Computer Vision project 1

Olaf Duda

January 2023

1 Introduction

In this task, we were asked to create a segmentation for a data set of leaves, in my case potato leaves with late blight [1]. In digital image processing and computer vision, image segmentation is the process of partitioning a digital image into multiple image segments, also known as image regions or image objects. [2]

Our task was to first create a so called "ground truth", which was segmentation created from pictures of leaves without any background. Next part was to create the whole segmentation based on the whole image and later to compare our results from subtask 1 and 2.

2 Subtask 1 - Ground truth

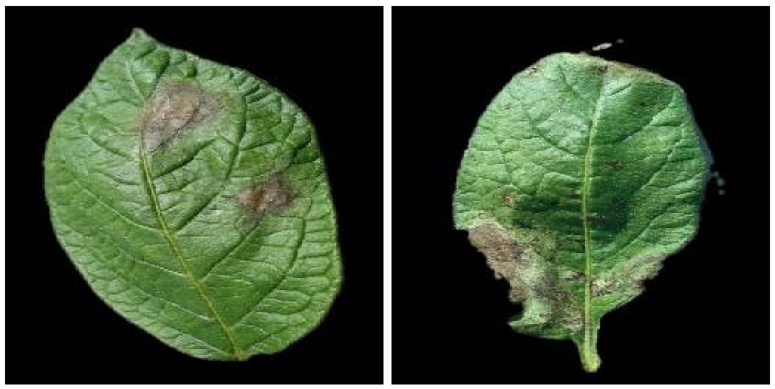

Figure 1: Exemplary pre-segmented leaves

In this part we were creating a mask that would only find a leaf on the pre-segmented picture. To create this mask I needed to basically find any colour that wasn't black. To do so we convert our image to shades of gray, blur it (to be easier to find) and find the threshold.

```
              ————————— Ground truth —————————
img_gry = cv2.cvtColor(img_seg, cv2.COLOR_BGR2GRAY)
img_gry_blr = cv2.GaussianBlur(img_gry, (9, 9), 0)

ret, tsh = cv2.threshold(img_gry_blr, 20,
                         255, cv2.THRESH_BINARY)
```

Where img_seg is a read image such as on Figure 1. And we got the following results

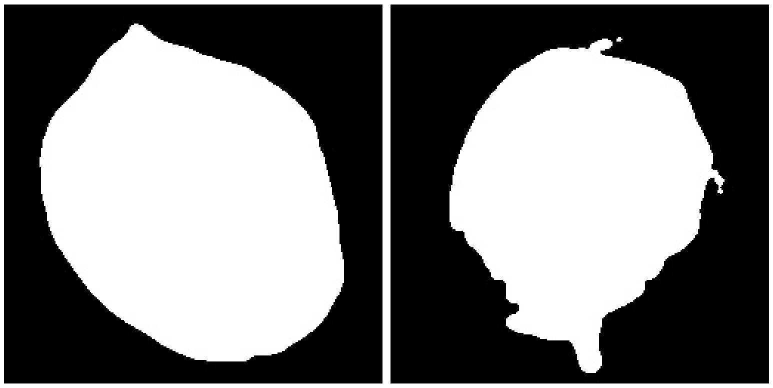

Figure 2: Exemplary ground truth

3 Subtask 2 - My segmentations

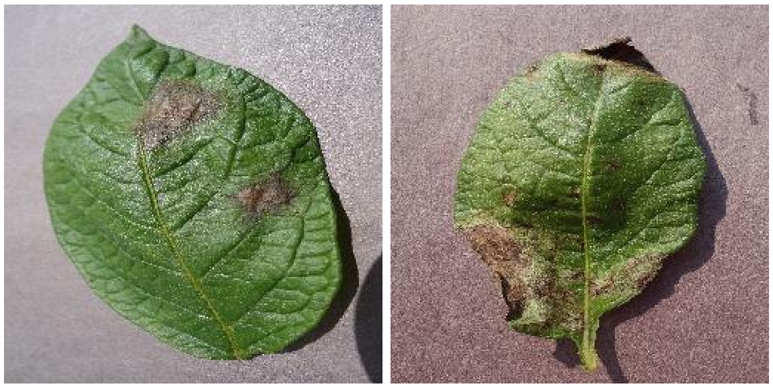

Figure 3: Exemplary full images

Our second task was the hard part. Here we were looking at the full picture and we wanted to find mask to cut out the leaf without the background. My work can be divided into some parts. Before everything I blurred our image, once again, so the colour will be in the similar range, and turn the image to hsv colours. We turn to hsv because it is much easier to find ranges of similar shades of the same colour in this representation.

```
——————— Finding my answer ———————

hsv = cv2.GaussianBlur(img, (9, 9), 0)
hsv = cv2.cvtColor(hsv, cv2.COLOR_BGR2HSV)
```

