Using Fully Convolutional Networks for Rumex Obtusifolius segmentation and detection

Schori Damian
Vertiefungsarbeit 1, MSE
ILT, Hochschule für Technik Rapperswil

damian.schori@hsr.ch

***Abstract* – Segmentation and Detection of specific plants is an important task in precision farming. Several influences such as changing light, varying arrangement of leaves and similarly looking plants are challenging. We present a solution for segmenting and detecting individual Rumex obtusifolius out of 2D-images of complicated natural scenes in grassland. We are making use of a fully convolutional deep neural network (FCN) trained with hand labeled images. The proposed segmentation and detection scheme is validated with images taken under outdoor conditions.**

The overall masks segmentation rate is 83.9% measured by the dice coefficient whereas the root detection shows a accuracy of 84.4% considering a maximum of 6 plants per image. The developed solution is therefore a robust method to segment and detect Rumex obtusifolius plants under real-world conditions in short time.

***Keywords* - Deep Learning, Image Segmentation, Computer Vision, Rumex obtusifolius**

I. INTRODUCTION

The detection of specific plants and their corresponding leaves is one of the essential tasks for the automation of plant specific treatments. While the research on organic treatment of Rumex obtusifolius has constantly been improved, the detection of R. obtusifolius stalled.

A. Previous Work

Two solution paths have been researched. The major path is based on a variety of image analysis approaches in 2D including Fourier Analysis and Sliding Window Classifier [1, 2, 3, 4]. The second solution approach focuses on the 3D point clouds analysis and had promising results [5, 6, 7]. A comprehensive summary of current state of the art detection solutions is explained by Binch and Fox [8].

Due to recent advances in Deep Learning, exploration of new processing procedures became far less time consuming. In this work we present promising results towards the solution of real-time plant segmentation and detection in grassland using deep learning. The results originate from a student project, WeedEraser: An autonomous ground vehicle for weed detection and treatment in cooperation with Agroscope, Swiss Federal Office for Agriculture.

II. PROBLEM DESCRIPTION

The segmentation and detection of individual plants in their natural environment is an extremely challenging task due to plants showing significantly varying poses, sizes and complex shapes in natural environmental conditions.

R. obtusifolius differs clearly from most weed in grassland by its shape and size. However, there are plants like Plantago lanceolata or Taraxacum officinale, which show significant similarities in shape and size that influence the detection process. Furthermore, the habitus of R. obtusifolius varies extremely in function of its natural environment and appears in different sizes and constellations in a field. Additional factors such as changing light and field properties influence the detection.

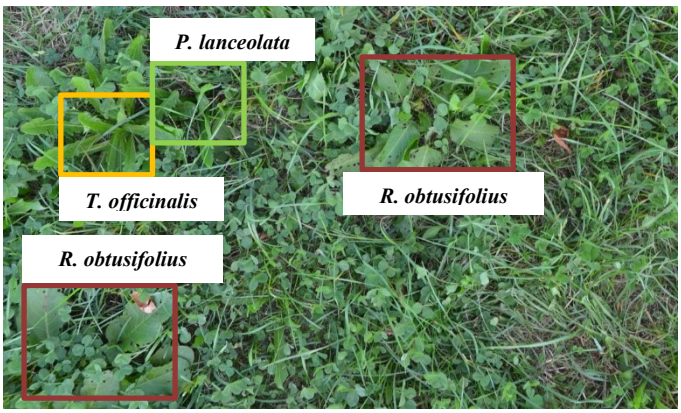

***Fig. 1.** Detail of a training image containing similar plants in size and shape to R. obtusifolius.*

Therefore, a major goal is not only to identify the leaves and corresponding roots but exclude similar species reliably and

minimize the treatment time. The treatment of the plant is purely organic: The hot water jet (80 °C, 120– 130 bar) penetrates the ground up to 15cm and heats the plant root area above 60 °C. Detailed treatment description can be found in [9].

First robotic systems for R. obtusifolius detection and treatment were realized more than ten years ago and the research lasts till today (2019) [1, 2, 3, 10, 11]. A major issue of these systems are low detection rates due to suboptimal segmentation/ classification algorithms.

III. Image Data

A. Training Data

The data were kindly provided by Agroscope. Field images gained on several pastures in Switzerland containing R. obtusifolius were provided as RGB Image Files with different resolutions but mostly similar ratios. In order to be able to work with uniform data, the images were all resized to 1024x1536 pixels.

