Project Number: 774571
Start Date of Project: 2017/11/01
Duration: 48 months

Type of document D4.6 – V1.0

Fruit Detection

Dissemination level	PU
Submission Date	2021-10-15
Work Package	WP4
Task	T4:8
Type	Report
Version	1.0
Authors	Mario Gilcher, Adonis Shabani
Approved by	Andrea Gasparri + PMC

Executive Summary

This document describes the details of the activities conducted in Task 4.8 “Fruit Detection”. The main goals of the fruit detection are briefly introduced. In the next step, the data sources of the campaign in 2020 are described and illustrated. Then the two major methodological approaches are outlined, and summarized, and then validated in a final chapter. In conclusion, the proposed methodology is well suited to detect hazelnut fruits on the field. A fully operational fruit detection with these methods is held back by hardware limitations and robust software-based approaches to match detected nuts to specific trees.

Table of Content

Abbreviations and Acronyms

UGV	Unmanned Ground Vehicle
CNN	Convolutional Neural Network
D	Deliverable
LiDAR	Light Detection And Ranging
RGB	Red, Green and Blue
GPS	Global Positioning System
SIFT	Scale invariant Feature Transform
SURF	Speeded up robust features
RANSAC	Random Sample Consensus

1 Introduction

While the importance of large-scale yield production assessment and estimation has been already discussed in deliverable D5.5 “Fruit development and production monitoring”, this deliverable is focused on a tree level fruit detection. The main objective was to count visible hazelnuts based on the RGB imagery acquired by the ground vehicle described in deliverable 3.1. The experiment in this document has been conducted in 2020 on the hazelnut orchard number 16, monitoring 10 six-year-old hazelnut plants during the growing season 2020. The orchard layout was 4.5 m x 3.0 m, and the plants were grown as multi stemmed bushes (see Figure 1). The image shows a subset of one orthomosaic produced for deliverable D4.3 “Aerial Orthophoto-Mosaic”, with 4 of the 10 trees visible.

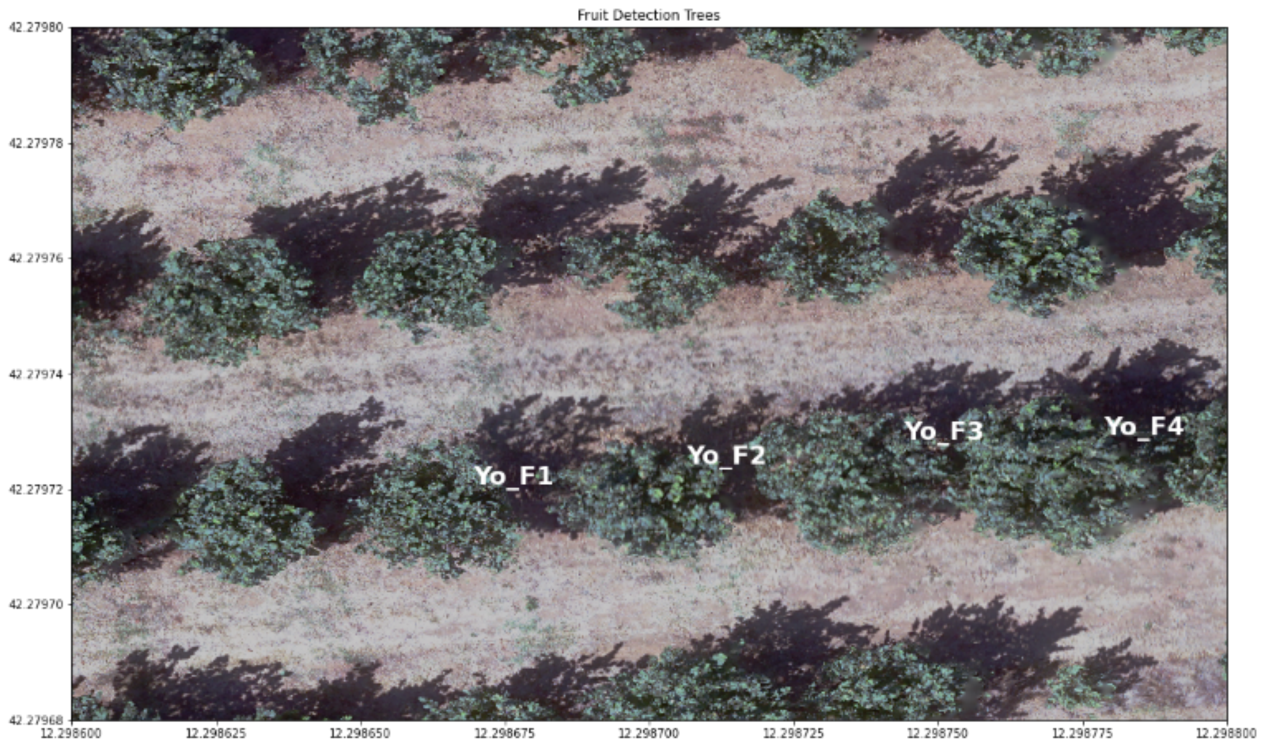

Figure 1: Selected Trees for the Fruit Detection Experiment 2020, Background: Orthomosaic from 2021

