EX.NO 1:

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

img = cv2.imread(r'D:\dl\peacock.png')
if img is None: raise FileNotFoundError("Image not found")
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
k = 7

filters = [
    cv2.blur(img, (k, k)),
    cv2.GaussianBlur(img, (k, k), 0),
    cv2.medianBlur(img, k),
    cv2.bilateralFilter(img, 9, 75, 75),
    cv2.Canny(gray, 100, 200),
    cv2.dilate(gray, np.ones((5, 5), np.uint8), 1),
    cv2.erode(gray, np.ones((5, 5), np.uint8), 1)
]

sobel = cv2.Sobel(gray, cv2.CV_64F, 1, 0, 5) + cv2.Sobel(gray, cv2.CV_64F, 0, 1, 5)
sobel = cv2.normalize(np.abs(sobel), None, 0, 255, cv2.NORM_MINMAX, cv2.CV_8U)
filters.insert(4, sobel)

titles = ['Original', 'Average', 'Gaussian', 'Median', 'Sobel', 'Bilateral', 'Canny', 'Dilated', 'Eroded']
imgs = [img] + filters

plt.figure(figsize=(15, 10))
for i in range(9):
    plt.subplot(3, 3, i + 1)
    plt.imshow(imgs[i], cmap='gray' if len(imgs[i].shape) == 2 else None)
    plt.title(titles[i]); plt.axis('off')
plt.tight_layout(); plt.show()
```

EX.NO 2:

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

img = cv2.imread(r'D:\dl\pe.jpg')
if img is None: raise FileNotFoundError("Image not found")
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Grayscale histogram
plt.figure(); plt.title('Grayscale Histogram')
plt.plot(cv2.calcHist([gray], [0], None, [256], [0,256]), 'k'); plt.show()

# Color histogram (B, G, R)
colors = ('b', 'g', 'r')
plt.figure(); plt.title('Color Histogram')
for i, c in enumerate(colors):
    plt.plot(cv2.calcHist([img], [i], None, [256], [0,256]), color=c)
plt.show()

# Histogram Equalization
eq = cv2.equalizeHist(gray)
plt.figure(); plt.title('Equalized Grayscale Histogram')
plt.plot(cv2.calcHist([eq], [0], None, [256], [0,256]), 'k'); plt.show()

cv2.imshow('Original', gray)
cv2.imshow('Equalized', eq)
cv2.waitKey(0); cv2.destroyAllWindows()

# 2D HSV Histogram
hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
hist2d = cv2.calcHist([hsv], [0,1], None, [30,32], [0,180,0,256])
plt.imshow(hist2d, interpolation='nearest'); plt.title('2D Hue-Saturation'); plt.show()
```

EX.NO 3:

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

img = cv2.imread(r'D:\dl\pe.jpg')
if img is None: raise FileNotFoundError("Image not found")
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

# 1. Simple Threshold
_, th1 = cv2.threshold(gray, 127, 255, cv2.THRESH_BINARY)

# 2. Adaptive Threshold
th2 = cv2.adaptiveThreshold(gray, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
                            cv2.THRESH_BINARY, 11, 2)

# 3. Otsu Threshold
_, th3 = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)

# 4. Watershed Segmentation
_, th4 = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY_INV + cv2.THRESH_OTSU)
kernel = np.ones((3,3), np.uint8)
opening = cv2.morphologyEx(th4, cv2.MORPH_OPEN, kernel, iterations=2)
sure_bg = cv2.dilate(opening, kernel, iterations=3)
dist = cv2.distanceTransform(opening, cv2.DIST_L2, 5)
_, sure_fg = cv2.threshold(dist, 0.7*dist.max(), 255, 0)
sure_fg = np.uint8(sure_fg)
unknown = cv2.subtract(sure_bg, sure_fg)
_, markers = cv2.connectedComponents(sure_fg)
markers = markers + 1
markers[unknown == 255] = 0
markers = cv2.watershed(img, markers)
water = img.copy()
water[markers == -1] = [255, 0, 0]
```