The images were taken by hand with a Camera from about one meter above ground. This resulted in a recording area of approx. 1.2 x 1.8m. For training, the images were resized to 512x768 pixels. Higher resolutions didn't improve accuracy and resulted in longer training times.

The training and validation dataset consist of 507 images. For the leaf segmentation a hand-labeled binary mask was already provided by Agroscope.

For the root estimation a new hand labeled mask placed at the center of each root was created. For the labelling of the masks, the following points have proven to be decisive:

- The center of the mask must match the center of the root.
- There must be a sufficiently large area of leaves in the mask.
- With large plants it is sufficient if only a part of the plant lies in the mask.

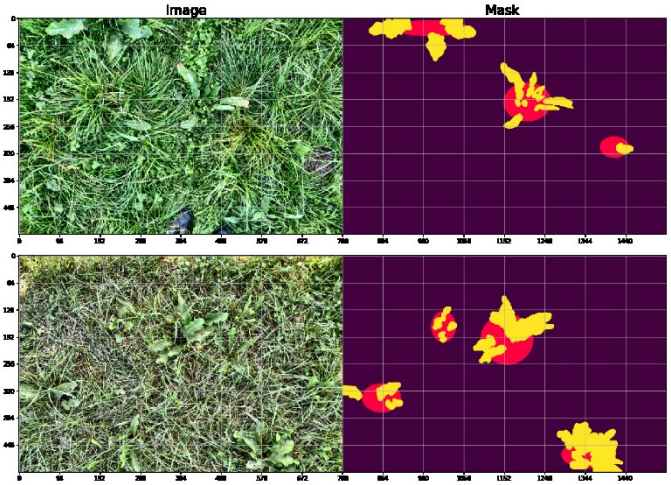

Fig. 2. *Samples of the dataset: field images and ground truth leaf segmentation (yellow) and root estimation (red) masks.*

B. Analysis

Additionally to the masked leaves, the position of the root of each plant was also marked. This made it possible to perform some statistics concerning the number and position of individual plants per image.

If the images are sorted by number of plants per image, the following statistics result:

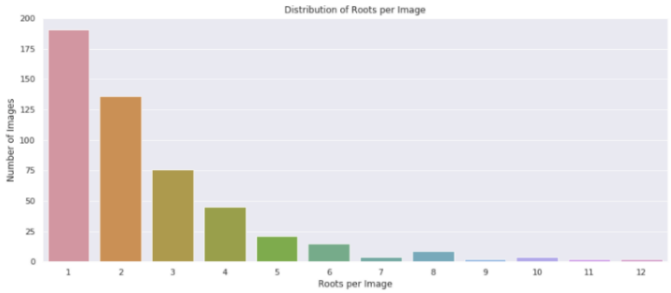

Fig. 3. *Number of individual plants (roots) per image*

There are images containing up to 12 individual plants per image. However, almost 80% of all images containing 3 or less plants per image.

If we look at the spatial distribution of each individual root center, we get the following image (blue points are training and orange points are validation set):

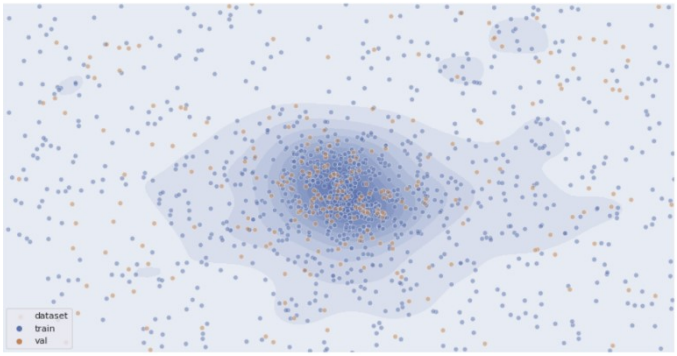

Fig. 4. *Spatial distribution of individual plants of the whole dataset*

Because the pictures were taken by hand, the plants were probably (unconsciously) centered to the camera. Therefore, a majority of the plants are in the center of the images.

IV. Why Deep Neural Networks

A. The Advantages of Deep Learning

One of the main advantages of very deep networks is the automated feature extraction. Previous machine-learning proceedings such as shallow neural networks or support vector machines (SVMs) required a manual feature extraction of the important features. Especially when dealing with high dimensional data it is required to extract important features and throw away unnecessary ones to break down the feature space [12].

