Vietnamese-German University

FRANKFURT UNIVERSITY OF APPLIED SCIENCES

Plant Phenotyping for plant growth monitor

17242 - Tran Minh Hieu
Supervisor: Vo Bich Hien
25 December 2024

Table of Contents

1. **Introduction** 2
2. **Dataset acquired** 4
3. **Methodologies** 5
   - **3.1.HSV filtering** 5
   - **3.2.Watershed Algorithm** 6
   - **3.3.Leaves segmentation** 8
4. **Result** 9
   - **4.1.Overall performance** 9
   - **4.2.Over-segmentation** 11
   - **4.3.Under-segmentation** 12
5. **Conclusion** 13
6. **Reference** 14

1. Introduction

As the human population continues to grow, food requirements are also rising steadily. To address this challenge, it's crucial to identify and analyze plant varieties that exhibit faster growth and higher yields (Shook et al., 2021). Several factors, including daily temperature fluctuations, limited rainfall, soil fertility, biotic and abiotic stresses, and crop management practices, influence plant growth and productivity (Liliane et al., 2020). Plant phenotypes, such as leaf size, leaf count, plant height, and biomass, offer a quantitative measure of plant growth, reflecting the synergy between its genetic composition and the external environment. Plant breeders, geneticists, and researchers must comprehend and these intricate dependencies and harmonize difference languages to accurately track plant growth, optimize yields, and select suitable genotypes for specific environmental conditions (Costa et al., 2019).

Tracking plant growth is a complex and challenging task, especially considering the rapid dynamic growth, leaf occlusion, and changes in leaf angular orientation (Kolhar & Jagtap, 2023). Plants continuously grow through the emergence of new leaves and roots at meristematic tissues during vegetative development. As leaves emerge from the shoot, they undergo various developmental stages, resulting in significant variations in leaf characteristics among different plant species and even within accessions of the same species (Kolhar & Jagtap, 2023). These leaf traits, particularly size, shape, and count, are crucial for photosynthesis and yields. Therefore, tracking plant leaves at different development stages becomes crucial for estimating and monitoring plant growth.

Plant phenotypes can be categorized into three main groups: structural, physiological, and temporal. Structural phenotypes, such as leaf count and projected plant area (PPA), provide insights into the component-level structure of plants. Physiological phenotypes, including chlorophyll content, water content, and

leaf surface temperature, are crucial for analyzing plant functions like photosynthesis and transpiration. Temporal phenotypes, such as leaf emergence time, offer a holistic view of plant growth over time.

This study focuses on structural phenotypes, particularly leaf count and PPA, and temporal phenotypes, such as leaf emergence time, for growth estimation and tracking. By utilizing method described in (Szachowicz, 2020/2024), we gain a comprehensive understanding of plant growth dynamics.

2. Dataset acquired

The dataset, Komatsuna, utilized for this experiment is publicly accessible for download. This dataset serves as a valuable resource for monitoring the growth of Japanese mustard spinach. It comprises two datasets: a multi-view dataset comprising RGB images captured by three RGB cameras positioned at distinct angles, and an RGB-D dataset capturing RGB and depth images using an RGB-D camera.

The multi-view RGB image dataset presents a growth sequence of Komatsuna plants, accompanied by manually labeled ground-truth regions for the plant leaves. The dataset includes images of five Komatsuna plants captured continuously over a ten-day period, with an interval of four hours between each image sequence. Each image sequence comprises a total of sixty images. Consequently, the multi-view dataset contains a total of 900 images. All plant images were captured under controlled lighting, temperature, and humidity conditions. Each plant image is characterized by varying numbers of leaves.

Figure 2.1: rgb_01_04_009_05

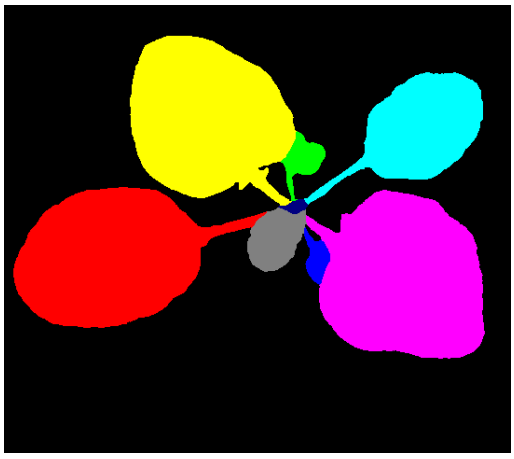


Figure 2.2: label_01_04_009_05

3. Methodologies

3.1.HSV filtering

To obtain the segmentation for each leaf in the Komatsuna set, the HSV range filtering and watershed algorithm are employed to estimate each individual leaf base based on its shape. The OpenCV watershed algorithm utilizes a marker-based approach to assign distinct colors to marked areas.