```
# 5. K-means
Z = img.reshape((-1, 3)).astype(np.float32)
_, label, center = cv2.kmeans(Z, 4, None, (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 10, 1.0),
                              10, cv2.KMEANS_RANDOM_CENTERS)
img_km = center[np.uint8(label)].reshape(img.shape)

# 6. GrabCut
mask = np.zeros(img.shape[:2], np.uint8)
bg, fg = np.zeros((1,65), np.float64), np.zeros((1,65), np.float64)
rect = (10, 10, img.shape[1]-30, img.shape[0]-30)
cv2.grabCut(img, mask, rect, bg, fg, 5, cv2.GC_INIT_WITH_RECT)
mask2 = np.where((mask==2)|(mask==0), 0, 1).astype('uint8')
grab = img * mask2[:,:,np.newaxis]

# Display
titles = ['Original', 'Simple', 'Adaptive', 'Otsu', 'Watershed', 'K-means', 'GrabCut']
imgs = [img_rgb, th1, th2, th3, cv2.cvtColor(water, cv2.COLOR_BGR2RGB), img_km, cv2.cvtColor(grab,
cv2.COLOR_BGR2RGB)]

plt.figure(figsize=(15,10))
for i in range(7):
    plt.subplot(2,4,i+1)
    plt.imshow(imgs[i], cmap='gray' if len(imgs[i].shape)==2 else None)
    plt.title(titles[i]); plt.axis('off')
plt.tight_layout(); plt.show()
```

EX.NO 4:

```
import cv2, numpy as np

def label_objects(path, color_ranges, labels):
    img = cv2.imread(path)
    if img is None: raise FileNotFoundError("Image not found")
    hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

    for i in range(0, len(color_ranges), 2):
        mask = cv2.inRange(hsv, np.array(color_ranges[i]), np.array(color_ranges[i+1]))
        contours, _ = cv2.findContours(mask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

        for c in contours:
            if cv2.contourArea(c) < 100: continue
            x, y, w, h = cv2.boundingRect(c)
            cv2.rectangle(img, (x,y), (x+w,y+h), (0,255,0), 2)
            cv2.putText(img, labels[i//2], (x, y-10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0,255,0), 2)

    cv2.imshow("Labeled Objects", img)
    cv2.waitKey(0); cv2.destroyAllWindows()

# Example
color_ranges = [
    (0,100,100),(10,255,255),    # Red
    (25,100,100),(35,255,255),   # Yellow
    (100,100,100),(120,255,255)  # Blue
]
labels = ["Red", "Yellow", "Blue"]
label_objects(r"D:\dl\apple.jpg", color_ranges, labels)
```

EX.NO 5:

```
import cv2

def detect_faces(img_path):
    model = r"D:\DL\res10_300x300_ssd_iter_140000.caffemodel"
    config = r"D:\DL\deploy.prototxt"
    net = cv2.dnn.readNetFromCaffe(config, model)
    img = cv2.imread(img_path)
    if img is None: raise FileNotFoundError("Image not found")
    h, w = img.shape[:2]

    blob = cv2.dnn.blobFromImage(cv2.resize(img, (300, 300)), 1.0, (300,300), (104,177,123))
    net.setInput(blob)
    detections = net.forward()

    count = 0
    for i in range(detections.shape[2]):
        conf = detections[0,0,i,2]
        if conf > 0.5:
            box = (detections[0,0,i,3:7] * [w,h,w,h]).astype(int)
            x1, y1, x2, y2 = box
            cv2.rectangle(img, (x1,y1), (x2,y2), (0,255,0), 2)
            cv2.putText(img, "Face", (x1, y1-10), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0,255,0), 2)
            count += 1

    print("Number of faces detected:", count)
    cv2.putText(img, f"Faces: {count}", (10,30), cv2.FONT_HERSHEY_SIMPLEX, 1, (0,0,255), 2)
    cv2.imshow("Face Detection", img)
    cv2.waitKey(0); cv2.destroyAllWindows()