3.1 The masks

We begin with finding masks of two colours. We want to find the green colours, since, well, it is a leaf but also we need to find the browns that are on the leaves. Those browns that represent sickness and are not on the background. The second mask (the one to find brown colours) was a bit problematic, since some images, like the one on the following Figure, had some really red background which was a problem, since this brownish-red background was similar in colour with some of the sickness:

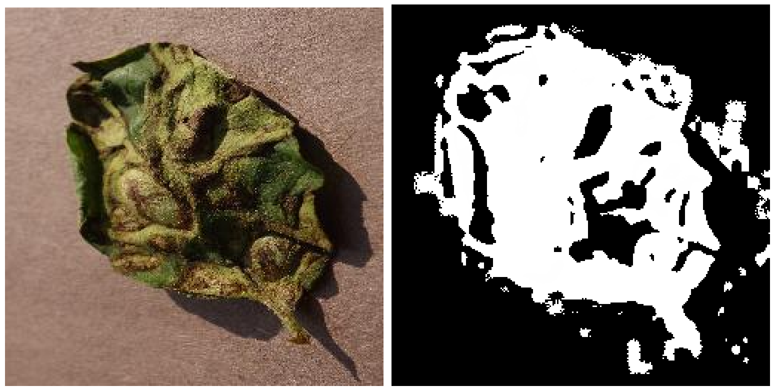

Figure 4: The original image and my mask for it before any further changes to it

To create the masks, I needed to find range of each colour. To do so I went through a lot of trial and error and as tools I used GIMP and converters of colour from RGB to HSV to find some thresholds.

```
        # Finding mask of the leaf
lower_green = np.array([30, 45, 35])
upper_green = np.array([100, 255, 255])

mask_g = cv2.inRange(hsv, lower_green, upper_green)

        # Finding mask of the sickness
lower_brown = np.array([11, 35, 45])
upper_brown = np.array([35, 255, 240])

mask_b = cv2.inRange(hsv, lower_brown, upper_brown)

mask_f = mask_b + mask_g
```

As seen on Figure 4 our masks may have a lot of holes and irregularities outside of the leaf, to remove them we use the so-called Morphology

3.2 Morphology

Morphology is a comprehensive set of image processing operations that process images based on shapes.[3] We use it to remove both some random pixels outside of our chosen image and fill holes inside it.

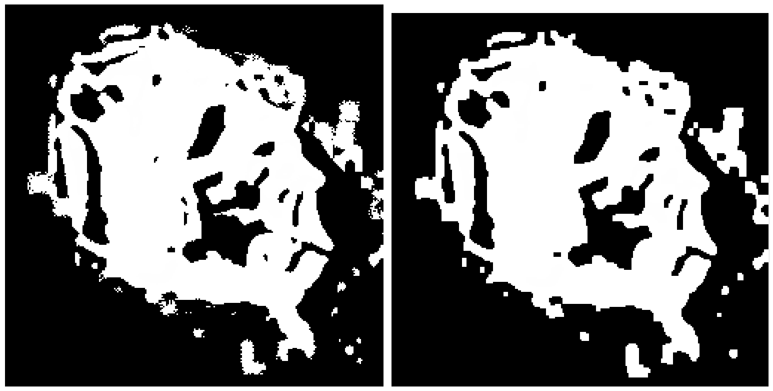

Figure 5: Mask before and after morphology

In my code I used morphology on mask of brown colours, green ones and on the sum of those masks. Where on the sum I used also the opening, not only closing, which means I removed some small dots outside of our leaf.[4]

```
        # Morphology
mask_g = cv2.morphologyEx(mask_g, cv2.MORPH_CLOSE,
                kernel=np.ones((3, 3), dtype=np.uint8))

mask_b = cv2.morphologyEx(mask_b, cv2.MORPH_CLOSE,
                kernel=np.ones((3, 3), dtype=np.uint8))

mask_f = cv2.morphologyEx(mask_b + mask_g,
                cv2.MORPH_CLOSE, kernel=np.ones((3, 3)
                , dtype=np.uint8))
mask_f = cv2.morphologyEx(mask_f, cv2.MORPH_OPEN,
                kernel=np.ones((3, 3), dtype=np.uint8))
```

For my morphology to work properly I needed to choose proper kernel size. Too small kernel would make our morphology not do any changes, where too big size would put some parts of the background outside of the leaf into the mask as well. I found out that a 3x3 matrix was the best size for our images.

