Practical Exercises: GreenEyes 2

Iuliia Alekseenko, ajulyav@gmail.com

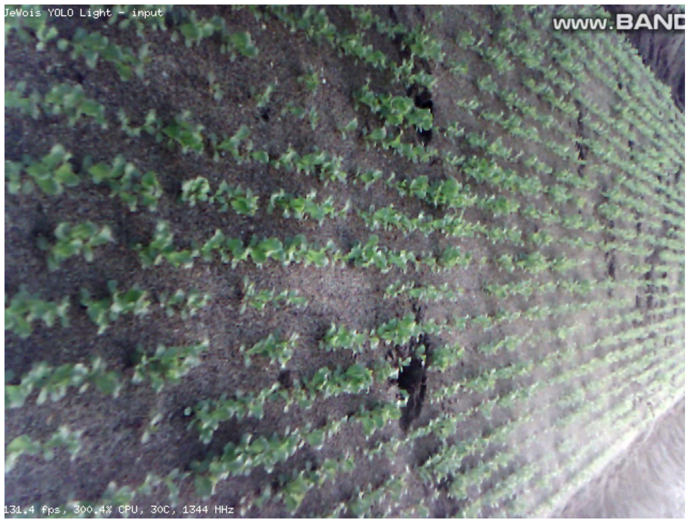

JeVois YOLO Light - input
www.BAND
131.4 fps, 300.4% CPU, 30C, 1344 MHz

Fig. 1: salads.png for image processing

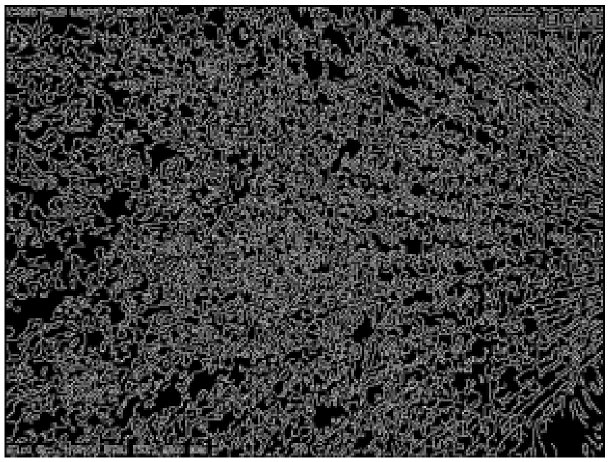


Fig. 2: Canny edge-detector applied on the salads.png file

I. Image Processing

A. Segmentation

The exercise consist of two problems - image segmentation and transformation. The provided image is presented in Figure 1.

When choosing a segmentation algorithm, a number of factors were taken into account, namely, the time allotted for solving the problem and the file itself. As we can see, the image quality is not good, as there are overexposures, and the resolution of the camera is not too high. Both of these factors will directly affect the quality of segmentation, since, for example, the contours of a plant are not clear even when they are located near the camera. In the area of the bright daylight spot (the lower right corner of the image) it is a rather difficult task, since it is simply not visible where the background is and where the foreground object is.

There are a large number of proposed methods for solving the segmentation problem. When evaluating the file and scene, it is possible to assume that segmentation methods based on object edges are unlikely to perform well (even if noise reduction techniques or smoothing are used). The point is that the background is the soil, which is represented by a lot of structure. For example, the edges using the Canny edge-detector are shown in Figure 2. Ideally, these segmentation techniques work with a more uniform background. Moreover, upon closer examination of the image, we will notice the presence of small areas of plants on the ground (leaves or grass).

Probably, according to approximately the same logic, algorithms such as egion-based, for example, the well-known the watershed transform, will not be used.

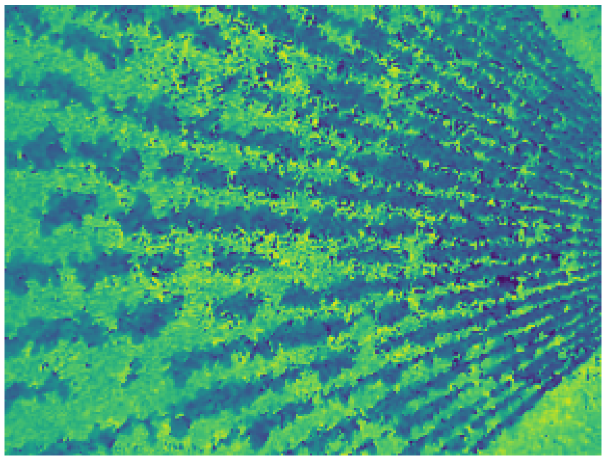


Fig. 3: H channel of the salads.png file

The proposed solution is a simple color-based segmentation method in the HSV color space. Although the colors of the plants and the earth are quite similar in some areas, more of the H channel allows us to distinguish the background from the target object as shown in Figure 3 (the darker green is the foreground, the lighter green is the background.)