With deep neural networks, however this step is completely automated. Features are learned entirely in one training

procedure and do not have to be extracted manually. With this it is possible to feed "raw" data into the network.

Two essential properties characterize the learning procedure in deep learning:

- A tree of hierarchical representations is learned.
- The tree of representations is learned at once.

B. *Convolutional Networks*

In our case of 2D image data, especially Convolutional Neural Networks (CNNs) have proven to be state of the art when it comes to the point to image analysis. CNNs convolve two dimensional input images with kernel matrices which are learned during the training process. This allows the network to learn local patterns, which start mostly with low level features like edges and end in our case with whole leaves of the Rumex Plant. This behaviour gives the network two important properties:

- The learned patterns are translation invariant. Because the kernel is shifted over the whole image, a learned pattern is recognized everywhere in the image. This is very helpful in this case because a plant can appear anywhere in the picture.
- CNN's are learning spatial hierarchies of a pattern. Usually the first layers are learning simple low level patterns such as edges, corners or angles. Deeper layers start to learn more complex features such as leaves in this case. With this, CNNs can very efficiently learn increasingly complex and abstract visual concepts.

C. *Environmental influences*

Another advantage of deep neural networks is the ability to generalize to almost any data. So, if we want the network to make predictions on a data from a specific distribution we "just" have to train the network on this type of data. Therefore we are able to address environmental influences mentioned in Chapter "II Problem description" by sampling and training on exactly this situations. The network is then able to make predictions on new data taken under similar conditions.

Since it is usually difficult to acquire and label enough data for all required situations, there is the possibility of data augmentation. This can help to fill data gaps and to enlarge the data set in general.

V. Model Architecture

A. *Model Architecture*

The proposed plant segmentation and root detection scheme is mainly making use of an adapted version of U-Net (Fig. 5), a Fully Convolutional Neural Network (FCNs) proposed in [13]. FCNs were introduced [14, 13, 15] in the literature as a natural extension of CNNs to tackle per pixel prediction problems such as semantic image segmentation. FCNs add up-sampling layers to standard CNNs to recover the spatial resolution of the input at the output layer. Consequently, FCNs can process images of arbitrary size. In order to compensate the resolution loss induced by pooling layers, FCNs introduce skip connections between their down-sampling (encoder) and up-sampling (decoder) paths. Skip connections help the up-sampling path recover fine-grained information from the down-sampling layers.

The proposed network features an encoder path with six resolution levels, which reduce the spatial dimensions of the input image from 512x768 to 16x24 pixels at the lowest resolution and encode the relevant context and information into 1024-dimensional feature vectors. The decoding path is used to enable precise localization of the leaves using upsampling layers.

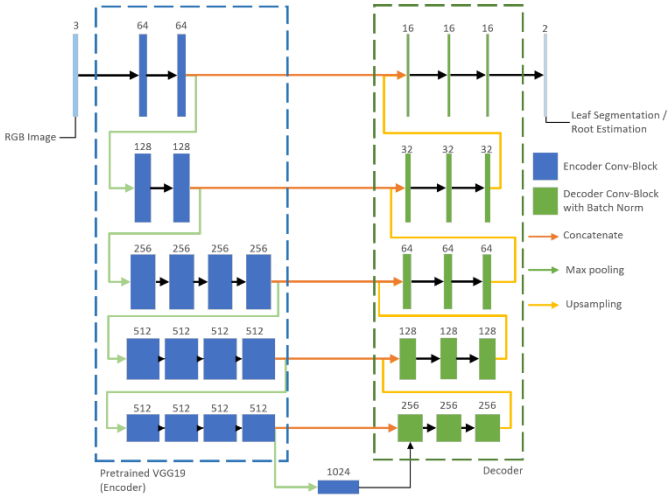

***Fig. 5**. Proposed Network Architecture based on Unet [13]*

B. *Training*

For better and faster convergence, a VGG19 Network [16] pretrained on ImageNet [17] was used as the encoder part. The ImageNet Dataset is only partly similar to our Rumex- Dataset but one can still benefit from the following advantages:

- Sensible initialized weights.
- The first layer tends to learn often similar low level features like edges and corners.
- Avoids overfitting with small datasets.

