Practical Work 13
Developing a Program and Algorithm for Object Segmentation / Recognition in Images

1. Objectives of the Laboratory Work

- Understand the principles of image segmentation and object recognition.
- Learn how to separate objects from the background using thresholding, clustering, and contour-based methods.
- Implement basic object recognition using feature extraction and classification.
- Develop a complete algorithm for object segmentation and recognition in Python.
- Evaluate segmentation quality and recognition accuracy.

2. Theoretical Background

Image segmentation and object recognition are key steps in computer vision and digital image processing.
Segmentation refers to dividing an image into meaningful regions — typically separating the object(s) of interest from the background.
Recognition, on the other hand, involves identifying or classifying these segmented objects based on their shape, color, or texture.
In digital images, objects are often represented by connected regions of similar pixel values. The main challenge is to extract these regions accurately despite variations in illumination, noise, or occlusion.

2.1 Image Segmentation

Segmentation can be achieved through several techniques:

1. **Thresholding**
   The simplest segmentation technique. It converts a grayscale image into a binary image using a threshold value *T*.

    - If f(x, y) > T, the pixel is assigned to the object region.
    - Otherwise, it belongs to the background.

   Otsu's Method automatically determines an optimal threshold by minimizing intra-class variance.

   Advantages: Simple, fast, suitable for images with clear contrast.

   Limitations: Sensitive to lighting and noise.

2. **Edge-Based Segmentation**

   Detects object boundaries using edge detectors such as Sobel, Prewitt, or Canny. Edges are then connected to form closed contours.

   Useful when object boundaries are well defined.

3. **Region-Based Segmentation**

   Groups neighboring pixels with similar intensity or color values. Examples: region growing, region merging, and watershed algorithms.

4. **Clustering-Based Segmentation**

   Treats image pixels as data points in feature space (e.g., RGB color or intensity).

   Algorithms like **K-Means clustering** can partition pixels into *k* groups representing different objects or regions.

5. **Watershed Transform**

   Based on topographic interpretation of image gradients. It finds "catchment basins" corresponding to object regions.

2.2 Object Recognition

Once segmentation is complete, recognition involves identifying what the object is. Typical steps include:

1. **Feature Extraction** – extracting meaningful characteristics (e.g., color histograms, shape descriptors, edge counts).
2. **Feature Matching or Classification** – comparing extracted features with known templates or using machine learning classifiers (e.g., SVM, KNN, CNN).

3. **Decision Making** – assigning a label to the object.

For simple cases, geometric features like **area**, **perimeter**, **aspect ratio**, or **contour moments** are sufficient to distinguish between shapes.

2.3 Comparison of Segmentation Techniques

Method	Approach	Advantage	Limitation
Thresholding	Intensity-based	Simple, fast	Sensitive to illumination
Edge Detection	Gradient-based	Clear boundaries	May miss weak edges
Region Growing	Pixel similarity	Smooth regions	Sensitive to noise
K-Means Clustering	Statistical grouping	Works on color & texture	Needs number of clusters
Watershed	Gradient topology	Good shape separation	May over-segment

3. Practical Section: Implementation of Object Segmentation and Recognition

Task 1: Image Segmentation using Otsu Thresholding

Objective: Separate foreground objects from background automatically.

Instructions:

1. Load a grayscale image.
2. Apply global thresholding using Otsu's method.
3. Display the binary segmented result.

Code Example:

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

```
img = cv2.imread('/content/sample_data/objects.png',
cv2.IMREAD_GRAYSCALE)

# Apply Otsu's thresholding
ret, thresh = cv2.threshold(img, 0, 255, cv2.THRESH_BINARY +
cv2.THRESH_OTSU)

plt.figure(figsize=(8,4))
plt.subplot(1,2,1), plt.imshow(img, cmap='gray'), plt.title("Original Image")
plt.subplot(1,2,2), plt.imshow(thresh, cmap='gray'), plt.title("Otsu Segmented")
plt.show()
```

Expected Outcome:

The background is separated from the object(s) automatically, even with uneven illumination.