Possible improvements:

- image pre-processing, for example increasing the saturation to make the green of the plant more distinguishable from the gray-brown earth, for example Figure 4 shows the H channel of the same image but multiplied by the factor of 1.7 to change the saturation and by the factor of 0.6 to reduce the

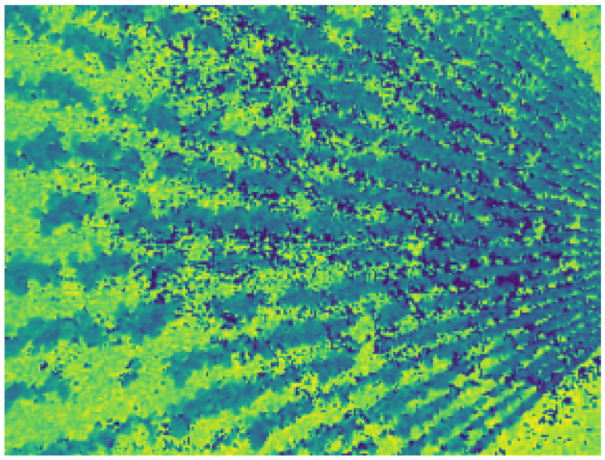


Fig. 4: H channel of the salads.png file

brightness. Here, the hue of the background and the object are more clearly visible.

- also, the algorithm of detecting overexposure and balance it will help to produce segmentation in conditions of excessive lighting.

The OpenCV *inRange()* method takes three parameters: the image, the lower range, and the higher range. It returns a binary mask (an array of 1s and 0s) the size of the image where values of 1 indicate values within the range, and zero values indicate values outside.

Later, to impose the mask on top of the image, the *bitwise_and()* function is used, which keeps every pixel in the given image if the corresponding value in the mask is 1.

In order to find the optimal values and then hard code them in the algorithm, a simple graphical interface was created with the sliders for defining each H, S and V values as shown in Figure 5.

Thus, the found values are low = [H = 9, S = 3, V = 77] high = [H = 107, S = 255, V= 255]. The segmentation result is presented in Figure 6.

B. Transformation

Parallel lines appear to converge on images due to perspective. In order to keep parallel lines parallel for photogrammetry a bird's eye view transformation can be applied.

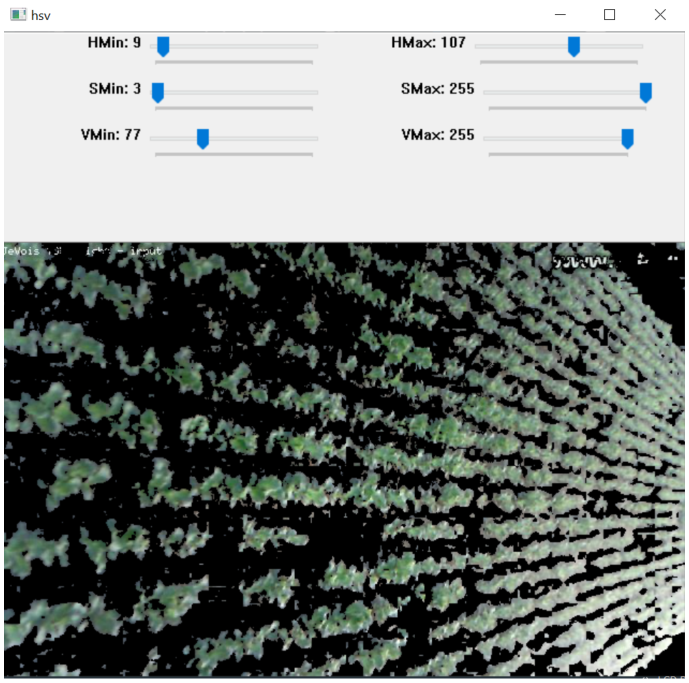
It was decided to transform a part of the original image as it is accepted according to the exercise description. In particular, Figure 7 shows which part was selected.

It is possible to transform the image into Bird's Eye View with two different approaches:

- stretch the top row of pixels while keeping the bottom row unchanged (Figure 8);

- shrinking the bottom of the image while keeping the top row unchanged (Figure 9).

