**Department of Computing**

**Digital Image Processing**

**Class: BSCS-9ABC**

**Lab 11**

**Submitted By: Fatima Seemab**

**CMS ID: 291310**

**BSCS 9B**

**TASK 1**

| import cv2  import numpy as np  def nothing(x):  pass  # cap = cv2.VideoCapture(0);  cv2.namedWindow("Tracking")  cv2.createTrackbar("LH", "Tracking", 0, 255, nothing)  cv2.createTrackbar("LS", "Tracking", 0, 255, nothing)  cv2.createTrackbar("LV", "Tracking", 0, 255, nothing)  cv2.createTrackbar("UH", "Tracking", 255, 255, nothing)  cv2.createTrackbar("US", "Tracking", 255, 255, nothing)  cv2.createTrackbar("UV", "Tracking", 255, 255, nothing)  while True:  frame = cv2.imread('smarties.jpg')  hsv = cv2.cvtColor(frame, cv2.COLOR\_BGR2HSV)  l\_h = cv2.getTrackbarPos("LH", "Tracking")  l\_s = cv2.getTrackbarPos("LS", "Tracking")  l\_v = cv2.getTrackbarPos("LV", "Tracking")  u\_h = cv2.getTrackbarPos("UH", "Tracking")  u\_s = cv2.getTrackbarPos("US", "Tracking")  u\_v = cv2.getTrackbarPos("UV", "Tracking")  l\_b = np.array([l\_h, l\_s, l\_v])  u\_b = np.array([u\_h, u\_s, u\_v])  mask = cv2.inRange(hsv, l\_b, u\_b)  res = cv2.bitwise\_and(hsv, hsv, mask=mask)  cv2.imshow("frame", hsv)  cv2.imshow("mask", mask)  cv2.imshow("res", res)  key = cv2.waitKey(1)  if key == 27:  break  # cap.release()  cv2.destroyAllWindows() |
| --- |

**OUTPUTS:**

|  |
| --- |

**TASK 2**

**CODE:**

| #importing the packages  import cv2  import numpy as np  from skimage.morphology import disk    def nothing(x):  pass  #morphological opertaions  def closing(img,kernel):  dilation=cv2.dilate(img,kernel,iterations=1)  closing=cv2.erode(dilation,kernel,iterations=1)  return closing  def opening(img,kernel):  erosion=cv2.erode(img,kernel,iterations=1)  opening=cv2.dilate(erosion,kernel,iterations=1)  return opening  # Open the camera  cap = cv2.VideoCapture(0)    # Create a window  cv2.namedWindow('image')    # create trackbars for color change  cv2.createTrackbar('lowH','image',0,179,nothing)  cv2.createTrackbar('highH','image',179,179,nothing)    cv2.createTrackbar('lowS','image',0,255,nothing)  cv2.createTrackbar('highS','image',255,255,nothing)    cv2.createTrackbar('lowV','image',0,255,nothing)  cv2.createTrackbar('highV','image',255,255,nothing)    while(True):  ret, frame = cap.read()    # get current positions of the trackbars  ilowH = cv2.getTrackbarPos('lowH', 'image')  ihighH = cv2.getTrackbarPos('highH', 'image')  ilowS = cv2.getTrackbarPos('lowS', 'image')  ihighS = cv2.getTrackbarPos('highS', 'image')  ilowV = cv2.getTrackbarPos('lowV', 'image')  ihighV = cv2.getTrackbarPos('highV', 'image')    # convert color to hsv because it is easy to track colors in this color model  hsv = cv2.cvtColor(frame, cv2.COLOR\_BGR2HSV)  lower\_hsv = np.array([ilowH, ilowS, ilowV])  higher\_hsv = np.array([ihighH, ihighS, ihighV])  # Apply the cv2.inrange method to create a mask  mask = cv2.inRange(hsv, lower\_hsv, higher\_hsv)  # Apply the mask on the image to extract the original color  frame = cv2.bitwise\_and(frame, frame, mask=mask)  cv2.imshow('image', frame)  # Press q to exit  if cv2.waitKey(1) & 0xFF == ord('q'):  cv2.imwrite('image\_captured.png', frame)  break  cap.release()  cv2.destroyAllWindows()  noise\_removal=closing(frame,disk(2.5))  cv2.imwrite('noise\_removal.png', noise\_removal)  # checks whether the image is RGB or grayscale  if(len(noise\_removal.shape)>2):  print("RGB")  else:  print("gray")  #converting to white and black image  noise\_removal\_gray = cv2.cvtColor(noise\_removal, cv2.COLOR\_BGR2GRAY)  ret,thresh=cv2.threshold(noise\_removal\_gray,127,255,0)  #For findingg the contours  contours,hierarchy = cv2.findContours(thresh, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_NONE)  print("Number of Contours="+str(len(contours)))  #For iterating over all the contours  for c in contours:  #to create the bounding boxes  rect = cv2.boundingRect(c)  if rect[2] < 100 or rect[3] < 100: continue  print(cv2.contourArea(c))  x, y, w, h = rect  #to create the rectangle bounding boxes  cv2.rectangle(noise\_removal, (x, y), (x + w, y + h), (255, 255, 0), 2)  cv2.putText(noise\_removal, 'Mobile Detected', (x + w + 10, y + h), 0, 0.3, (255, 255, 0))  # for finding the center of contours  M = cv2.moments(c)  cX = int(M["m10"] / M["m00"])  cY = int(M["m01"] / M["m00"])  # for placing the circle at the center of contours  cv2.circle(noise\_removal, (cX, cY), 5, (255, 0,0), -1)  #For drawing over all the contours  cv2.drawContours(noise\_removal,contours,-1,(0,255,0),3)  cv2.imwrite('contours.png', noise\_removal) |
| --- |

**OUTPUT:**

| **Image Captured** | **After Noise Removal** | **Bounding boxes and contours** |
| --- | --- | --- |
|  |  |  |