To identify the suitable application area for the watershed algorithm, the input image must be converted to the HSV color space. Subsequently, thresholding is employed to segment the image into distinct regions of interest. The thresholding process effectively separates the objects from the background. Given our objective of identifying the leaf area of the plant, the HSV color space proved instrumental in precisely filtering out the plant and background regions. Although not a fundamental component of image processing, the HSV filter step is crucial as it enables the inclusion of objects with sufficient detail while precisely separating the background.

```
im_threshold = cv.inRange(hsv_image, (low_H, low_S,
               low_V), (high_H, high_S, high_V),
               cv.THRESH_BINARY_INV+cv.THRESH_OTSU)
```

This line is performing color thresholding on an image that's been converted to the HSV color space. im_threshold which will store the result of the thresholding operation. This result will be a binary image (black and white).

The cv.inRange is the OpenCV function that does the actual thresholding. The image, which is converted to HSV color space, lower bound and upper bound of HSV color space is to ensure all shade of the color is captured.

`cv.THRESH_BINARY_INV+cv.THRESH_OTSU`: This argument specifies the type of thresholding to be applied.

`cv.THRESH_BINARY_INV` means that pixels within the specified color range will be set to black (0), and pixels outside the range will be set to white (255) – it's inverted.

`cv.THRESH_OTSU` is a method that automatically calculates an optimal threshold value based on the image histogram, further refining the thresholding process.

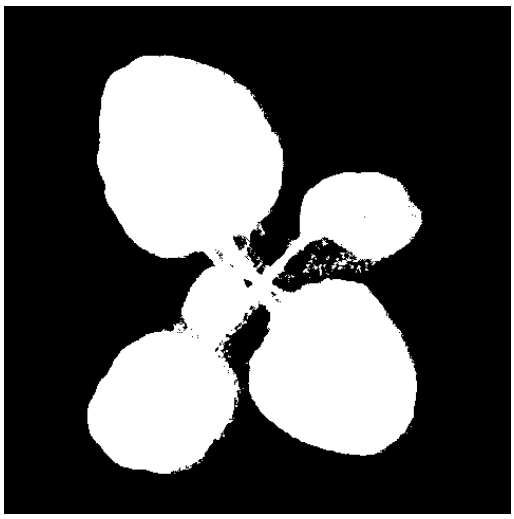


Figure 3.1: masked_00_02_009_02

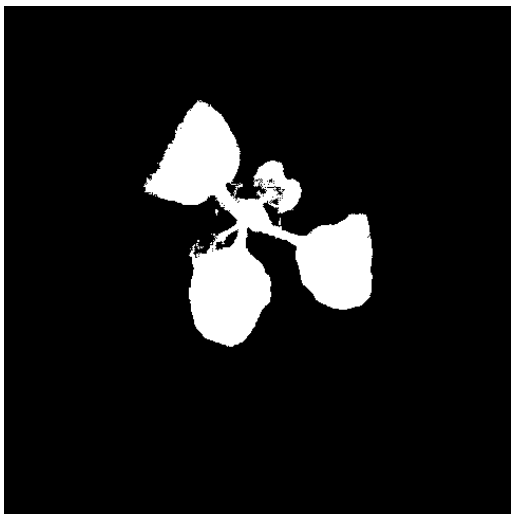


Figure 3.2: masked_00_03_007_00

The default range which somewhat satisfies all 900 images in the dataset is (40, 30, 30) to (75, 255, 255).

