**IP Task- 1**

**Group- 3**

**Aim:** Shaft Detection

**Procedure**:

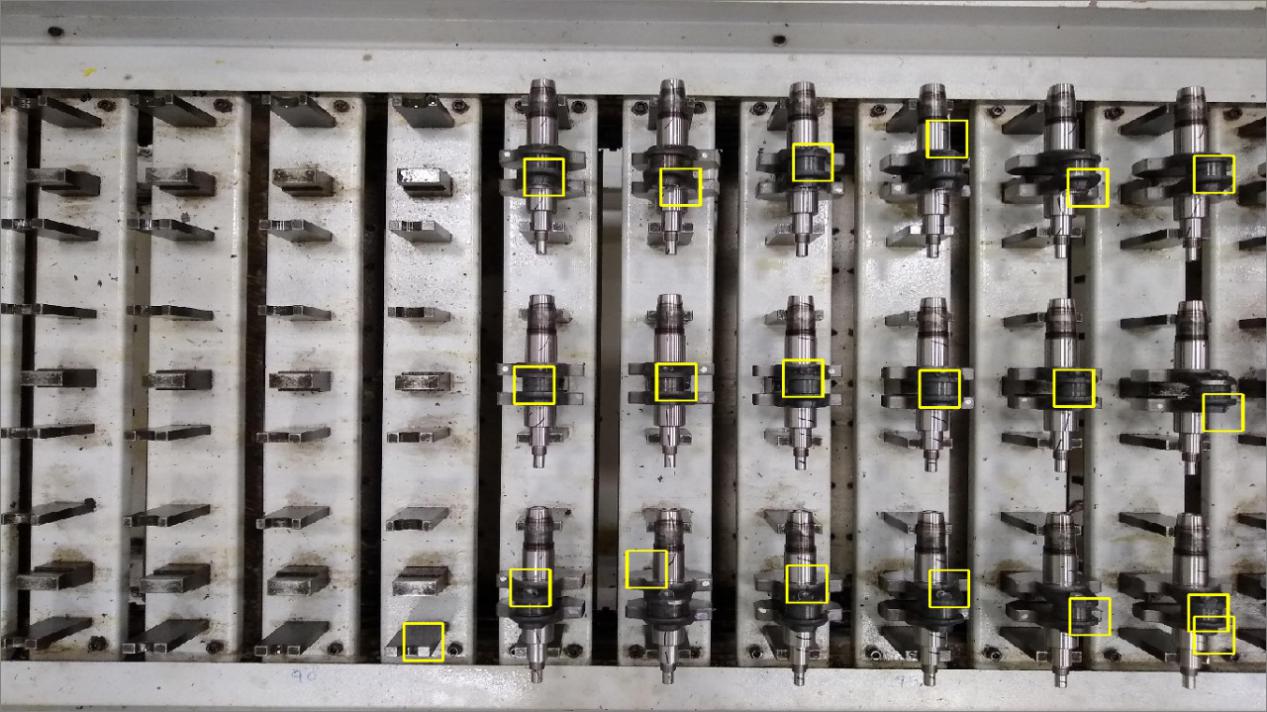
1. Using Morphological Opening followed by Dilation (Best fit, 2nd attempt)
2. First convert the image to HSV.
3. Set lower and higher bounds for masking the image at [47,0,0] and [255,48,77] respectively.
4. Masking the image.
5. Use Morphological Opening on the masked image to eliminate noise.

Morphological Opening consists of Erosion followed by Dilation. Erosion removes noise but erodes the boundaries of the object, thus we use dilation to increase the area of the object, achieving the required objective.

1. Using Dilation again to increase the area of the object.
2. Finally drawing rectangular contours on the objects whose area is greater than 350.
3. Print number of contours thus detecting the number of shafts in the image.

**Code:**

**import** cv2  
 **import** numpy **as** np  
 *# Load an image* image = cv2.imread(**'images/1.jpeg'**)  
 frame = cv2.cvtColor(image, cv2.COLOR\_BGR2HSV)  
 *# masking* bgrl = np.array([42, 0, 0]) *# black* bgrh = np.array([255, 48, 77]) *# black* mask = cv2.inRange(frame, bgrl, bgrh)  
 *# image clearing techniques* kernel = np.ones((2, 2), np.uint8)  
 morph = cv2.morphologyEx(mask, cv2.MORPH\_OPEN, kernel)  
 blur = cv2.dilate(morph, kernel, iterations=2)  
  
 cv2.imshow(**'masked'**, blur)  
 *# Find Contours* contours, \_ = cv2.findContours(blur, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)  
 *# print("Total number of contours : ", len(contours))* count = 0  
 **for** i **in** contours:  
 area = cv2.contourArea(i)  
 *# print(i)  
 # print(area)  
 # finding best contour* **if** area > 395:  
 count += 1  
 cv2.drawContours(image, i, -1,(0, 230, 0),2)  
 x, y, w, h = cv2.boundingRect(i)  
 cv2.rectangle(image,(x, y),(x+40,   
 y+50),(0,255,255),2)  
  
 print(**"Number of contours with marked rectangle : "**, count)  
  
 cv2.namedWindow(**'Contours'**, cv2.WND\_PROP\_FULLSCREEN)  
 cv2.setWindowProperty(**'Contours'**, cv2.WND\_PROP\_FULLSCREEN, cv2.WINDOW\_FULLSCREEN)  
  
 cv2.imshow(**'Contours'**, image)  
 cv2.waitKey(0)



1. Using Median Blur (1st attempt)
2. Reading the image and applying Median Blur to it.

Median blur reduces the noise while retaining the edges of the objects.

1. Set lower and higher bounds for masking the image at [47,0,0] and [255,48,77] respectively.
2. Masking the image.
3. Finding the contours.
4. Applying rectangular contours on all objects whose area is greater than 350.
5. Print number of contours thus detecting the number of shafts in the image.

**Code:**

**import** cv2  
 **import** numpy **as** np  
 *# Load an image* image = cv2.imread(**'images/1.jpeg'**)  
  
 *# Changing the colour-space  
 # blur = cv2.GaussianBlur(image, (3, 3), 0)* blur = cv2.medianBlur(image, 11)  
 frame = cv2.cvtColor(blur, cv2.COLOR\_BGR2HSV)  
 *# Find edges* bgrl = np.array([42, 0, 0]) *# black* bgrh = np.array([255, 40, 77]) *# black  
  
 # bgrl = np.array([62, 12, 46]) # shaft  
 # bgrh = np.array([250, 255, 255]) # shaft* mask = cv2.inRange(frame, bgrl, bgrh)  
  
 cv2.imshow(**'masked'**, mask)  
 *# Find Contours* contours, \_ = cv2.findContours(mask, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)  
 print(**"Total number of contours : "**, len(contours))  
 count = 0  
 **for** i **in** contours:  
 area = cv2.contourArea(i)  
 *# print(i)  
 # print(area)* **if** area > 350:  
 count += 1  
 cv2.drawContours(image, i, -1,(0, 230, 0),2)  
 x, y, w, h = cv2.boundingRect(i)  
 cv2.rectangle(image,(x, y),(w+x,   
 h+y),(0,255,255),2)  
 print(**"Number of contours with marked rectangle : "**, count)  
  
 cv2.namedWindow(**'Contours'**, cv2.WND\_PROP\_FULLSCREEN)  
 cv2.setWindowProperty(**'Contours'**, cv2.WND\_PROP\_FULLSCREEN, cv2.WINDOW\_FULLSCREEN)  
  
 cv2.imshow(**'Contours'**, image)  
 cv2.waitKey(0)