3.3 Comparision

Now our masks were done and the last thing to do is to compare them with correlated ground truths. To compare all leaves I use Dice coefficient and Intersection over Union (IoU), to check how much of our mask overlaps with the ground truth. The following equations show how to do that:

$$IoU = \frac{target \cap prediction}{target \cup prediction}$$

$$Dice = \frac{2\,|A \cap B|}{|A| + |B|}$$

In our case $target \cap prediction$ is area of the intersection and $target \cup prediction$ is area of the union of my mask and ground truth. $|A \cap B\,|$ is also area of the intersection of my mask and ground truth and $|A|, |B|$ are area of our mask and ground truth. I calculate both of them for all my masks, to sum them up and calculate mean of IoU and Dice

```
        #Dice Coefficient
dice_coef = np.sum(mask_f & tsh)*2 /
                (np.sum(tsh) + np.sum(mask_f))
sum_coef += dice_coef

        #IoU
iou_step = np.sum(mask_f & tsh) / np.sum(mask_f | tsh)
sum_iou += iou_step
```

```
#Mean of IoU and Dice
print("Dice coefficient:")
print(sum_coef/i)
print("Dice IoU:")
print(sum_iou/i)
```

Which gives the following output:

```
Dice coefficient:
0.958524162316212
Dice IoU:
0.9221251743653083
```

We can see that my results are pretty close to the ground_truth. I give some plausible improvements for my work later in this report.

4 My results

Now I will show some best and worst examples of my segmentations. I compared my masks with respect to their Dice coefficient.

4.1 Best segmentations

In the top 5 best segmentations we can see that those were pretty simple leaves, with a similar background and very good lighting. My code didn't catch any big shadows as a part of the mask, which is very good, only in some places it chose small parts of shadows but it is not very visbile at the first glance.

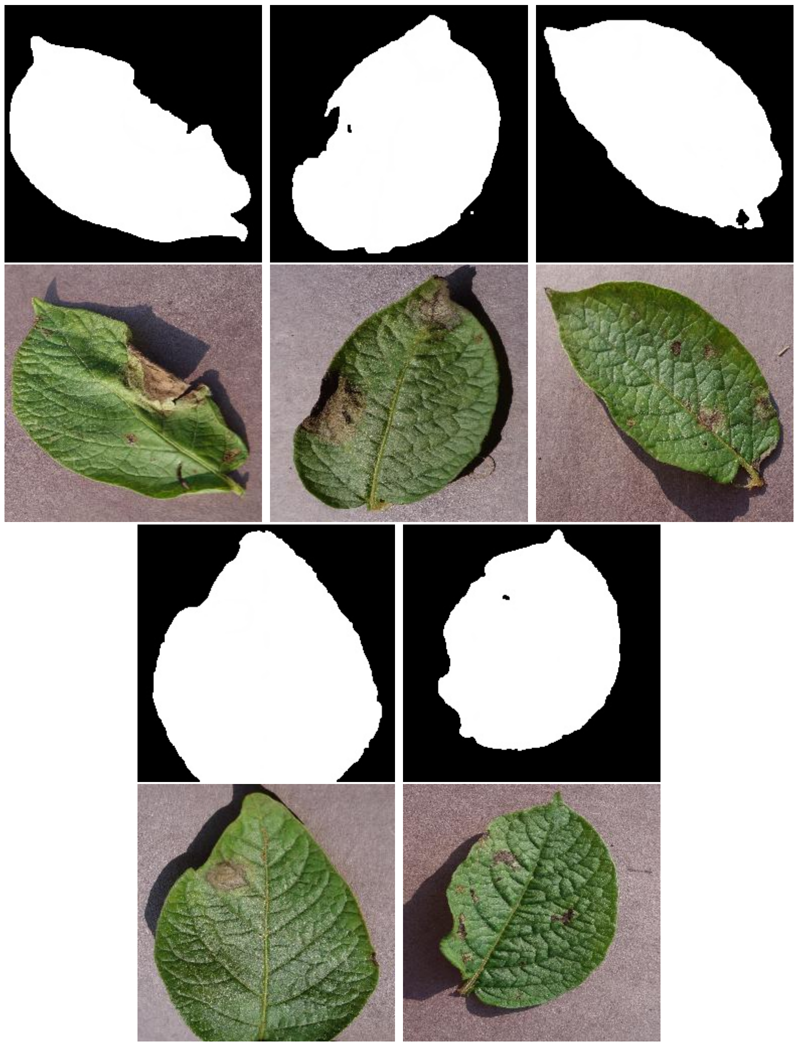
Figure 6: Top 5 segmentations , with given Dice coefficients: 99,1%, 99,1%, 99,1%, 99,2%, 99,2%

4.2 Worst segmentations

In my worst results we can see that mostly on those pictures there is a very specific lighting. For. example in the last image, as mentioned before,we see this warm brownish-red background which is very similar to sickness, hence it might be also chosen as part of the leaf. In our first image we can see that the lighting is very white and a bit blurred. What I assume happened is that when in my code I blur once again the colours on the leaf get gray like the background behind the leaf, hence there are so big holes in it.