3.2.Watershed Algorithm

This code snippet demonstrates a series of image processing techniques using OpenCV, focusing on morphological operations to extract important features from an image and prepare it for further analysis. The process begins by creating a kernel, which is a matrix of ones, used to perform convolution operations on the image. The morphologyEx function is then used to perform morphological opening

on the thresholded image, im_threshold. This operation helps remove noise by eroding and then dilating the image. The result, opening, is further subjected to dilation using the same kernel to ensure a solid background (sure_bg).

```
kernel = np.ones((3, 3), np.uint8)
opening = cv.morphologyEx(im_threshold,
cv.MORPH_OPEN, kernel, iterations=5)
sure_bg = cv.dilate(opening, kernel, iterations=7)
```

Following this, the distance transform is computed on the opened image to highlight the distance of each foreground pixel from the nearest zero pixel. This transform assists in identifying regions that are surely part of the foreground (sure_fg). A threshold is applied to the distance transform output to segment these sure foreground regions. This processed image can now be used to aid in tasks such as segmentation, object recognition, or other image analysis applications.

```
dist_transform = cv.distanceTransform(opening,
                 cv.DIST_L2, 5)
_, sure_fg = cv.threshold(dist_transform, 0.3 *
             dist_transform.max(), 255, 0)
sure_fg = np.uint8(sure_fg)
```

After calculating the foreground and background of the image, the regions that are not within either of the mentioned regions are classified as unknown.

```
unknown = cv.subtract(sure_bg, sure_fg)
```

3.3.Leaves segmentation

The marker is employed to mark all the valleys which will be later sign to a color for visualization. The 'connectedComponents' function is used to identify distinct components within an image, effectively labeling each discrete region in the 'sure_fg' binary image, which represents the foreground. The 'markers' array is initialized to label these regions, and each label is incremented by one to ensure that the background is labeled with a different marker. Then the ‘markers' is set to zero wherever the 'unknown' regions are detected. Finally, the 'watershed' function of OpenCV is applied to the input image ('input_im') using the modified 'markers'. This function treats the pixel values as topographic surfaces and floods basins (segments) with different colors, effectively separating different objects in the image by their watershed lines, which form the boundaries between different segments.

```
_, markers = cv.connectedComponents(sure_fg)
markers = markers + 1
markers[unknown == 255] = 0
markers = cv.watershed(input_im, markers)
```

4. Result

4.1.Overall performance

After the segmentation process, the result is saved at “masked_images” folder with the naming conventions similar to the original images name, with only the prefix of the file name change from ‘rgb’ to ‘masked’. The growth curve then can be plotted using line chart. The chart compose of 5 difference plant id, the timestamp, and the average coverage area. The naming rule of plant images are ‘masked_AA_BB_CCC_DD.png’. There are five parameters: AA, which represents the camera ID (ranging from 00 to 02), BB, which represents the plant ID (ranging from 00 to 04), CCC, which represents the day ID (ranging from 000 to 009), and DD, which represents the capture time (ranging from 00 to 05). It is important to note that days begin at 15:00 and end at 15:00 the following day, and images are captured every four hours.