The core element is the binary segmentation of data (see Figure 2. We start with a set of Sony images, and a set of labeled images which we produced by carefully digitizing visible nuts in all of the images. These images are then cut into square 512 by 512 pixel patches and split into training and validation sets. With this data, a U-Net image segmentation model is trained, which predicts the presence of hazelnut pixels for a given patch. Then we identified tree-facing images for each tree and summed up the nuts counted in these images. These nut counts will then be compared to the verified in-field nut counts carried out by the partner UNITUS carried out by hand during the kernel filling stages and at harvesting. Conceptually, counting nuts based on images sounds problematic, since there will be nuts counted multiple times, since they are visible on multiple images. On the other hand, there will be nuts counted zero times, because they are not visible from the angles that were captured. For this reason, the initial plan was to facilitate the enriched Laser scans, since they provide a more complete dataset, all features are unique and feature occlusion should be minimal. Since we ran into some technical and methodical issues with the spectral enrichment, an additional, purely image-based approach was pursued in parallel to the point-cloud based fruit detection. Some potential solutions we are currently pursuing are mentioned in section 0. In the following document, we will describe the data sources of the experiment, and then outline the methodology of the two approaches. We will then describe the results of the image-based approach and conclude by commenting on its limitations.

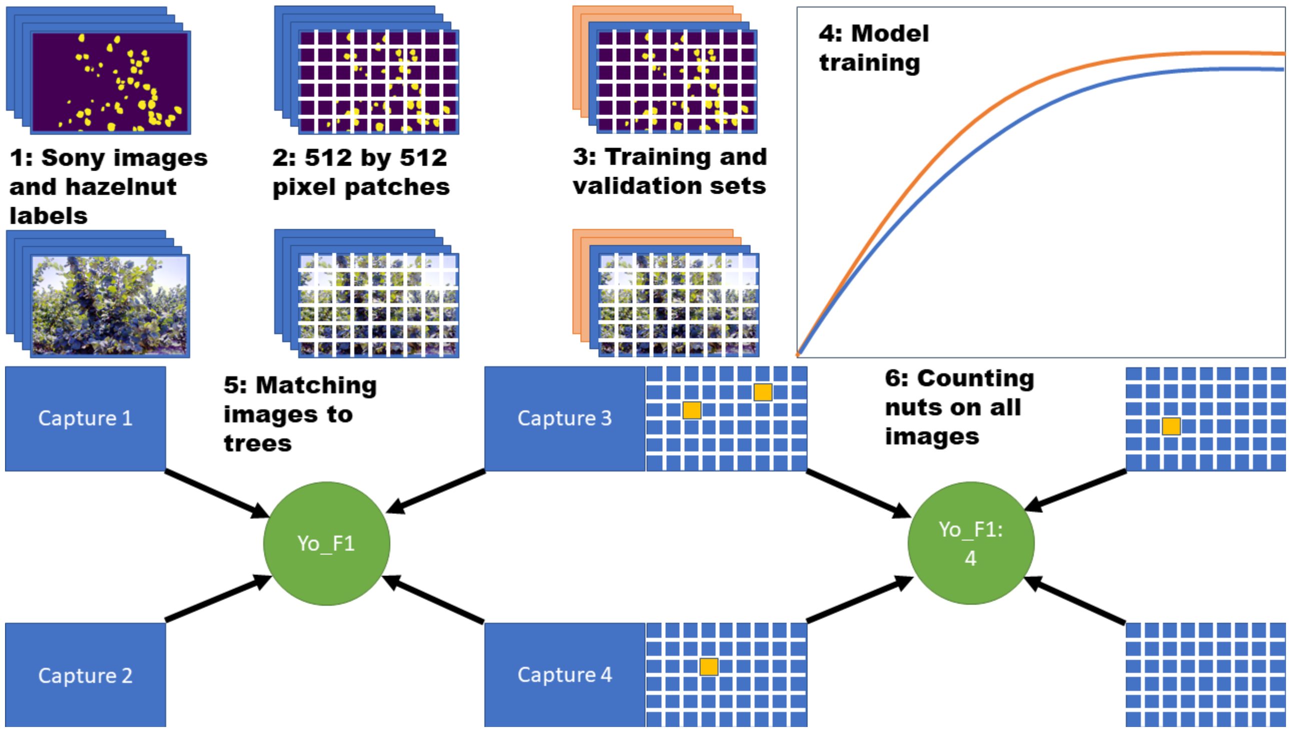
1: Sony images and hazelnut labels
2: 512 by 512 pixel patches
3: Training and validation sets
4: Model training
5: Matching images to trees
6: Counting nuts on all images
Capture 1
Capture 2
Capture 3
Capture 4
Yo_F1
Yo_F1: 4