# Example
detect_faces(r"D:\DL\grppeople.jpg")
```

EX.NO 6:

```
import cv2, numpy as np, pytesseract, os

pytesseract.pytesseract.tesseract_cmd = r'C:\Program Files\Tesseract-OCR\tesseract.exe'
cascade_path = r"D:\dl\haarcascade_russian_plate_number.xml"
if not os.path.exists(cascade_path): raise FileNotFoundError("Cascade not found")

states = {"TN":"Tamil Nadu","KL":"Kerala","DL":"Delhi","MH":"Maharashtra","KA":"Karnataka"}

def detect_plate(img_path):
    cascade = cv2.CascadeClassifier(cascade_path)
    img = cv2.imread(img_path)
    if img is None: raise FileNotFoundError("Image not found")
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    plates = cascade.detectMultiScale(gray, 1.1, 4)

    for (x,y,w,h) in plates:
        plate = img[y:y+h, x:x+w]
        plate_gray = cv2.cvtColor(plate, cv2.COLOR_BGR2GRAY)
        _, bin_plate = cv2.threshold(plate_gray, 127, 255, cv2.THRESH_BINARY)
        text = pytesseract.image_to_string(bin_plate, config='--psm 8')
        text = ''.join(e for e in text if e.isalnum()).upper()
        state = states.get(text[:2], "Unknown")

        cv2.rectangle(img, (x,y), (x+w,y+h), (0,0,255), 2)
        cv2.putText(img, text, (x, y-10), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255,255,255), 2)
        cv2.putText(img, state, (x, y+h+25), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0,255,0), 2)
        print(f"Detected Number: {text} | State: {state}")

    cv2.imshow("License Plate Detection", img)
    cv2.waitKey(0); cv2.destroyAllWindows()

# Example
detect_plate(r"D:\dl\car.jpg")
```

EX.NO 7:

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

def process_medical_image(path):
    img = cv2.imread(path, cv2.IMREAD_GRAYSCALE)
    if img is None: raise FileNotFoundError("Image not found")

    blur = cv2.GaussianBlur(img, (5,5), 0)
    _, thresh = cv2.threshold(blur, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)
    contours, _ = cv2.findContours(thresh, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
    contoured = cv2.cvtColor(img, cv2.COLOR_GRAY2BGR)
    cv2.drawContours(contoured, contours, -1, (0,255,0), 2)

    titles = ["Original", "Blurred", "Thresholded", "Contours"]
    images = [img, blur, thresh, contoured]
    plt.figure(figsize=(12,5))
    for i in range(4):
        plt.subplot(1,4,i+1); plt.imshow(images[i], cmap='gray'); plt.title(titles[i]); plt.axis('off')
    plt.tight_layout(); plt.show()

# Example
process_medical_image(r'D:\dl\brain.png')
```

EX.NO 8:

```
import cv2, numpy as np

img = cv2.imread('star.jpg', cv2.IMREAD_GRAYSCALE)
if img is None: raise FileNotFoundError("Image not found")

# Edge Detection
edges = cv2.Canny(img, 50, 150)

# Corner Detection
corners = cv2.goodFeaturesToTrack(img, 200, 0.01, 10)
corn_img = cv2.cvtColor(img, cv2.COLOR_GRAY2BGR)
if corners is not None:
    for c in np.intp(corners):
        x, y = c.ravel()
        cv2.circle(corn_img, (x, y), 3, (0, 255, 0), -1)

# Line Detection
line_img = cv2.cvtColor(img, cv2.COLOR_GRAY2BGR)
lines = cv2.HoughLines(cv2.Canny(img, 50, 150), 1, np.pi/180, 150)
if lines is not None:
    for rho, theta in lines[:,0]:
        a, b = np.cos(theta), np.sin(theta)
        x0, y0 = a*rho, b*rho
        x1, y1 = int(x0 + 1000*(-b)), int(y0 + 1000*(a))
        x2, y2 = int(x0 - 1000*(-b)), int(y0 - 1000*(a))
        cv2.line(line_img, (x1,y1), (x2,y2), (0,0,255), 1)

cv2.imshow("Original", img)
cv2.imshow("Edges", edges)
cv2.imshow("Corners", corn_img)
cv2.imshow("Lines", line_img)
cv2.waitKey(0); cv2.destroyAllWindows()
```