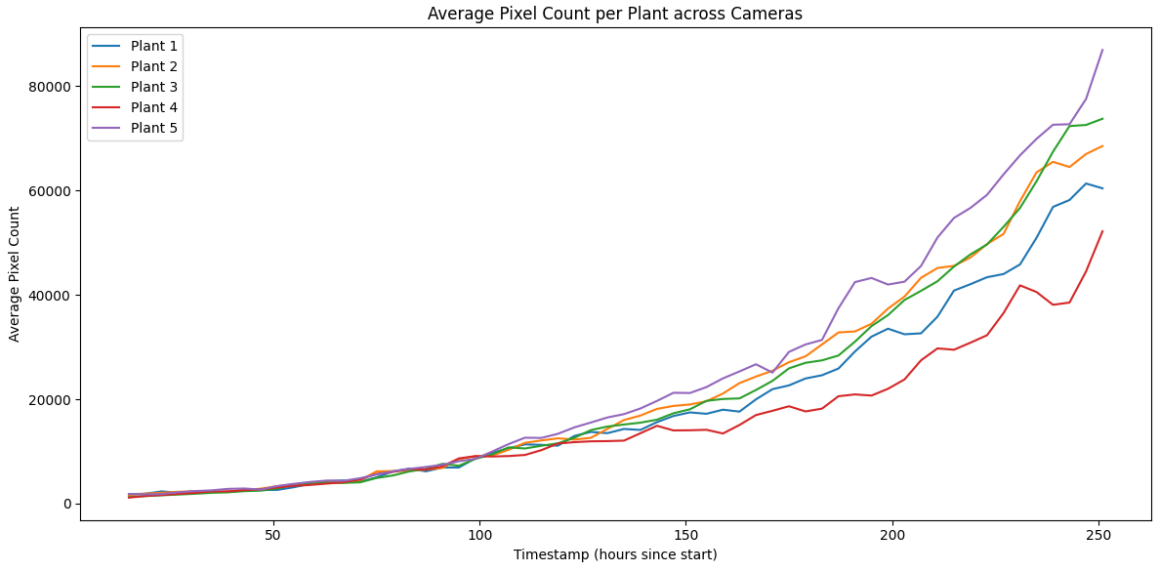


Figure 4.1: Average coverage area of each plant

Plant ID	Mean Score	Min Score	Max Score
0	0.9230	0.6846	0.9767
1	0.9433	0.8750	0.9785
2	0.9269	0.7818	0.9848
3	0.9191	0.7183	0.9661

Plant ID	Mean Score	Min Score	Max Score
4	0.9355	0.8022	0.9694

Table 4.1: Detail dice coefficient of 5 plant

The dice coefficient indicates that the average of 900 segmented images has 92.96% similarity, with an average minimum difference of 77.24% and an average maximum similarity of 97.51%. The initial results appear promising, as many images have well-fitted masks, as evidenced by the following examples:

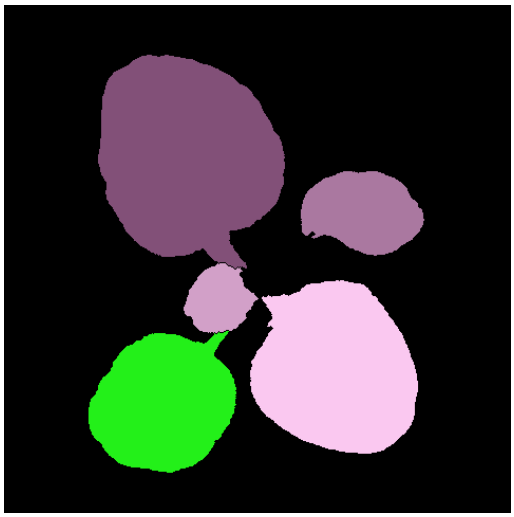


Figure 4.4: masked_00_04_002_03

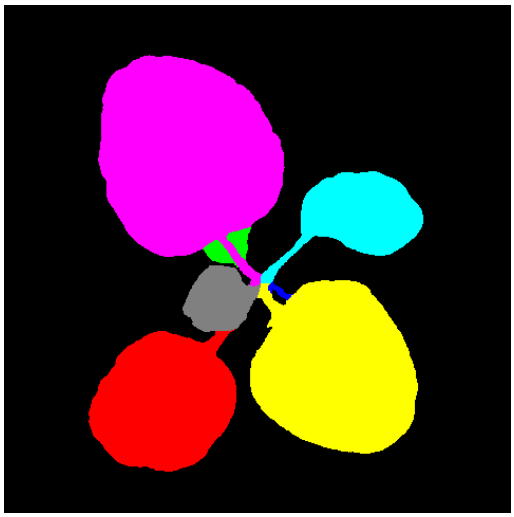
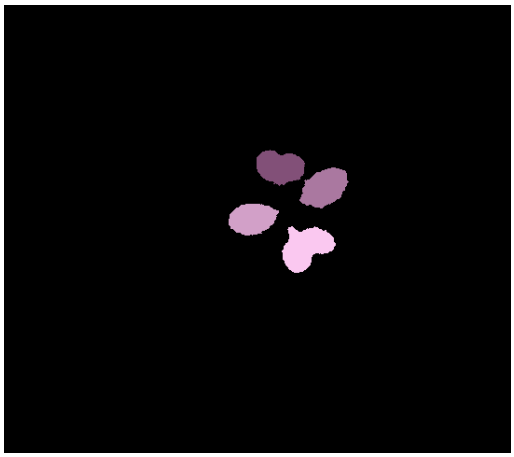
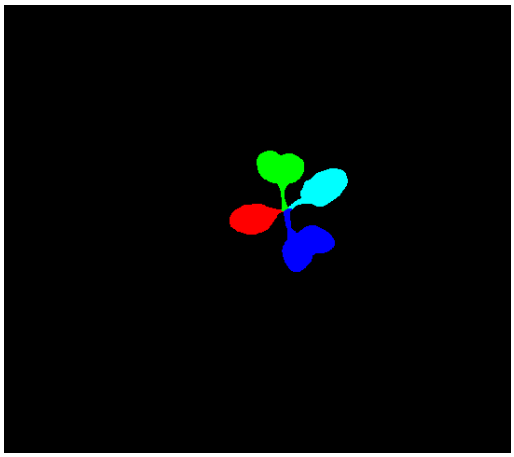


