EXPT 8	Image Segmentation

AIM:-

To implement various image segmentation approaches including watershed (region-based), mean shift (cluster-based), active contour (model-based), and normalized cut (graph-based) methods for partitioning images into meaningful regions.

CODE:-

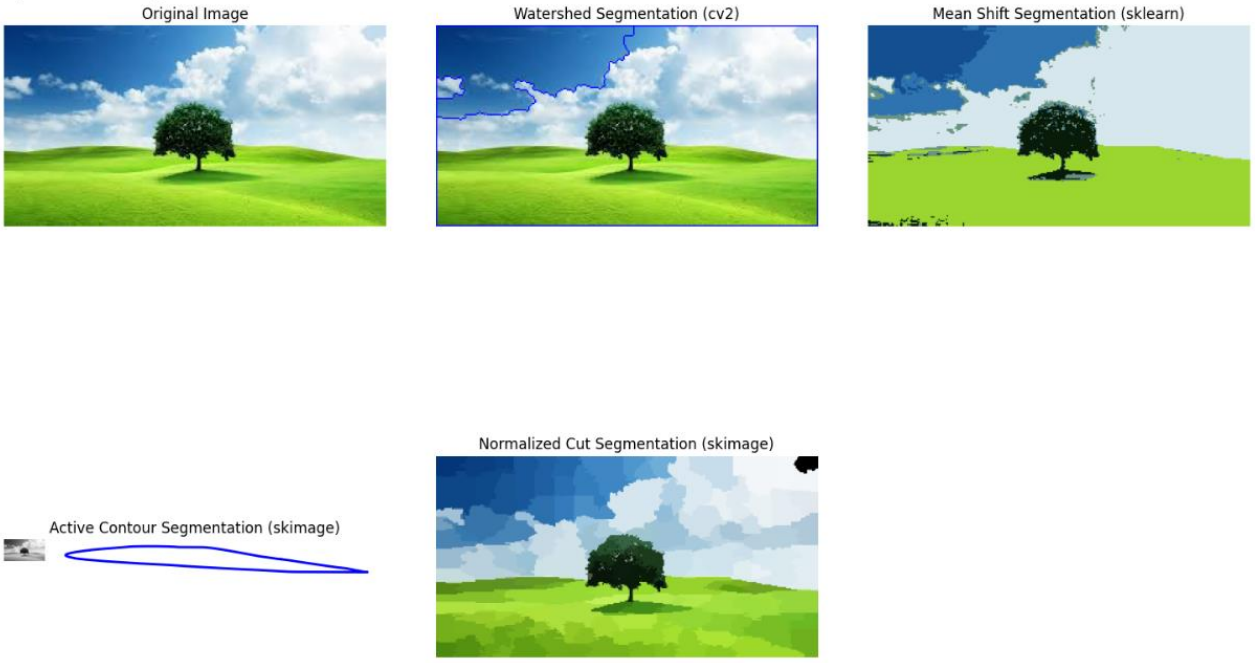
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from google.colab.patches import cv2_imshow # Keep this import for potential
future display outside subplots if needed, though plt.show() is used here.
from skimage import io, color, filters, morphology, segmentation, feature
from scipy import ndimage as ndi
from sklearn.cluster import MeanShift, estimate_bandwidth

# Load image
img_color = cv2.imread('input.jpg')
img_gray = cv2.imread('input.jpg', cv2.IMREAD_GRAYSCALE)

if img_color is None or img_gray is None:
    print("Upload 'input.jpg' to Colab.")
else:
    print("Image loaded successfully.")