EX.NO 9:

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

def hist3d(img):
    hist = cv2.calcHist([img], [0,1,2], None, [8,8,8], [0,256,0,256,0,256])
    return cv2.normalize(hist, hist).flatten()

def compare_faces(test, knowns, thr=0.5):
    test_hist = hist3d(test)
    dists = [cv2.compareHist(test_hist, hist3d(k), cv2.HISTCMP_BHATTACHARYYA) for k in knowns]
    idx = np.argmin(dists)
    return idx if dists[idx] <= thr else None

test = cv2.imread('female1.jpg')
knowns = [cv2.imread(f'female.jpg') for _ in range(3)]

idx = compare_faces(test, knowns)
print(f"Face recognized as person {idx+1}" if idx is not None else "Face not recognized")

plt.figure(figsize=(10,5))
plt.subplot(1,4,1); plt.imshow(cv2.cvtColor(test, cv2.COLOR_BGR2RGB)); plt.title("Test")
for i, k in enumerate(knowns):
    plt.subplot(1,4,i+2); plt.imshow(cv2.cvtColor(k, cv2.COLOR_BGR2RGB)); plt.title(f"Known {i+1}")
plt.tight_layout(); plt.show()
```

EX.NO 10:

```
import cv2
import numpy as np

# Load authorized image
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
auth_img = cv2.imread('obma1.jpg', 0)
orb = cv2.ORB_create()
kp1, des1 = orb.detectAndCompute(auth_img, None)

def is_authorized(test_path):
    img = cv2.imread(test_path)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, 1.1, 4)
    for (x, y, w, h) in faces:
        roi = gray[y:y+h, x:x+w]
        kp2, des2 = orb.detectAndCompute(roi, None)
        if des2 is not None:
            bf = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)
            matches = bf.match(des1, des2)
            if len([m for m in matches if m.distance < 50]) > 10:
                return True
    return False

print("Authorized" if is_authorized('Obma.jpg') else "Unauthorized")
```

EX.NO 11:

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


def disparity_map(left, right):
    L = cv2.imread(left, 0)
    R = cv2.imread(right, 0)
    stereo = cv2.StereoSGBM_create(minDisparity=0, numDisparities=96, blockSize=7)
    disp = stereo.compute(L, R).astype(np.float32) / 16.0
    disp = cv2.normalize(disp, None, 0, 255, cv2.NORM_MINMAX).astype(np.uint8)
    plt.subplot(1,3,1), plt.imshow(L, 'gray'), plt.title('Left')
    plt.subplot(1,3,2), plt.imshow(R, 'gray'), plt.title('Right')
    plt.subplot(1,3,3), plt.imshow(disp, 'plasma'), plt.title('Disparity')
    plt.show()


disparity_map('D:/dl/ryt.png', 'D:/dl/sstero.png')
```

EX.NO 12:

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


net = cv.dnn.readNetFromTensorflow('D:/dl/graph_opt.pb')
BODY = {"Nose":0,"Neck":1,"RShoulder":2,"RElbow":3,"RWrist":4,"LShoulder":5,"LElbow":6,"LWrist":7,
      "RHip":8,"RKnee":9,"RAnkle":10,"LHip":11,"LKnee":12,"LAnkle":13,"REye":14,"LEye":15,"REar":16,"LEar":17}
PAIRS = [("Neck","RShoulder"),("Neck","LShoulder"),("RShoulder","RElbow"),("RElbow","RWrist"),
       ("LShoulder","LElbow"),("LElbow","LWrist"),("Neck","RHip"),("RHip","RKnee"),("RKnee","RAnkle"),
       ("Neck","LHip"),("LHip","LKnee"),("LKnee","LAnkle"),("Neck","Nose"),
       ("Nose","REye"),("REye","REar"),("Nose","LEye"),("LEye","LEar")]