Figure 2: Overview of the data processing concept

2 Data sources

2.1 Ground Vehicle

All data collected in the 2020 Fruit Detection Campaign was acquired from the robotic prototype (Unmanned Ground Vehicle, UGV) developed specifically for the PANTHEON project and described in detail in deliverable D3.1 "Robotic Prototypes". The remote sensing concept with all sensors was described in deliverable D4.1 " Multispectral LiDAR Point Clouds". In theory, each tree should be observed from 4 different positions (see Figure 3). Each position has 4 captures in total. One capture is the laser scan, and 3 captures at varying pitches from the camera sensor. This configuration ensures that each part of the tree is covered from multiple angles while at the same time overall tree positions, and therefore operational hours and cost are minimized.

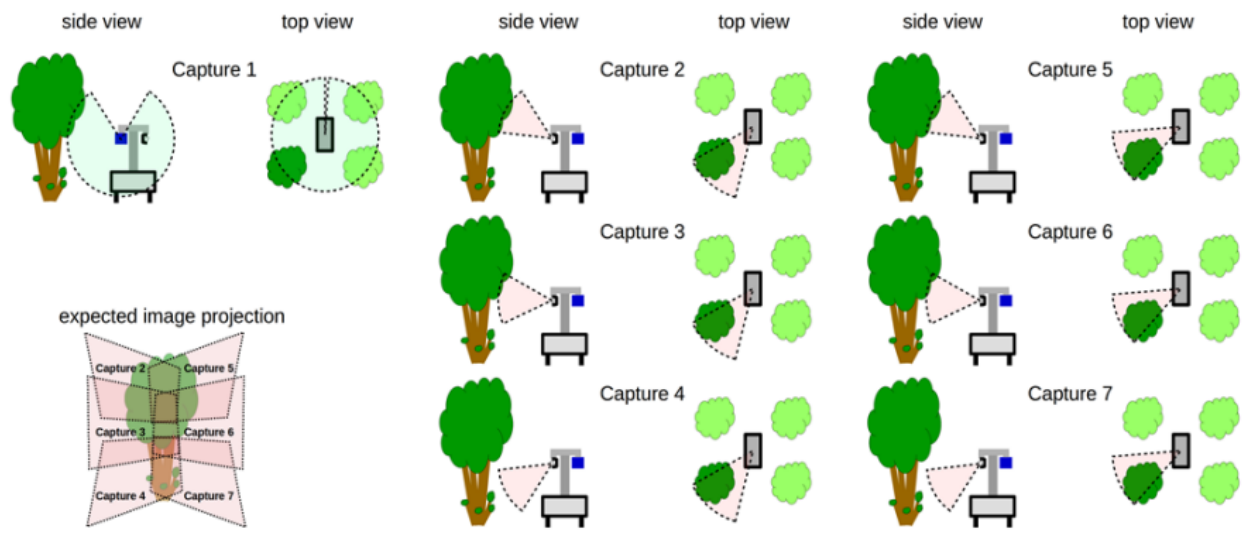

Figure 3: Remote sensing concept for the fruit detection trees

The high-resolution captures were acquired with a Sony α5100 RGB camera, also described in detail in deliverable D4.1 "Multispectral LiDAR Point Clouds". The camera is equipped with a 28mm wide angle lens. While it does not have any multispectral bands, its high resolution of 24 MP is highly suitable for image segmentation tasks with hazelnuts, which are relatively small targets. Figure 4 shows a raw image from the campaign. On closer look, many nuts and nut clusters can be identified immediately. They can be distinguished by both color, which is a lighter shade of green, as well as texture and position relative to the branch. A second image shows all Sony captures of tree Yo_F3 (see Figure 1). Each row represents