    # Watershed Segmentation (using cv2)
    gray_ws = cv2.cvtColor(img_color, cv2.COLOR_BGR2GRAY)
    ret, thresh_ws = cv2.threshold(gray_ws, 0, 255, cv2.THRESH_BINARY_INV +
cv2.THRESH_OTSU)
    kernel_ws = np.ones((3, 3), np.uint8)
```

```
    opening_ws = cv2.morphologyEx(thresh_ws, cv2.MORPH_OPEN, kernel_ws,
iterations=2)
    sure_bg_ws = cv2.dilate(opening_ws, kernel_ws, iterations=3)
    dist_transform_ws = cv2.distanceTransform(opening_ws, cv2.DIST_L2, 5)
    ret, sure_fg_ws = cv2.threshold(dist_transform_ws, 0.7 *
dist_transform_ws.max(), 255, 0)
    sure_fg_ws = np.uint8(sure_fg_ws)
    unknown_ws = cv2.subtract(sure_bg_ws, sure_fg_ws)
    ret, markers_ws = cv2.connectedComponents(sure_fg_ws)
    markers_ws = markers_ws + 1
    markers_ws[unknown_ws == 255] = 0
    watershed_segmented_cv2 = img_color.copy()
    markers_ws = cv2.watershed(watershed_segmented_cv2, markers_ws)
    watershed_segmented_cv2[markers_ws == -1] = [255, 0, 0]

    # Mean Shift Segmentation (using sklearn)
    img_rgb_ms = cv2.cvtColor(img_color, cv2.COLOR_BGR2RGB)
    flat_image_ms = np.reshape(img_rgb_ms, [-1, 3])
    bandwidth_ms = estimate_bandwidth(flat_image_ms, quantile=0.1,
n_samples=500)
    ms = MeanShift(bandwidth=bandwidth_ms, bin_seeding=True)
    ms.fit(flat_image_ms)
    labels_ms = ms.labels_
    cluster_centers_ms = ms.cluster_centers_
    meanshift_segmented_sk =
cluster_centers_ms[labels_ms].reshape(img_rgb_ms.shape).astype(np.uint8)

    # Active Contour Segmentation (using skimage)
    img_norm_ac = img_gray / 255.0
    s_ac = np.linspace(0, 2*np.pi, 400)
    x_ac = 220 + 100*np.cos(s_ac)
    y_ac = 150 + 100*np.sin(s_ac)
    init_ac = np.array([x_ac, y_ac]).T
```

```
    activecontour_segmented_sk = segmentation.active_contour(img_norm_ac,
init_ac, alpha=0.015, beta=10, gamma=0.001)

    # Normalized Cut Segmentation (approx. with quickshift from skimage)
    img_rgb_nc = cv2.cvtColor(img_color, cv2.COLOR_BGR2RGB)
    labels_nc = segmentation.quickshift(img_rgb_nc, kernel_size=3, max_dist=6,
ratio=0.5)
    normalizedcut_segmented_sk = color.label2rgb(labels_nc, img_rgb_nc,
kind='avg')

    # Create subplots
    fig, axes = plt.subplots(nrows=2, ncols=3, figsize=(15, 10))
    axes = axes.ravel() # Flatten the axes array for easy indexing
    fig.tight_layout()

    # Display results
    axes[0].imshow(cv2.cvtColor(img_color, cv2.COLOR_BGR2RGB))
    axes[0].set_title('Original Image')
    axes[0].axis('off')

    axes[1].imshow(cv2.cvtColor(watershed_segmented_cv2,
cv2.COLOR_BGR2RGB))
    axes[1].set_title('Watershed Segmentation (cv2)')
    axes[1].axis('off')

    axes[2].imshow(meanshift_segmented_sk)
    axes[2].set_title('Mean Shift Segmentation (sklearn)')
    axes[2].axis('off')

    axes[3].imshow(img_gray, cmap='gray')
    axes[3].plot(activecontour_segmented_sk[:, 0], activecontour_segmented_sk[:, 1],
'-b', lw=2)
    axes[3].set_title('Active Contour Segmentation (skimage)')
    axes[3].axis('off')
```

```
    axes[4].imshow(normalizedcut_segmented_sk)
    axes[4].set_title('Normalized Cut Segmentation (skimage)')
    axes[4].axis('off')

    # Hide the unused subplot
    axes[5].axis('off')

    plt.show()
```

OUTPUT:-



RESULT:-

Successfully segmented images using different methodologies, each showing unique advantages based on image characteristics. The techniques effectively separated objects from backgrounds and partitioned images into homogeneous regions further analysis.