def pose(frame):
   blob = cv.dnn.blobFromImage(frame, 1.0, (368,368), (127.5,127.5,127.5), swapRB=True, crop=False)
   net.setInput(blob); out = net.forward()[:, :len(BODY), :, :]
   h, w = frame.shape[:2]; pts=[]
   for i in range(len(BODY)):
      hm = out[0, i, :, :]; _, conf, _, p = cv.minMaxLoc(hm)
      x, y = int(w*p[0]/hm.shape[1]), int(h*p[1]/hm.shape[0])
      pts.append((x,y) if conf>0.2 else None)
   for a,b in PAIRS:
      if pts[BODY[a]] and pts[BODY[b]]:
         cv.line(frame, pts[BODY[a]], pts[BODY[b]], (0,255,0), 2)
   return frame


img = cv.imread('D:/dl/humafomal.jpg')
plt.imshow(cv.cvtColor(pose(img), cv.COLOR_BGR2RGB)); plt.axis('off'); plt.show()
```

EX.NO 13:

```
import cv2, numpy as np

net = cv2.dnn.readNet('D:/dl/yolov4.weights', 'D:/dl/yolov4.cfg')
classes = open('D:/dl/coco.names').read().splitlines()
cap = cv2.VideoCapture('traffic.mp4')

while True:
    ret, frame = cap.read()
    if not ret: break
    h, w = frame.shape[:2]
    blob = cv2.dnn.blobFromImage(frame, 1/255, (416,416), swapRB=True)
    net.setInput(blob)
    outs = net.forward([net.getLayerNames()[i - 1] for i in net.getUnconnectedOutLayers().flatten()])
    boxes, confs, ids = [], [], []
    for o in outs:
        for det in o:
            s = det[5:]; id = np.argmax(s); conf = s[id]
            if conf > 0.3 and classes[id] in ["car","bus","truck","motorbike"]:
                cx, cy, ww, hh = (det[:4]*[w,h,w,h]).astype(int)
                boxes.append([cx-ww//2, cy-hh//2, ww, hh]); confs.append(float(conf)); ids.append(id)
    idx = cv2.dnn.NMSBoxes(boxes, confs, 0.3, 0.4)
    for i in idx.flatten():
        x,y,ww,hh = boxes[i]
        cv2.rectangle(frame,(x,y),(x+ww,y+hh),(0,255,0),2)
        cv2.putText(frame, classes[ids[i]], (x,y-10), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0,255,0),2)
    cv2.imshow('Traffic Detection', frame)
    if cv2.waitKey(10)&0xFF==ord('q'): break
cap.release(); cv2.destroyAllWindows()
```

EX.NO 14:

```
import numpy as np, cv2
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image

model = load_model('model.h5')
actions = ['walking','running','jumping','standing','sitting','falling','other']

def predict_action(img_path):
    img = image.load_img(img_path, target_size=(90,3), color_mode='grayscale')
    arr = image.img_to_array(img)/255.0
    arr = np.expand_dims(arr, 0)
    pred = actions[np.argmax(model.predict(arr))]
    return pred

print("Action:", predict_action('jump4.jpg'))
```

EX. NO 15:

```
import cv2, numpy as np


def detect_road(img_path):
    img = cv2.imread(img_path)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    edges = cv2.Canny(cv2.GaussianBlur(gray,(5,5),0),50,150)
    h,w = edges.shape
    mask = np.zeros_like(edges)
    roi = np.array([[(0,h),(w*0.1,h*0.5),(w*0.9,h*0.5),(w,h)]], np.int32)
    cv2.fillPoly(mask, roi, 255)
    masked = cv2.bitwise_and(edges, mask)
    lines = cv2.HoughLinesP(masked,1,np.pi/180,30,minLineLength=50,maxLineGap=30)
    res = img.copy()
    if lines is not None:
        for x1,y1,x2,y2 in lines[:,0]:
            slope=(y2-y1)/(x2-x1+1e-6)
            color=(255,0,0) if slope<-0.2 else (0,255,0)
            cv2.line(res,(x1,y1),(x2,y2),color,2)
    cv2.imshow('Road Detection', res)
    cv2.waitKey(0); cv2.destroyAllWindows()


detect_road('road.png')
```