The network was trained in the Keras-Framework [18] with a TensorFlow [19] backend until convergence (50 epochs, leaving the pretrained encoder part frozen for the first 2 epochs) using the ADAM solver [20] with the first momentum set to 0.9 and the second to 0.99 and a learning rate decay of 0.1 after every 20 epochs.

The loss function is defined as a combination of binary cross entropy and the dice coefficient. It is calculated by a pixel-wise sigmoid activation over the output-feature map. The loss-function is there for defined as:

$$H = H_{BCE} + H_{Dice}$$

where:

$$H_{BCE} = -\sum_{i=0}^{M} (y_i \cdot \log(\hat{y}_i) + (1 - y_i) \cdot \log(1 - \hat{y}_i))$$

$$H_{Dice} = -\frac{2 \cdot \sum_{i=0}^{M} (\hat{y}_i \cdot y_i)}{\sum_{i=0}^{M} (\hat{y}_i + y_i)}$$

- Let $i = \{0 \,..\, M\}$ denote the indices of the flatted vectors of y and $\hat{y}$.

A single Nvidia GTX 1070 GPU was used for the training of the described model. The full training and prediction cycle took about four hours.

C. Histogram Equalization

On each input image a "Contrast Limited Adaptive Histogramm Equalization" (CLAHE) and then "LinearContrast" [21] was applied.

D. Data Augmentation

The network has a large number of trainable parameters (about 31 million) compared to the amount of available training images. To teach the network the desired invariance and robustness properties we made use of real time data augmentation [21] during training time.

In the case of field-images primarily shift and rotation invariance as well as robustness to light, size and bluring variations is needed. The number of augmentation operations applied is chosen randomly from zero to four operations per image.

E. Trained Models

With the described procedure, four Models were trained:

Model	***Description***
Model 0	**Model trained of 80% of the data and validated on 20%. Two output maps, one for leaf segmentation and one for root estimation**
Model 0b	**Model trained of 20% of the data and validated on 80%. Two output maps, one for leaf segmentation and one for root estimation**
Model 1	**Model trained of 80% of the data and validated on 20%. One output map for leaf estimation**
Model 2	**Model trained of 80% of the data and validated on 20%. One output map for root estimation**

Table 1*. Trained Models*

F. Training Visualization

During training the corresponding loss and dice coefficient was logged and plotted:

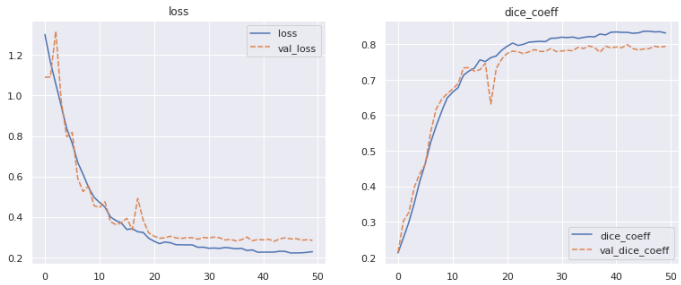

Fig. 6. *Training history of Model 0*

From this the following can be interpreted:

- The loss and dice coefficient are constantly decreasing and rising respectively. The validation curves then seem to converge after about 30 epochs. This generally indicates a good "training behavior".
- The gap between the training and validation curve is rather small which indicates that there is no overfitting on the training data.
- The jumps in the validation curves in the first 20 epochs probably result from the rather small validation set. However, these smooth out in the later course. This is also due to data augmentation which "increases" the training set.

VI. Visualizing Intermediate Activations

To get an understanding of what the network learns one can plot the intermediate activations given a certain input image. With that it is possible to receive the outputs of intermediate convolution and other layers. This gives a view into how an input is decomposed into the different filters learned by the network [12].

Fig. 7. *Input Image*

A. Intermediate Activations

Analyzing the intermediate Activations given the input image in Fig. 7 one can interpret the following (more layers in 40_layer_visualization.ipynb visible):

- The first layers are retaining a major part of the input image and keep much information of the image. Mostly lower-level structures such as edges, corners and angles are becoming visible:

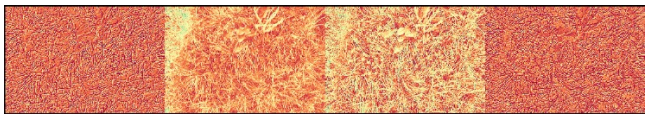

Fig. 8. *4 of 64 intermediate Activations from Layer 1 of 58*

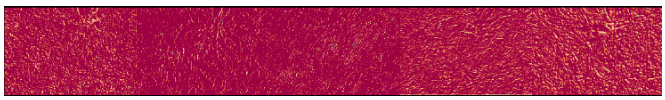

Fig. 9. *5 of 128 intermediate Activations from Layer 5 of 58*

- As the network gets deeper, the activations become more abstract and less visually interpretable. The network is detecting more higher-level features such as the coarse leaf structure:

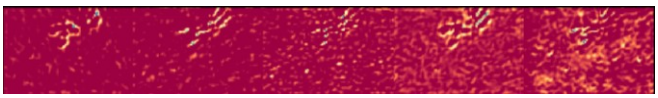

Fig. 10. *5 of 512 intermediate Activations from Layer 15 of 58*

- Reaching the bottleneck of the Network the layers increasingly carry less information about the visual contents of the image but rather more information related to what and where such as shape and position of the whole plant:

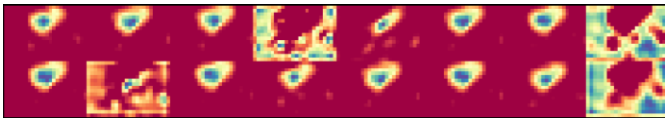

Fig. 11. *16 of 1024 intermediate Activations from Layer 25 of 58*

- On the up-sampling part, the layers are combined through skip connections with the down-sampling path and are thus recovering fine-grained information. With this the network is able to create more and more a pixel-wise prediction heatmap:

Fig. 12. *5 of 512 intermediate Activations from Layer 37 of 58*

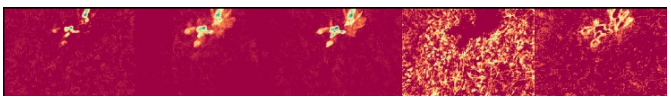

Fig. 13. *5 of 128 intermediate Activations from Layer 49 of 58*

- Finally the segmentation mask is created after a single Convolution followed by Sigmoid Activation to keep the Pixelvalues in 0..1 Range:

Fig. 14. *Output Layer*

VII. Results Leaf Segmentation

A. Evaluation Metrics

The leaf segmentation scores of network were evaluated by computing image -wise dice coefficient (DCs), pixel-based true positive (TPs) and false positive (FPs) rates on the "good" segmentations with DC $>$ t and object-based false negative (FNs), measuring the rate of "bad" segmentations (DCs $\leq$ t) where t = 0.7 is the DCs threshold.

The metrics are therefore defined as follows:

- $DCs = \frac{2 \cdot \sum_{i=0}^{M} (\hat{y}_i \cdot y_i)}{\sum_{i=0}^{M} (\hat{y}_i + y_i)}$
- $TPs = \frac{\sum_{i=0}^{M} (\hat{y}_i \cdot y_i)}{M}$
- $FPs = \frac{\sum_{i=0}^{M} [\hat{y}_i = 1 \wedge y_i = 0]}{M}$
- $FNs = [DC \leq t]$
- Let y denote a vector of the flatted ground-truth mask
- Let $\hat{y}$ denote a vector of the flatted prediction mask with pixel values which have probability higher than 0.7
- Let $i = \{0 \,..\, M\}$ denote the indices of the flatted vectors of y and $\hat{y}$

B. Results

Pixels with detected Rumex were classified into TP (true positive = correct detection where there is a Rumex in the pixel) and FP (detection of a Rumex where is none in that pixel) and FN (missed out Rumex, i.e. DC < 0.7). Using 20% of the images for training and 80% for validation (Model 0b) already resulted in a dice coefficient (DC) value of 82.87%. The high respectively low values on the TP and FP Rate come from the fact that they are measured pixelwise.

