# 📘 Practical Code: Segmentation in Image Processing

**Objective:** To apply and understand simple image segmentation techniques using OpenCV and Python.

## 🔧 ****Step 1: Load Image and Convert to Grayscale****

python

CopyEdit

import cv2import numpy as npimport matplotlib.pyplot as plt

# Load color image

img = cv2.imread('shapes.png') # Replace with your own image file

img\_rgb = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)

# Convert to grayscale

gray = cv2.cvtColor(img\_rgb, cv2.COLOR\_RGB2GRAY)

# Show grayscale image

plt.imshow(gray, cmap='gray')

plt.title("Grayscale Image")

plt.axis('off')

plt.show()

## 🎯 ****1. Thresholding****

python

CopyEdit

# Convert grayscale to binary using a threshold

\_, binary = cv2.threshold(gray, 127, 255, cv2.THRESH\_BINARY)

# Show binary result

plt.imshow(binary, cmap='gray')

plt.title("Binary Image (Thresholding)")

plt.axis('off')

plt.show()

🧠 Explanation:

White = objects (pixel value ≥ 127)

Black = background (pixel value < 127)

## 🔢 ****2. Connected Components****

python

CopyEdit

# Apply connected components labeling

num\_labels, labels = cv2.connectedComponents(binary)

# Visualize labels

plt.imshow(labels, cmap='nipy\_spectral')

plt.title(f"Connected Components: {num\_labels - 1} objects")

plt.axis('off')

plt.show()

🧠 Explanation:

Each separate white region gets a unique ID

Background is usually label 0

Helps in object counting and separation

## ✏️ ****3. Edge Detection (Canny)****

python

CopyEdit

# Apply Canny edge detection

edges = cv2.Canny(gray, 100, 200)

# Show edge-detected image

plt.imshow(edges, cmap='gray')

plt.title("Edge Detection (Canny)")

plt.axis('off')

plt.show()

🧠 Explanation:

Detects outlines where image intensity changes quickly

Helps isolate shape borders

## 🌱 ****4. Region Growing (Manual)****

python

CopyEdit

# Create empty image for the region mask

mask = np.zeros\_like(gray, dtype=np.uint8)

# Choose a seed pixel (manually)

seed\_point = (100, 100) # You can adjust this

intensity\_diff = 10

# Get seed pixel intensity

seed\_val = gray[seed\_point]

# Grow the region by checking similar neighbors

queue = [seed\_point]

mask[seed\_point] = 255

while queue:

x, y = queue.pop(0)

for dx in [-1, 0, 1]:

for dy in [-1, 0, 1]:

nx, ny = x + dx, y + dy

if 0 <= nx < gray.shape[0] and 0 <= ny < gray.shape[1]:

if mask[nx, ny] == 0 and abs(int(gray[nx, ny]) - int(seed\_val)) < intensity\_diff:

mask[nx, ny] = 255

queue.append((nx, ny))

# Show region grown

plt.imshow(mask, cmap='gray')

plt.title("Region Growing Result")

plt.axis('off')

plt.show()

🧠 Explanation:

Starts from one point

Adds neighboring pixels with similar intensity

Expands until no more similar pixels found

## ✅ Summary Table

| **Operation** | **Function** | **Description** |
| --- | --- | --- |
| Thresholding | cv2.threshold() | Convert grayscale to binary |
| Connected Components | cv2.connectedComponents() | Detect and label separate objects |
| Edge Detection | cv2.Canny() | Find borders of objects |
| Region Growing | Custom code / cv2.floodFill() | Segment similar areas from a seed point |