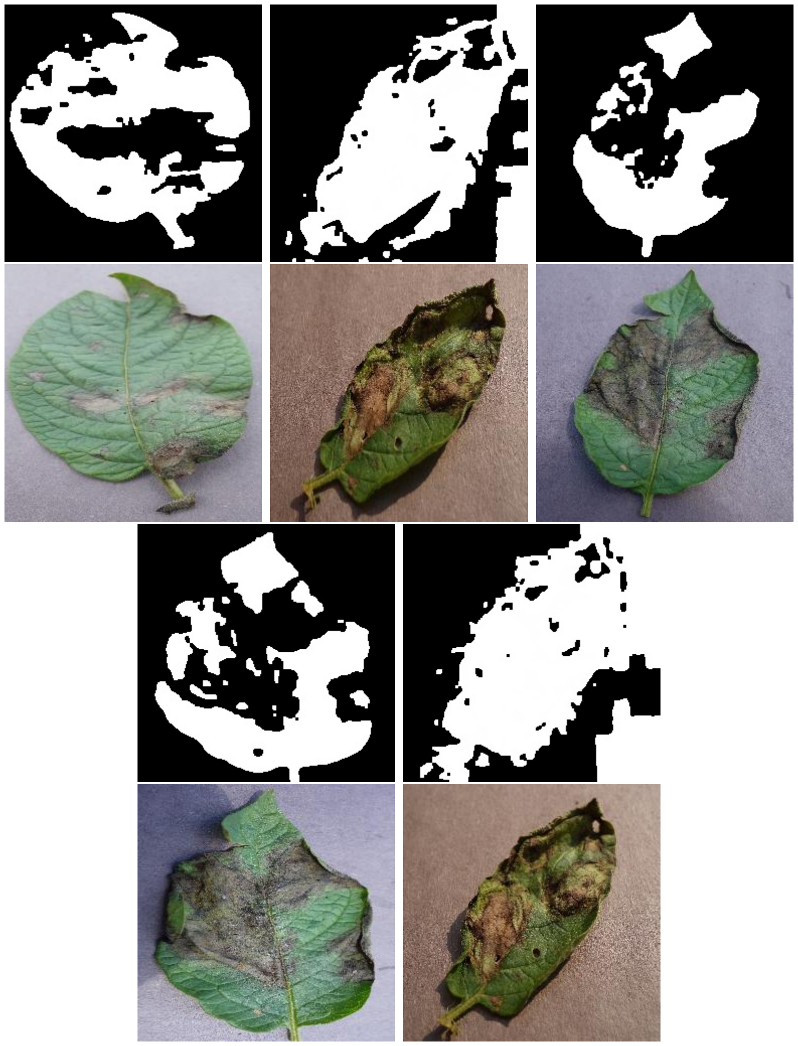

Figure 7: Bottom down worst segmentations, in order from the best to worst, with given Dice coefficients: 80,1%, 78,2%, 77,6%, 72,8%, 71,4%

4.3 Plausible improvements

I found out that method with 2 masks was very effective, but I suppose it would be possible to omit those brownish background by dividing our mask for detecting browns into two different masks.

But what I think would make the biggest change is to add enlargement of saturation at the beginning. This way our colours would differ more, so our brownish background for example, would be more reddish which would make it easier to not include them in our masks.

Obviously we could always increase kernel of our morphology, however it would make our final results ”squary” and as I checked with some small changes the mean result was overall better than mine by 0.2% but in worst and best cases it was approximetely 0.1% worse.

5 Remark

At the beginning of my work I removed some segmentations and images from the database that were really badly segmented to not include them in my Ground Truth.

References

[1] PlantVillage Dataset of potato leaves with late blight https://www.kaggle.com/datasets/abdallahalidev/plantvillage-dataset

[2] Image segmentation https://en.wikipedia.org/wiki/Image_segmentation

[3] P Soille. “Morphological Image Analysis, Principles and Applications”, 1999.

[4] Types of morphology https://docs.opencv.org/4.x/d9/d61/tutorial_py_morphological_ops.html