Figure 4.5: label_00_04_002_03





Although thoroughly evaluated, the dice coefficient function solely compares the overall shape of the segmented images to the labeled image, not each leaf

segmentation. Consequently, despite its high value, it cannot distinguish between under segmentation and over segmentation.

4.2.Over-segmentation

Over-segmentation is the condition of detecting individual leaf parts as separate leaves. This phenomenon is particularly common during connecting components phase, where the leaf veins are incorrectly identified as borders. This can be attributed to the darker or shaded regions of the leaves, which may result in fragmentation and subsequent cutting into smaller pieces.

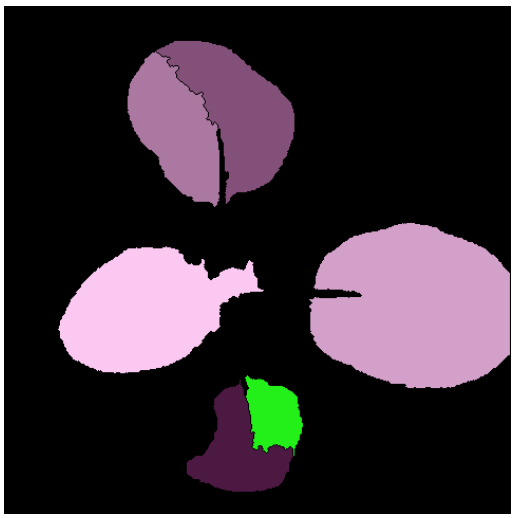


Figure 4.6: masked_02_01_009_01

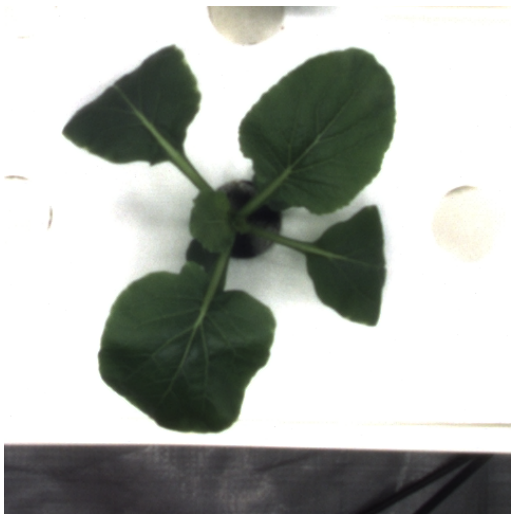


Figure 4.9: rgb_00_03_009_05

Figure 4.8: masked_00_03_009_05

Figure 4.7: rgb_02_01_009_01

4.3.Under-segmentation

Conversely, segmentation is a common issue observed in the masked image folder. This can occur when multiple leaves share similar HSV (Hue, Saturation, and Value) color ranges, causing the segmentation algorithm to interpret them as a single entity rather than recognizing each leaf as a distinct object. This misinterpretation can lead to inaccuracies in leaf count and even misleading leaves general shape. Although intuitively, the under segmentation can happened with every image, however, it is mostly occurs in the early development of the plant and reduce in later development state as each leaf have more distinct HSV color range.