Task 2: Object Segmentation using K-Means Clustering

Objective: Perform color-based segmentation using clustering.

Instructions:

1. Convert the image to RGB and reshape it into a 2D array.
2. Apply K-Means clustering with *k = 3* (for example).
3. Display the clustered result.

Code Example:

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

img = cv2.imread('/content/sample_data/flowers.jpg')
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
pixels = img_rgb.reshape((-1, 3))
pixels = np.float32(pixels)
```

```
# Apply K-Means
k = 3
criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER,
100, 0.2)
_, labels, centers = cv2.kmeans(pixels, k, None, criteria, 10,
cv2.KMEANS_RANDOM_CENTERS)

centers = np.uint8(centers)
segmented_img = centers[labels.flatten()]
segmented_img = segmented_img.reshape(img_rgb.shape)

plt.figure(figsize=(8,4))
plt.subplot(1,2,1), plt.imshow(img_rgb), plt.title("Original")
plt.subplot(1,2,2), plt.imshow(segmented_img), plt.title("Segmented (K-Means)")
plt.show()
```

Expected Outcome:

The image is divided into clusters based on color similarity, isolating main objects or regions.

Task 3: Object Recognition using Contour Features

Objective: Detect and identify distinct objects (e.g., shapes) based on contour features.

Instructions:

1. Use the segmented binary image.
2. Find contours using cv2.findContours().
3. Calculate area, perimeter, and approximate shape.
4. Display recognized objects with labels.

Code Example:

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

img = cv2.imread('/content/sample_data/shapes.png')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
_, thresh = cv2.threshold(gray, 120, 255, cv2.THRESH_BINARY)
contours, _ = cv2.findContours(thresh, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)

for c in contours:
    approx = cv2.approxPolyDP(c, 0.02 * cv2.arcLength(c, True), True)
    x, y, w, h = cv2.boundingRect(approx)
    if len(approx) == 3:
        shape = "Triangle"
    elif len(approx) == 4:
        shape = "Rectangle"
    elif len(approx) > 10:
        shape = "Circle"
    else:
        shape = "Unknown"
    cv2.putText(img, shape, (x, y - 5), cv2.FONT_HERSHEY_SIMPLEX, 0.6,
(255,0,0), 2)
    cv2.drawContours(img, [approx], 0, (0,255,0), 2)

plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
plt.title("Object Recognition using Contours")
plt.axis('off')
plt.show()
```

Expected Outcome:
The program detects shapes and labels them (e.g., triangle, rectangle, circle) based on contour approximation.

4. Assignments for Students

Assignment Task 1 — Threshold-Based Segmentation
Use Otsu and adaptive thresholding methods on a grayscale image and compare their performance in noisy conditions.
Assignment Task 2 — Color Segmentation using K-Means
Perform K-Means segmentation with $k = 2, 3, 4$ on a color image. Evaluate how the number of clusters affects the result.
Assignment Task 3 — Object Shape Recognition
Develop a Python program to detect and classify basic geometric shapes (circle, triangle, square) using contour analysis.
Assignment Task 4 — Real-World Object Recognition (Optional Advanced)
Capture or download a real-world image (e.g., fruits, coins, traffic signs). Apply segmentation, extract features, and attempt recognition using simple classifiers or template matching.

5. Control Questions

1. What is the main purpose of image segmentation?
2. Describe the difference between segmentation and object recognition.
3. How does Otsu's method determine the threshold value?
4. What role does clustering play in segmentation?
5. Explain how contours are used to recognize shapes.
6. What are the main advantages of using K-Means for image segmentation?
7. How can illumination changes affect segmentation results?
8. What is the difference between global and adaptive thresholding?
9. How do you handle noise before segmentation to improve accuracy?

10.Mention at least two real-world applications of object recognition.