	DC [%]	***TP [%]***	***FP [%]***	***FN [%]***
Model 0	**83.91** ±14.66	**98.64** ±1.20	**0.50** ±0.68	**6.86** ±25.28
Model 0b	**82.87** ±14.42	**98.53** ±1.28	**0.61** ±0.78	**6.86** ±25.28
Model 1	**83.77** ±16.22	**98.72** ±1.05	**0.46** ±0.54	**6.86** ±25.28

Table 2*. Classification of the results (average over all images and standard deviation)*

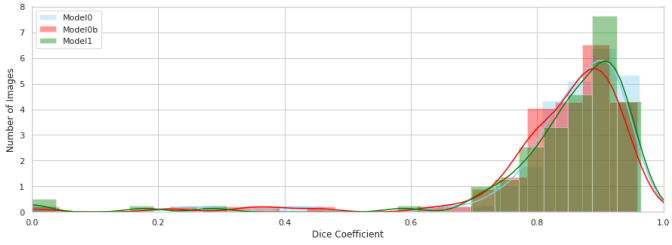

Fig. 15*. Distribution of the calculated Dice Coefficient of the Validation Set.* ***Blue****: Model0,* ***Red****: Model0b.* ***Green****: Model1.*

Fig. 15 shows the distribution of the dice coefficient over all validation images. The distribution of all models is very similar

with Model 0b showing slightly more values in the lower range. Most mistakes are made on images containing very little area of R. obtusifolius.

***Fig. 16**. Sample prediction on Validation Data (80% training / 20% validation). **(Left)** Field Images as Input. **(Right)** Predicted segmentation result (yellow mask) with framed ground truth (cyan border)*

VIII. Results Root Estimation

A. Evaluation Metrics

For finding the root centers a threshold to the corresponding output map was applied (Fig. 17), then for each connected component the corresponding center of mass was searched.

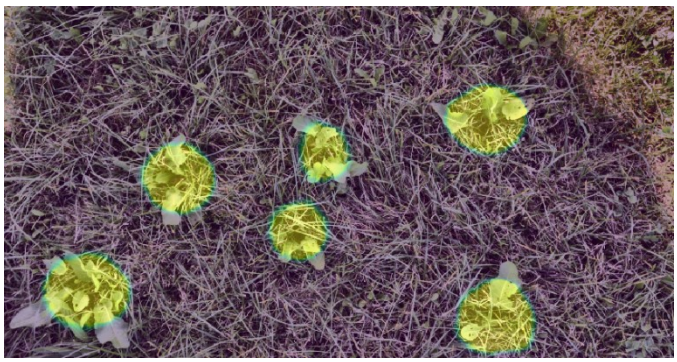

***Fig. 17**. Root Estimation Heatmap*

To evaluate accuracy, we measured the distances between the predictions and the ground truth positions. The Metric is defined as follows:

- Let D denote the Euclidian distance matrix between all predicted and ground truth points
- Let $I = \{1..m\}$ denote the indices of the column of the distance matrix D (Number of ground truth points)
- Let $J = \{1..n\}$ denote the indices of the row of the distance matrix D (Number of predicted points)
- Let t denote the maximum allowed distance between a predicted and a ground truth point.

Thus, the calculation for the True-Positives (TP), False-Negatives (FN) and False-Positives (FP) is composed as follows:

$$TP = \sum_{j \in J} (\min_{i \in I} D_j \leq t)$$

$$FN = \sum_{i \in I} (\min_{j \in J} D_i > t)$$

$$FP = J - TP$$

B. Results

Using this metric, we calculated Precision, Recall and the average between them (Balanced Accuracy) for the Validation-Set using a maximum of 6 Plants per Image. We've set t to 30 Pixels. This resulted in a "Tolerance-Circle" of about 9 cm diameter in real world.

	Precision [%]	***Recall [%]***	***Balanced Accuracy [%]***
Model 0	**88.02**	**80.85**	**84.43**
Model 0b	**87.25**	**76.97**	**82.11**
Model 2	**82.55**	**76.59**	**79.57**

***Table 3**. Root Estimation Results*

As in Table 3, a balanced accuracy of 84.43% resulted for Model 0. Model 0b is only about 2% worse compared to Model 0 considering only 20% of training data was used. However, the combination of both masks results in an improvement of about 5% (Model 0 vs. Model 2). Comparing Precision and Recall it is apparent that more errors are associated with false negatives than by false positives.

***Fig. 18**. Sample prediction on Validation Data. **Yellow**: Ground Truth labeled Root-Positions. **Black**: Predicted Positions. **Red**: Tolerance Circle. **Yellow-Shaped**: Ground Truth leaf mask*

C. Limitations

Because the center of the roots is calculated from connected components in the prediction mask, problems arise if plants overlap each other. This results in a decreasing accuracy because then the center of the connected plants is predicted as a root center (Fig. 20).

This happens especially if there is a large number of plants in the same image and therefore the accuracy of the algorithm decreases (Fig. 19).