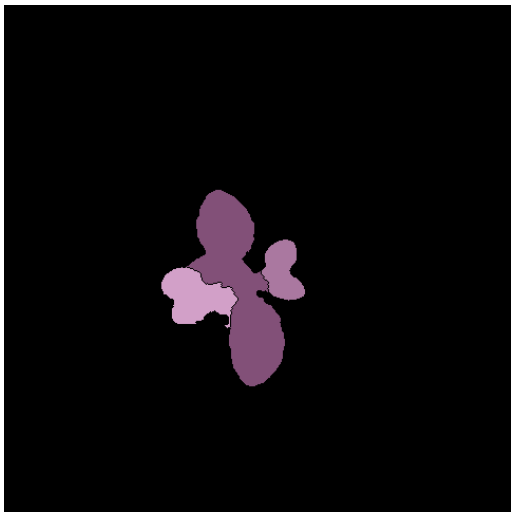


Figure 4.12: masked_00_01_004_00

rgb_00_01_004_00

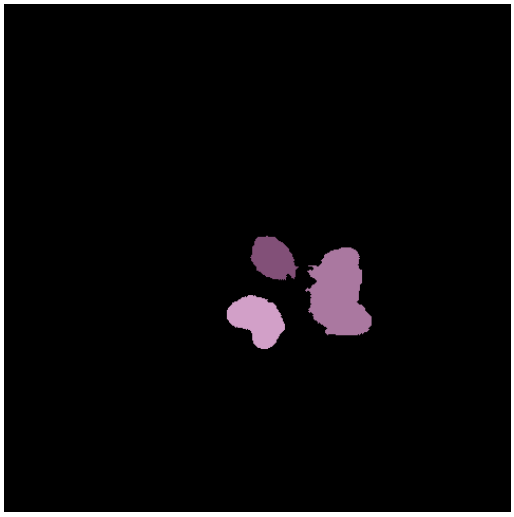


Figure 4.11: masked_00_00_003_00

Figure : rgb_00_00_003_00

5. Conclusion

This project aimed to recreate a comprehensive pipeline for plant phenotyping using OpenCV watershed functions to quickly estimate the plant's leaf area. Through the application of HSV filtering and watershed algorithm, we have achieved a somewhat accurate segmentation of the komatsuna plant. While some individual leaf segments exhibit issues caused by excessive segmentation and insufficient segmentation, the overall general shape segment retains high detail and closely resembles the ground truth images.

However, this method is not reliable due to varies in light conditions as well as leaf color. In order to enhance the precision and efficiency of plant phenotype tracking, a more advanced method employing machine learning capabilities is necessary. In the current machine learning literature, deep learning methods have established themselves as the leading approach in various image-based tasks, including object detection and localization, semantic segmentation, image classification, and others (Sharma et al., 2021). Deep learning methods in computer vision, particularly deep convolutional neural networks, seamlessly integrate image feature extraction with regression or classification within a single pipeline, which is trained end-to-end simultaneously. Nevertheless, despite their widespread adoption in other domains, deep learning applications have yet to be a general-purpose tools that tailored to the plant phenotyping community to providing support and encouragement for the adoption and utilization of these methods (Ubbens & Stavness, 2017).

6. Reference

1. *Shook J, Gangopadhyay T, Wu L, Ganapathysubramanian B, Sarkar S, Singh AK (2021) Crop yield prediction integrating genotype and weather variables using deep learning. PLoS ONE 16(6): e0252402. https://doi.org/10.1371/journal.pone.0252402.*

2. *Ngoune Liliane, T., & Shelton Charles, M. (2020). Factors Affecting Yield of Crops. IntechOpen. doi: 10.5772/intechopen.90672.*

3. *Costa C, Schurr U, Loreto F, Menesatti P and Carpentier S (2019) Plant Phenotyping Research Trends, a Science Mapping Approach. Front. Plant Sci. 9:1933. doi: 10.3389/fpls.2018.01933.*

4. *Shrikrishna Kolhar, Jayant Jagtap, Phenomics for Komatsuna plant growth tracking using deep learning approach, Expert Systems with Applications, Volume 215, 2023, 119368, ISSN 0957-4174, https://doi.org/10.1016/j.eswa.2022.119368.*

5. *Szachowicz, J. (n.d.). GitHub - julzerinos/python-opencv-leaf-detection: Python-based OpenCV program for detecting leaves and creating segmentation masks based on images in the Komatsuna dataset. GitHub. https://github.com/julzerinos/python-opencv-leaf-detection?tab=readme-ov-file.*

6. *Neha Sharma, Reecha Sharma, Neeru Jindal, Machine Learning and Deep Learning Applications-A Vision, Global Transitions Proceedings, Volume 2, Issue 1, 2021, Pages 24-28, ISSN 2666-285X, https://doi.org/10.1016/j.gltp.2021.01.004.*

7. *Ubbens JR and Stavness I (2017) Deep Plant Phenomics: A Deep Learning Platform for Complex Plant Phenotyping Tasks. Front. Plant Sci. 8:1190. doi: 10.3389/fpls.2017.01190.*