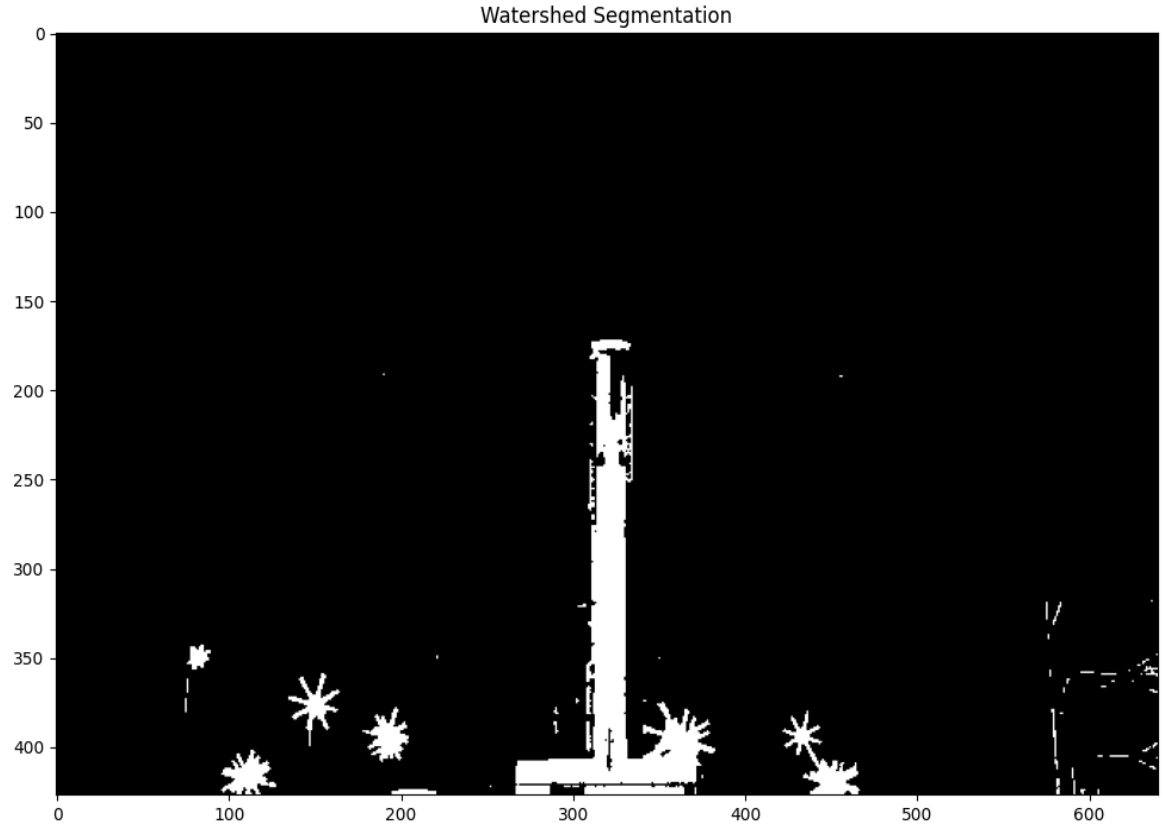
Output:

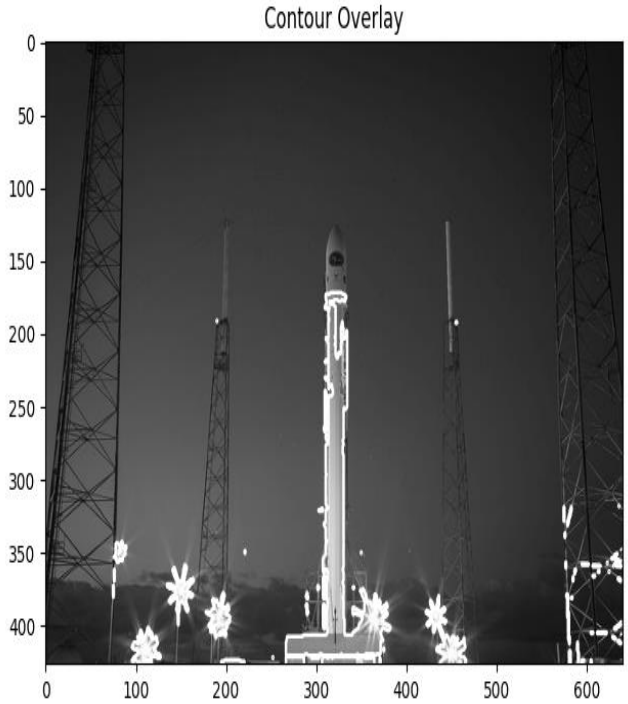

Original Image


Canny detector
0
50
100
150
200
250
300
350
400
0
100
200
300
400
500
600


Elevation Map
0
50
100
150
200
250
300
350
400
0
100
200
300
400
500
600


Markers
0
50
100
150
200
250
300
350
400
0
100
200
300
400
500
600


Watershed Segmentation
0
50
100
150
200
250
300
350
400
0
100
200
300
400
500
600


Contour Overlay
0
50
100
150
200
250
300
350
400
0
100
200
300
400
500
600


Label Overlay
0
100
200
300
400
500
600

Result:

Hence, the region based segmentation of an image has been successfully executed.