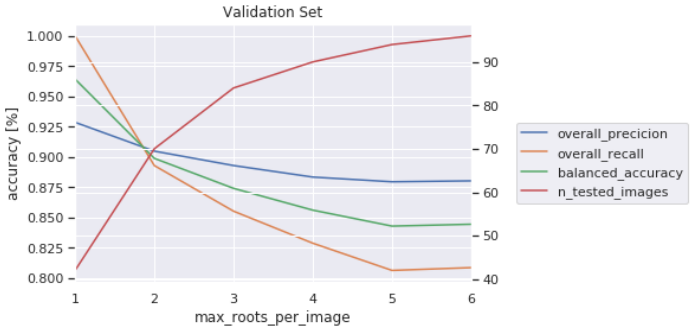

***Fig. 19**. Performance of the root estimation. The metric (Model0) was calculated for different maximal number of ground truth roots per image*

Some additional post-processing steps of the root estimation mask such as Watershed [22] were tested but didn't result in better overall accuracies.

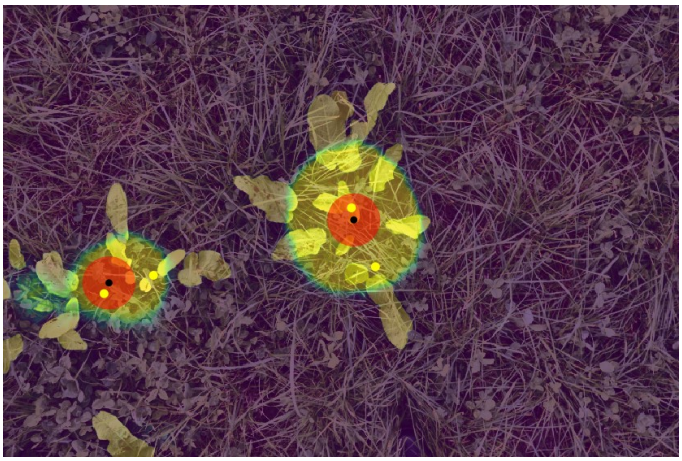

***Fig. 20**. Wrong Root Prediction, **Yellow Heatmap**: Root Estimation, **Yellow Points**: Ground Truth Roots, **Black Points:** Predicted Roots*

D. Tests on new Dataset

Acceptable results could also be achieved on a new data set recorded with different environmental conditions and sensor technology, see: 30_results.ipynb (not evaluated).

IX. Conclusion

We can show that our deep learning approach is able to segment and estimate the root position of R. obtusifolius plants with good accuracy. In earlier works [8], which are mostly based on Fourier Analysis, appealing results were achieved, but these depend on the fact that roughly the same environmental conditions always prevail.

However, our method is due to variations in the training set and additionally added light augmentation robust to changing light situations such as sunlight or shadows (Fig. 16). Also, the method can deal with partly covered leaves by other plants and softly blurred images.

But considering the results, there is still optimization potential which has to be investigated in further work. This includes a more accurate localization of the root positions especially if there are overlaid plants. Promising techniques to improve this could be object detection algorithms like:

- SSD: Single-Shot MultiBox Detector [23] or
- FRCNN: Faster R-CNN [24]

which are designed to detect also overlapping objects.

Moreover, the speed of the algorithm must be considered, as the plants must be detected in real time during the meadow crossing.

X. Appendix

A. Setup

For the programming and training process the following setup was used:

Software:

- Docker Engine 18.09.2 with ufoym/deepo v2.0.0

Hardware:

- CPU: Intel(R) Xeon(R) CPU E3-1271 V3
- 32 GB Ram
- GPU: Nvidia GTX 1070

XI. Acknowledgments

The authors would like to thank the Data Science community for many valuable insights and educational help in the field of machine/deep learning.

Furthermore we want to express our appreciation to companies that support our work: Omron Switzerland, Beckhoff Switzerland and NVIDIA Switzerland. Without their support our project WeedEraser would still be a vision.

References

[1] F. Van Evert, G. Polder, G. Van der Heijden, C. Kempenaar and L. Plotz, "Real-time vision-based detection of Rumex obtusifolius," p. 11, 2008.

[2] F. J. S. G. P. M. V. H.-J. v. D. A. L. e. a. Van Evert, „A robot to detect and control broad-leaved dock (Rumex obtusifolius L.) in grassland.," *Journal of Field Robotics,* Nr. 28, pp. 264-277.

[3] F. G. P. G. V. d. H. C. K. a. L. L. Van Evert, „Real-time, vision-based detection of Rumex obtusifolius L. in grassland.," *Weed Research,* Nr. 49, pp. 164-174, 2009.

[4] T. Kounalakis, G. A. Triantafyllidis und L. Nalpantidis, „Deep Learning-based Visual Recognition of Rumex for Robotic Precision Farming," p. 32, 2019.

[5] D. Šeatović, „A Segmentation Approach in Novel Real Time 3D Plant Recognition System," in *Computer Vision Systems*, Bd. 5008, A. Gasteratos, M. Vincze und J. Tsotsos, Hrsg., Springer Berlin / Heidelberg, 2008, pp. 363-372.

[6] D. Šeatović, H. Kutterer und T. Anken, „Automatic weed detection and treatment in grasslands," in *ELMAR, 2010 PROCEEDINGS*, 2010.

[7] D. Šeatović, „Methods for Real Time Plant Detection in 3-D Point Clouds," 2013.

[8] A. Binch und C. Fox, „Controlled comparison of machine vision algorithms for Rumex and Urtica detection in grassland," 2017.

[9] R. Latsch und J. Sauter, „Optimisation of hot-water application technology for the control of broad-leaved dock (Rumex obtusifolius)," *Journal of Agricultural Engineering,* Bd. 45, p. 137, 12 2014.

[10] D. Šeatović und R. Grüninger, „Smart Weeder: Novel Approach In 3D Object Recognition, Localization And Treatement Of Broad Dock In Its Natural Enviroment," 2007.

[11] F. A. L. A. D. J. E. G. G. V. d. H. L. L. Van Evert, „Sietse III: Or: Robot recognizes Rumex.," in *Proceedings of the 4th Field Robot Event 2006*, Stuttgart/Hohenheim, 2009.

[12] F. Chollet, Deep Learning mit Python und Keras: Das Praxis-Handbuch vom Entwickler der Keras-Bibliothek, mitp Professional, 2018.

[13] O. Ronneberger , P. Fischer und T. Brox, „U-Net: Convolutional Networks for Biomedical Image Segmentation," p. 8, 2015.

[14] J. Long, E. Shelhamer und T. Darell, „Fully Convolutional Networks for Semantic Segmentation," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* p. 10, 2015.

[15] S. Jégou, M. Drozdal, D. Vazquez, A. Romero und Y. Bendio, „The One Hundred Layers Tiramisu: Fully Convolutional DenseNets for Semantic Segmentation," 2017.

[16] K. Simonyan und A. Zisserman, „Very Deep Convolutional Networks for Large-Scale Image Recognition," *arXiv,* p. 14, 2014.

[17] „ImageNet," May 2019. [Online]. Available: http://www.image-net.org/.

[18] „TensorFlow," May 2019. [Online]. Available: https://www.tensorflow.org/.

[19] F. Chollet und others, „Keras," May 2019. [Online]. Available: https://keras.io.

[20] D. P. Kingma und J. Lei Ba, „Adam: A Method For Stochastic Optimization," *ICLR,* 2015.

[21] „Imgaug," 04 2019. [Online]. Available: https://github.com/aleju/imgaug.

[22] „OpenCV. Image Segmentation with Watershed Algorithm," 06 2019. [Online]. Available: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_watershed/py_watershed.html.

[23] D. A. D. E. C. S. S. R. C.-Y. F. A. C. B. Wei Liu, „SSD: Single Shot MultiBox Detector," 2015.

[24] K. H. R. G. J. S. Shaoqing Ren, „Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," 2015.

[25] K. Itakura und F. Hosoi, „Automatic Leaf Segmentation for Estimating Leaf Area and Leaf Inclination Angle in 3D Plant Images," p. 11, 2018.

[26] J. Redmon, S. Divvala, R. Girshick und A. Farhadi, „You Only Look Once: Unified, Real-Time Object Detection," p. 10, 2015.

[27] S. G. V. d. H. F. V. E. a. A. S. Hiremath, „The role of textures to improve the detection accuracy of Rumex obtusifolius in robotic systems.," Weed Research, 2012.

[28] F. J. S. G. P. M. V. H. V. D. E. L. e. a. Van Evert, „Robotic control of broad-leaved dock.," *Precision Agriculture,* pp. 725-732, 2009.