15)To verify Otsus image segmentation.

import numpy as np

import cv2

from matplotlib import pyplot as plt

img = cv2.imread(r'/content/A (5).png')

b,g,r = cv2.split(img)

rgb\_img = cv2.merge([r,g,b])

gray = cv2.cvtColor(img,cv2.COLOR\_BGR2GRAY)

ret, thresh = cv2.threshold(gray,0,255,cv2.THRESH\_BINARY\_INV+cv2.THRESH\_OTSU)

# noise removal

kernel = np.ones((2,2),np.uint8)

#opening = cv2.morphologyEx(thresh,cv2.MORPH\_OPEN,kernel, iterations = 2)

closing = cv2.morphologyEx(thresh,cv2.MORPH\_CLOSE,kernel, iterations = 2)

# sure background area

sure\_bg = cv2.dilate(closing,kernel,iterations=3)

# Finding sure foreground area

dist\_transform = cv2.distanceTransform(sure\_bg,cv2.DIST\_L2,3)

# Threshold

ret, sure\_fg = cv2.threshold(dist\_transform,0.1\*dist\_transform.max(),255,0)

# Finding unknown region

sure\_fg = np.uint8(sure\_fg)

unknown = cv2.subtract(sure\_bg,sure\_fg)

# Marker labelling

ret, markers = cv2.connectedComponents(sure\_fg)

# Add one to all labels so that sure background is not 0, but 1

markers = markers+1

# Now, mark the region of unknown with zero

markers[unknown==255] = 0

markers = cv2.watershed(img,markers)

img[markers == -1] = [255,0,0]

plt.subplot(211),plt.imshow(rgb\_img)

plt.title('Input Image'), plt.xticks([]), plt.yticks([])

plt.subplot(212),plt.imshow(thresh, 'hsv')

plt.imsave(r'thresh.png',thresh)

plt.title("Otsu's binary threshold"), plt.xticks([]), plt.yticks([])

plt.tight\_layout()

plt.show()

Result:

