CMPE 362

DIGITAL IMAGE PROCESSING

HOMEWORK 2 REPORT

Instructor: Aslı GENÇTAV
Name - Surname: Neslihan Pelin METİN
Student Number: 71047171244

Introduction

This homework aims to teach us how to detect leaves on an image using K-means clustering and Hough transform. It also aims to test what we have learned in the classes using Python.

Literature Review

◊ K-Means Clustering: It is an image segmentation technique that segments the image according to its colors or textures. In this homework, we have segmented the image with its colors.

◊ Hough Transform: It helps to detect circles through the image. We have used this technique to detect the leaves in the image.

◊ Binary Mask: Shows region of interest using binary values (0 and 1).

Problem Statement & Implementation

This assignment is composed of three parts of the segmentation of images. In the first part, we need to find the segmentation of the image. We need to choose the suitable k value and then compute the binary mask of the segmentation. Afterward, we must compute the distance transform of the binary mask. Secondly, we were asked to find the circles and mark them with red. Finally, using the binary mask we have found before, we will eliminate the circles whose center is not in the boundaries of the binary mask.

I began to implement my code by reading the input image. After that, I determined the k numbers and applied segmentation according to these values. Then, based on the segmentation results, I chose which k value is more appropriate. I found a binary mask following this k value, and I applied the distance transform to this binary mask.

```
pixels = input1.reshape(-1, 3).astype(np.float32)
criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 10, 1.0)

ks = [2,3,4,5,6,7]
segmented_image = [input1, input1, input1, input1, input1, input1]
random_centers = cv2.KMEANS_RANDOM_CENTERS

for i in range(6):
  ret, labels, centers = cv2.kmeans(pixels, ks[i], None, criteria, 10, random_centers)
  segmented_image[i] = labels.reshape(input1.shape[:2])
  plt.imshow(segmented_image[i])
  plt.title("k = %d" %ks[i])
  plt.colorbar()
  plt.show()

ret, labels, centers = cv2.kmeans(pixels, ks[3], None, criteria, 10, random_centers)

plant_label = np.argmax(centers[:, 1])
binary_mask = np.uint8(labels == plant_label)
binary_mask = binary_mask.reshape(input1.shape[:2])

plt.imshow(binary_mask)
plt.title("Binary Mask of the Image (k=4)")
plt.show()
```

Later, I found the maximum value and center value of the distance transform, and concerning them I calculated the minimum and maximum radius values of the circles. I found the La*b*

version of the image and blurred it's a* channel with the median blurring technique. Afterward, I defined parameter values as (40, 10), (80, 10), (80, 15), and used them in the method "HoughCircles" by rotating in a loop. Also, I drew the circles detected with "cv2.circle" method. Lastly, I printed the results one by one.

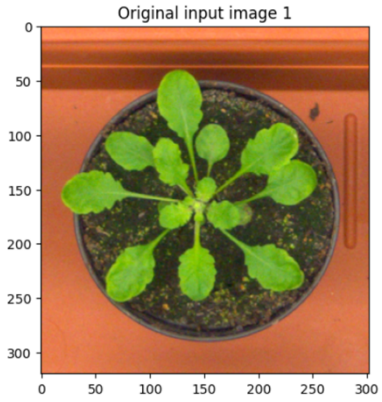
```
max_distance = np.max(dist_transform)
center_y, center_x = 155, 145
center_value = dist_transform[center_y, center_x]

min_radius = int(center_value)
max_radius = int(1.25 * max_distance)

lab_image = cv2.cvtColor(input1, cv2.COLOR_BGR2Lab)
a_channel = lab_image[:, :, 1]
filtered_a = cv2.medianBlur(a_channel, ksize=5)

param_settings = [(40, 10), (80, 10), (80, 15)]

for i, (param1, param2) in enumerate(param_settings):
    circles = cv2.HoughCircles(filtered_a, cv2.HOUGH_GRADIENT, dp=1,
                               minDist=0.1 * min(input1.shape[:2]),
                               param1=param1, param2=param2,
                               minRadius=min_radius, maxRadius=max_radius)

    image_rgb = cv2.cvtColor(input1, cv2.COLOR_BGR2RGB)
    if circles is not None:
        circles = np.uint16(np.around(circles))
        for circle in circles[0, :]:
            center = (circle[0], circle[1])
            radius = circle[2]
            cv2.circle(image_rgb, center, radius, (255, 0, 0), 2)

    plt.imshow(image_rgb)
    plt.title(f'Param1: {param1}, Param2: {param2}')
    plt.show()
```

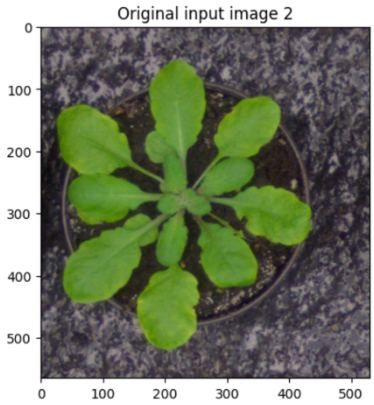
Finally, I used the loop logic I mentioned above to eliminate circles whose center is not located in the binary mask. In addition to that method, I have placed an if statement to check whether its center is located in the binary mask. And that's how I found the desired result and printed it.

```
for i, (param1, param2) in enumerate(param_settings):
    circles = cv2.HoughCircles(filtered_a, cv2.HOUGH_GRADIENT, dp=1,
                               minDist=0.1 * min(input1.shape[:2]),
                               param1=param1, param2=param2,
                               minRadius=min_radius, maxRadius=max_radius)

    image_rgb = cv2.cvtColor(input1, cv2.COLOR_BGR2RGB)
    if circles is not None:
        circles = np.uint16(np.around(circles))
        for circle in circles[0, :]:
            center = (circle[0], circle[1])
            radius = circle[2]
            if binary_mask[center[1], center[0]] != 0:
                cv2.circle(image_rgb, center, radius, (255, 0, 0), 2)

    plt.imshow(image_rgb)
    plt.title(f'Param1: {param1}, Param2: {param2}')
    plt.show()
```

Testing

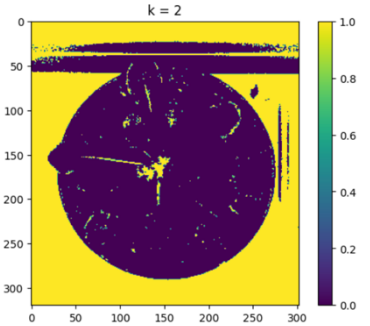
Here I will be testing my code with two input images given in below.

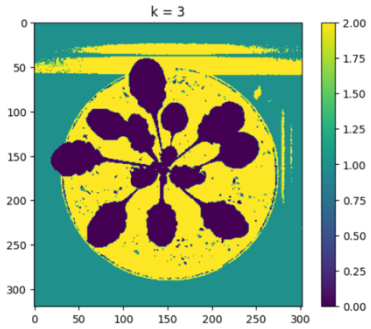


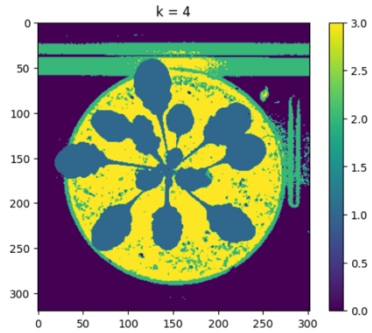


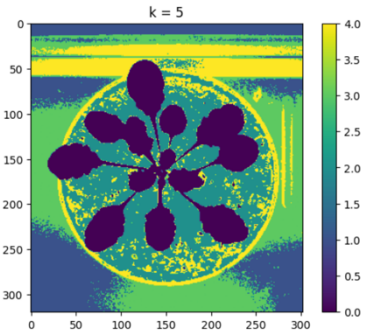
◊ Input 1:

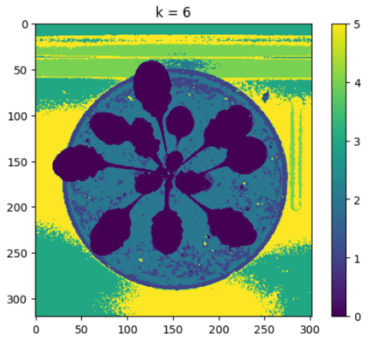
I chose k = 4 because it can segment the leaves better and covers fewer data outside the leaf.

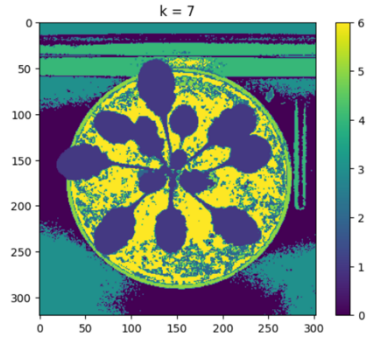


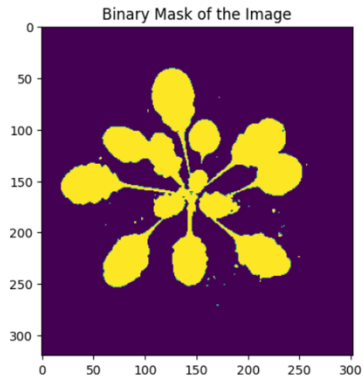


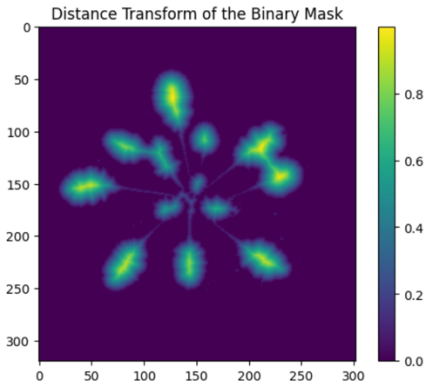




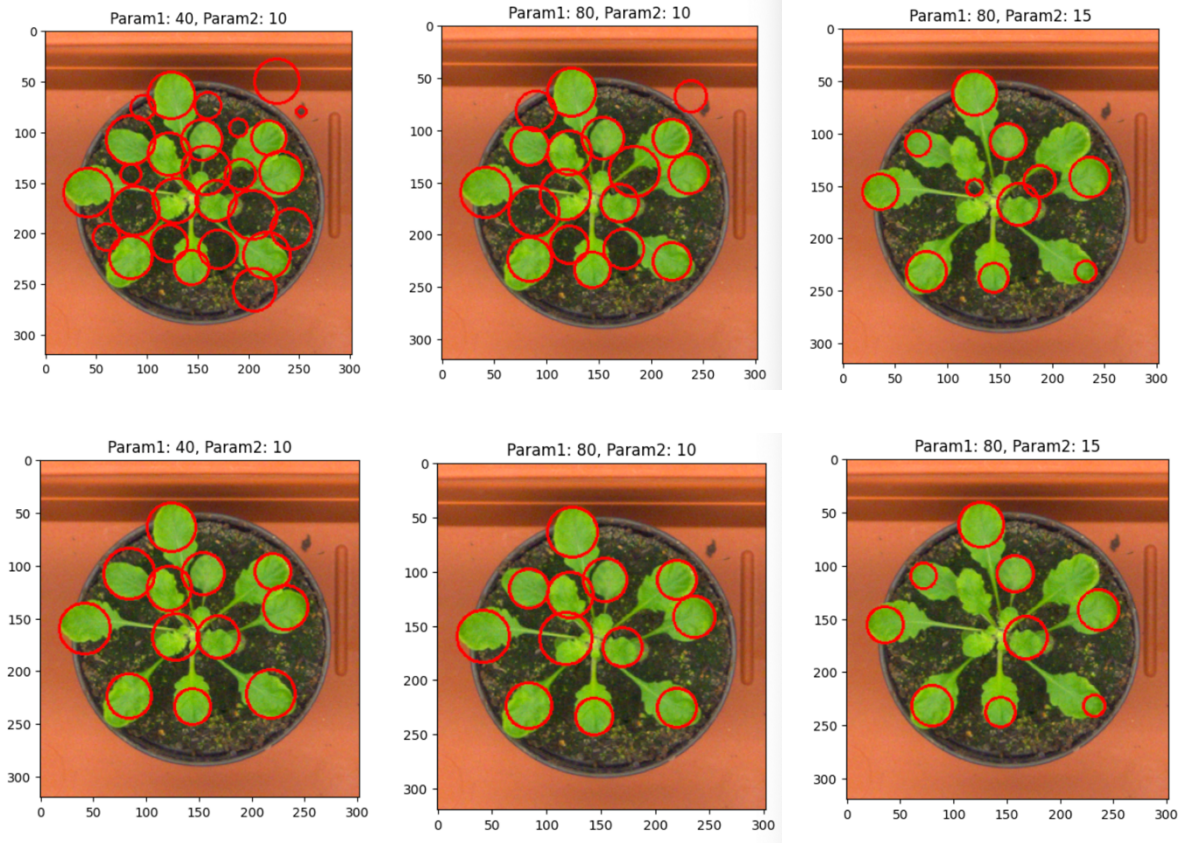






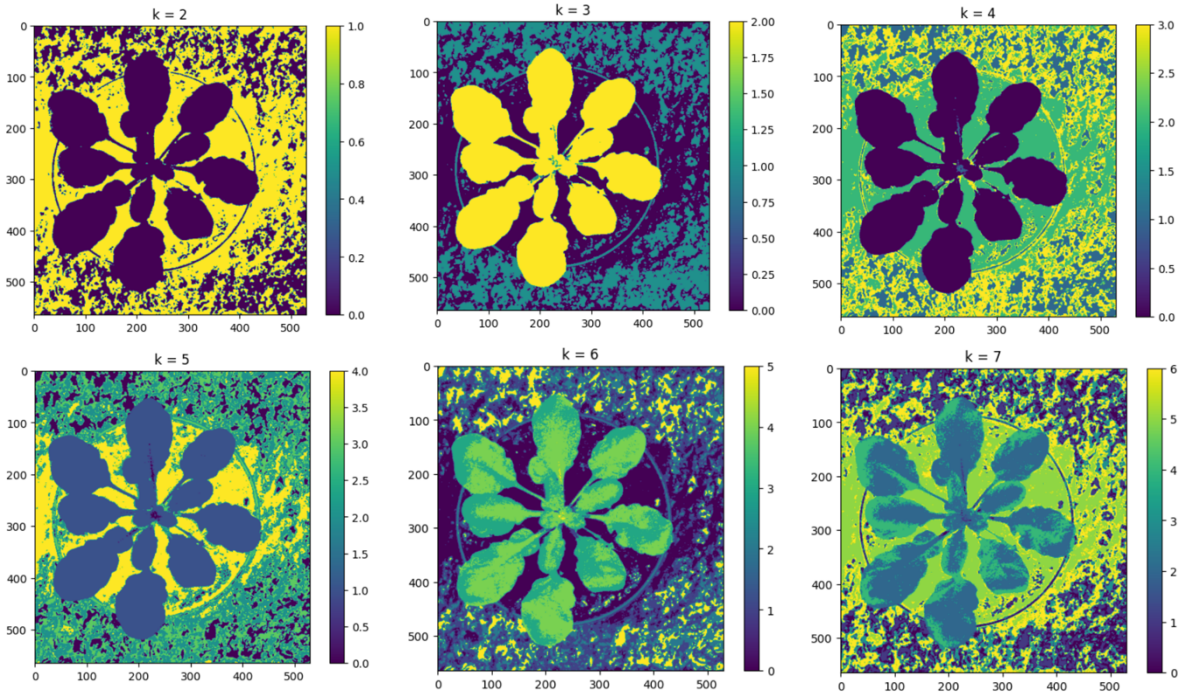


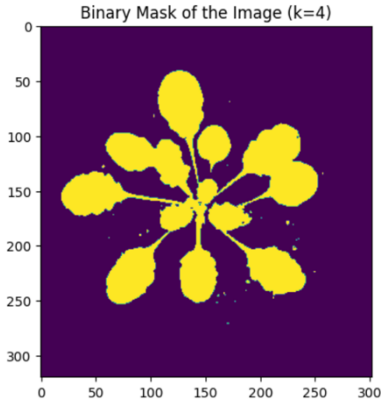
As we can see in the results below, as the parameter 1 value increased, fewer circles were detected, and the detected circles' radius are much larger. Also, as the parameter 2 value decreases, a larger number of circles were detected.