The first variant is more obvious. However, it increases the spatial resolution of the far part of the image (no information is added), and may cause line boundary erosion, so the gradient algorithm may be difficult to detect.



hsv

HMin: 9 HMax: 107

SMin: 3 SMax: 255

VMin: 77 VMax: 255

Fig. 5: Simple graphical interface for finding optimal H, S, V values

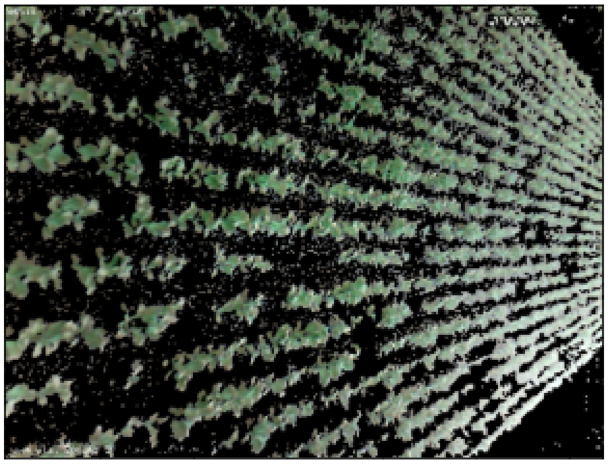


Fig. 6: Segmentation result

The second method may look better because it preserves all the available pixels on the top edge of the original image with a relatively low resolution.

Possible improvements:

- it is possible to carry out tests using more complex algorithms. For example, with a convolutional neural network (CNN) as proposed in [1].

II. Data Generation

In this part, only a few short comments will be given, as well as ideas for improvement.

- Since the task was set to generate from 0 to 4 objects in the background, then in the case of generating 0 objects

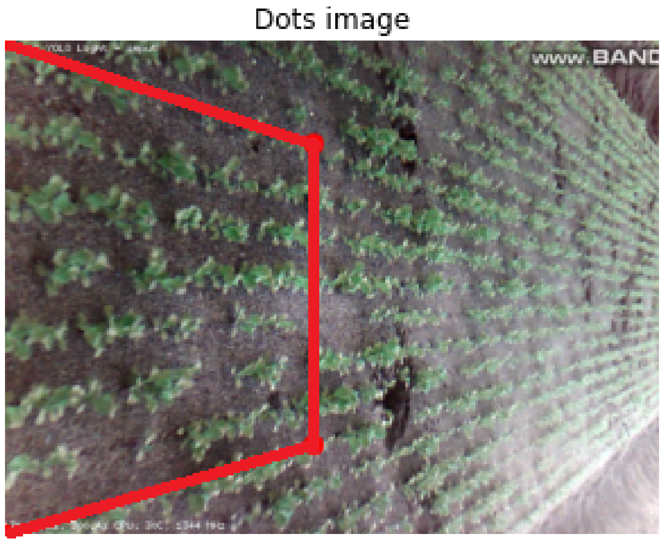


Fig. 7: Portion of image to be transformed

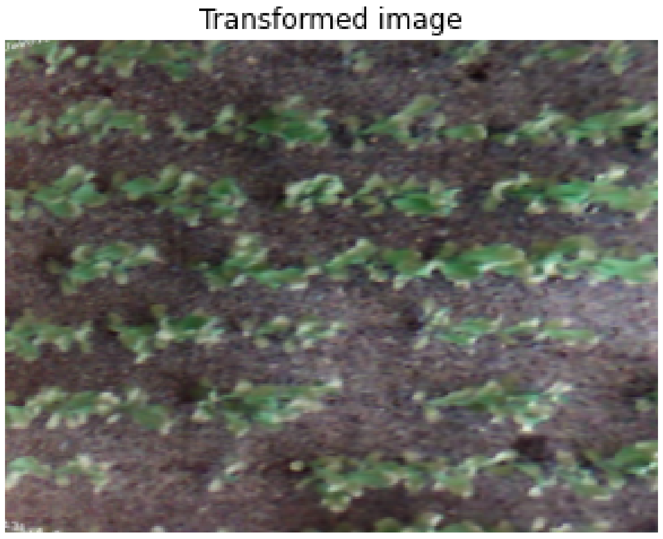


Fig. 8: Stretching the top row of pixels

(no object in the background), the annotation is still written to the JSON file with the bbox equals to (0, 0, 0, 0). Although visually examining the annotations of the original COCO dataset, no such instances were found. So, the other potential solution may be needed.

- Currently, annotations contain only ID and bbox according to the task, however, as part of the improvement, a complete annotation file can be generated.
- Currently, the code only supports transforming a foreground object in the form of its rotation from 0 to 359 degrees. However, other options for transformations can also be considered, for example, scale.
- Perhaps the best cropping of an object from its background can be implemented.

Bboxes have been checked for their correct data format, as shown in the Figure 10.

