**EX.NO.**: 04
**DATE**: 26.06.2025

## 2D AND 3D MULTI-OBJECT DETECTION AND SEGMENTATION

### AIM

To implement a system that performs **multi-object detection and segmentation** in both **2D images** and **3D point cloud data** using deep learning models, enabling automatic identification and localization of multiple objects along with their shapes.

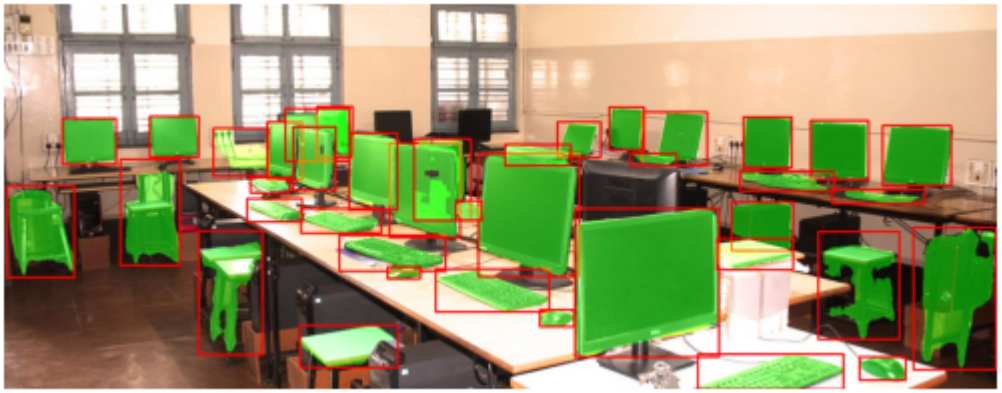
### ALGORITHM

1. **2D Object Detection and Segmentation (using Mask R-CNN)**
    a. Load a pre-trained Mask R-CNN model from PyTorch.
    b. Read and preprocess the input image (resize, normalize).
    c. Pass the image through the model to get:
        i. Bounding boxes
        ii. Class labels
        iii. Confidence scores
        iv. Segmentation masks
    d. Filter predictions by confidence threshold (e.g., > 0.5).
    e. Overlay bounding boxes and masks on the image.
    f. Display the segmented image.
2. **3D Object Segmentation (Simulated using Open3D)**
    a. Load a 3D point cloud file (`.pcd` or `.ply`).
    b. Convert the point cloud to a NumPy array of 3D points.
    c. Simulate object segmentation by assigning random class labels to each point.
    d. Assign unique colors to points based on their labels.
    e. Display the color-coded 3D point cloud using Open3D viewer.

### CODE AND OUTPUT

```
import torch
import torchvision
from torchvision.transforms import functional as F
from PIL import Image
import matplotlib.pyplot as plt
import numpy as np
import cv2

# Load pretrained Mask R-CNN
model = torchvision.models.detection.maskrcnn_resnet50_fpn(pretrained=True)
model.eval()

# Load and preprocess image
image_path = "lab.jpg"  # Replace with your 2D image
image = Image.open(image_path).convert("RGB")
image_tensor = F.to_tensor(image).unsqueeze(0)

# Perform detection
with torch.no_grad():
    output = model(image_tensor)[0]

# Visualize
```

```
def show_segmentation(image, output, threshold=0.5):
    image = np.array(image)
    masks = output['masks']
    boxes = output['boxes']
    labels = output['labels']
    scores = output['scores']

    for i in range(len(masks)):
        if scores[i] > threshold:
            mask = masks[i, 0].mul(255).byte().cpu().numpy()
            color_mask = np.zeros_like(image)
            color_mask[mask > 128] = [0, 255, 0]
            image = cv2.addWeighted(image, 1, color_mask, 0.5, 0)
            box = boxes[i].detach().numpy().astype(int)
            cv2.rectangle(image, tuple(box[:2]), tuple(box[2:]), (255, 0, 0), 2)
    plt.imshow(image)
    plt.axis("off")
    plt.show()

show_segmentation(image, output)
```



```
import open3d as o3d
import numpy as np

# Load sample point cloud
pcd = o3d.io.read_point_cloud("sample.pcd")  # Replace with your point cloud file
o3d.visualization.draw_geometries([pcd])

# Simulated segmentation mask (e.g., from PointNet)
# Assigning random labels (in practice, this comes from your PointNet model)
points = np.asarray(pcd.points)
num_points = points.shape[0]
labels = np.random.randint(0, 3, size=num_points)  # Simulating 3 object classes

# Assign colors based on labels
colors = np.zeros((num_points, 3))
colors[labels == 0] = [1, 0, 0]
colors[labels == 1] = [0, 1, 0]
colors[labels == 2] = [0, 0, 1]
pcd.colors = o3d.utility.Vector3dVector(colors)
```

```
# Visualize segmented point cloud
o3d.visualization.draw_geometries([pcd])
```