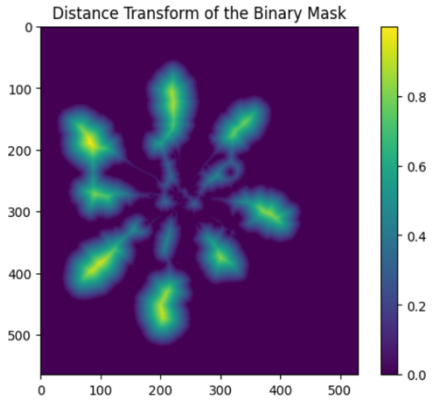


◊ Input 2:

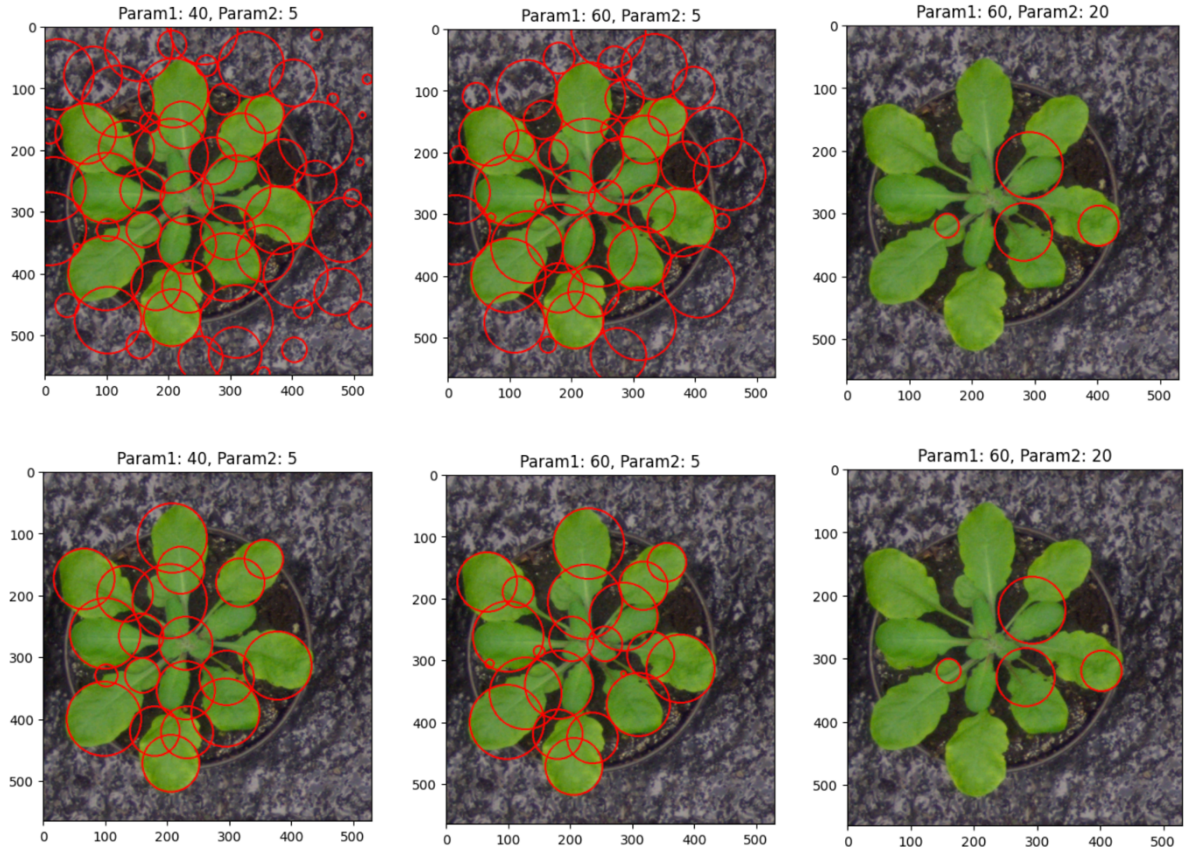
I chose k = 3 because it can segment the leaves better and covers fewer data outside the leaf.







As we can see in the results below, as the parameter 1 value increased, fewer circles were detected, and the detected circles' radius are much larger. Also, as the parameter 2 value decreases, a larger number of circles were detected like in the previous example.



Conclusion

I think this project was easier than the previous ones. The explanation of the question was very clear, and I think I did a good job. I have learned what are K-means and Hough Transform in image processing, how they work, and how we can use them. I hope to enjoy the next projects as much as I enjoyed this one. As a final remark, I hope I can use what I have learned in this project in the future.