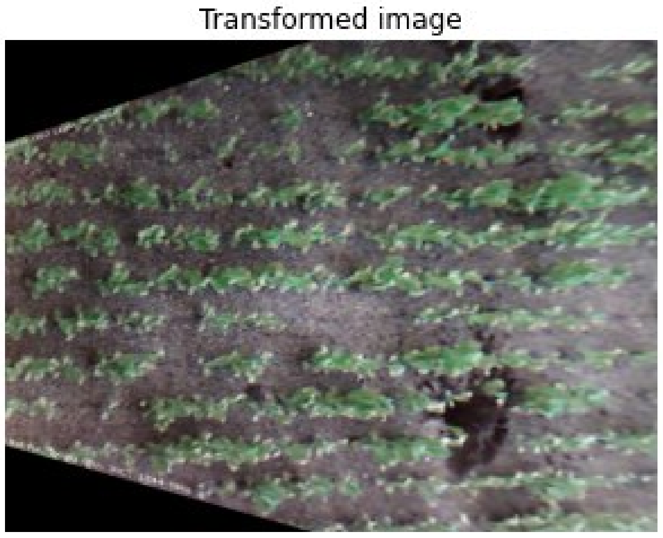


Fig. 9: Shrinking the bottom of the image

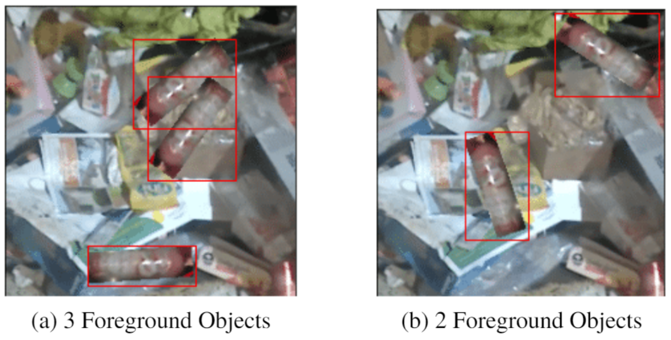


Fig. 10: Bboxes around foreground objects

III. Machine Learning

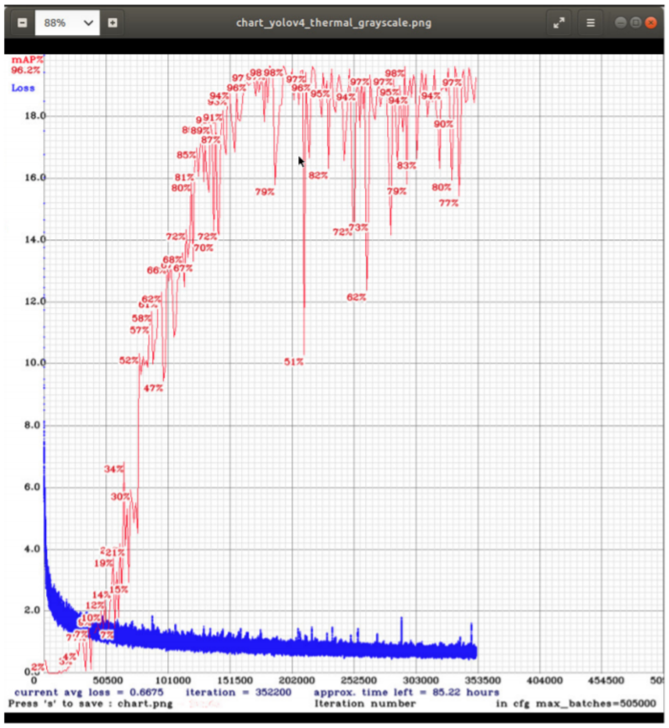
The plot shown in Figure 11 is given to analyze the behavior of the Yolo v4 model on the test data, in particular, some kind of decrease in its performance.

Without additional data about the training samples and training procedure, a number of hypotheses can be made.

It is important to notice a large number of drops on the map chart, it usually means that the mean precision is lower for certain mini-batch as compared to other mini-batches. Perhaps this is due to the fact that the training data is not balanced, i.e. some class dominates in the train dataset. When the drop occurs, it is possible to assume that the mini-batch contains the alternative class, thus, the network does not train enough on it.

From this it can follow that during testing, that non-dominant dataset now makes up the largest part, at the same time, the model performance might be significantly lower.

Ideally, for each object to detect - there must be at least one similar object in the training dataset with about the same: shape, side of object, relative size, angle of rotation, tilt, illumination. Thus, it is important to build a training dataset with objects at different: scales, rotations, lighting, from different sides, on different backgrounds.



```
current avg loss = 0.6675    iteration = 352200    approx. time left = 85.22 hours
Press 's' to save : chart.png                 Iteration number          in cfg max_batches=505000
```

Fig. 11: Yolo v4 training

Additionally, the training dataset should have as many images of negative samples as there are images with objects.

References

[1] Zisserman, Andrew. (2019). A Geometric Approach to Obtain a Bird's Eye View From an Image. 4095-4104. 10.1109/ICCVW.2019.00504.