a capture position with three different pitches, looking at bottom, middle and top of each tree. Here we can see a lot of ground pixels in the bottom capture, mostly leaves in the middle capture and leaves mixed with sky in the top capture. In some of the images, mainly the bottom and middle captures, other trees are shown, which can be considered as a problem in the image-based approach, since the image segmentation that detects the nuts does not distinguish between individual trees. Another issue is that the captures are not consistent in their position relative to the tree, which is very difficult to achieve in field conditions. Figure 6 visualizes the normal of each capture projected on a 2D plane, relative to the trees. The experimental trees which are used to validate the fruit detection algorithm are shown in green. For these 10 trees nut counts as well as yield values exist for 2020. As the image shows, the positions around each tree show some substantial variance, and Sony captures do not always show the tree in the middle but are tangential to a theoretical sphere around the tree. The variance in distance to a tree specifically poses a problem to the tree alignment, which determines the relationship between each capture and a tree. Initially, the strategy was to match the 4 captures closest to a given tree (see section 5.3.6 in deliverable D4.1). This clearly does not work under field conditions, since in some cases captures are selected which face in a different direction, while captures that face the tree but are a little bit further away are omitted.

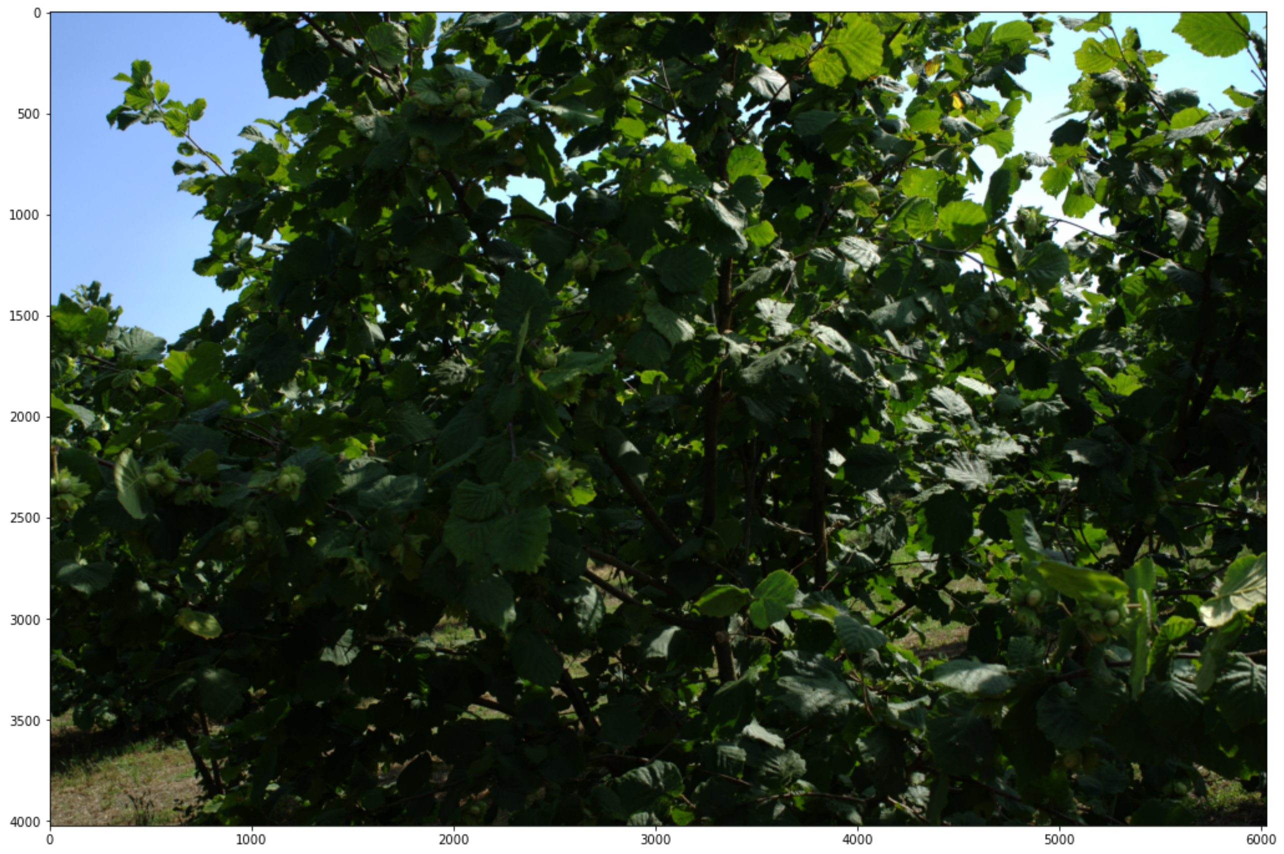

Figure 4: Raw Sony image of tree Yo_F3

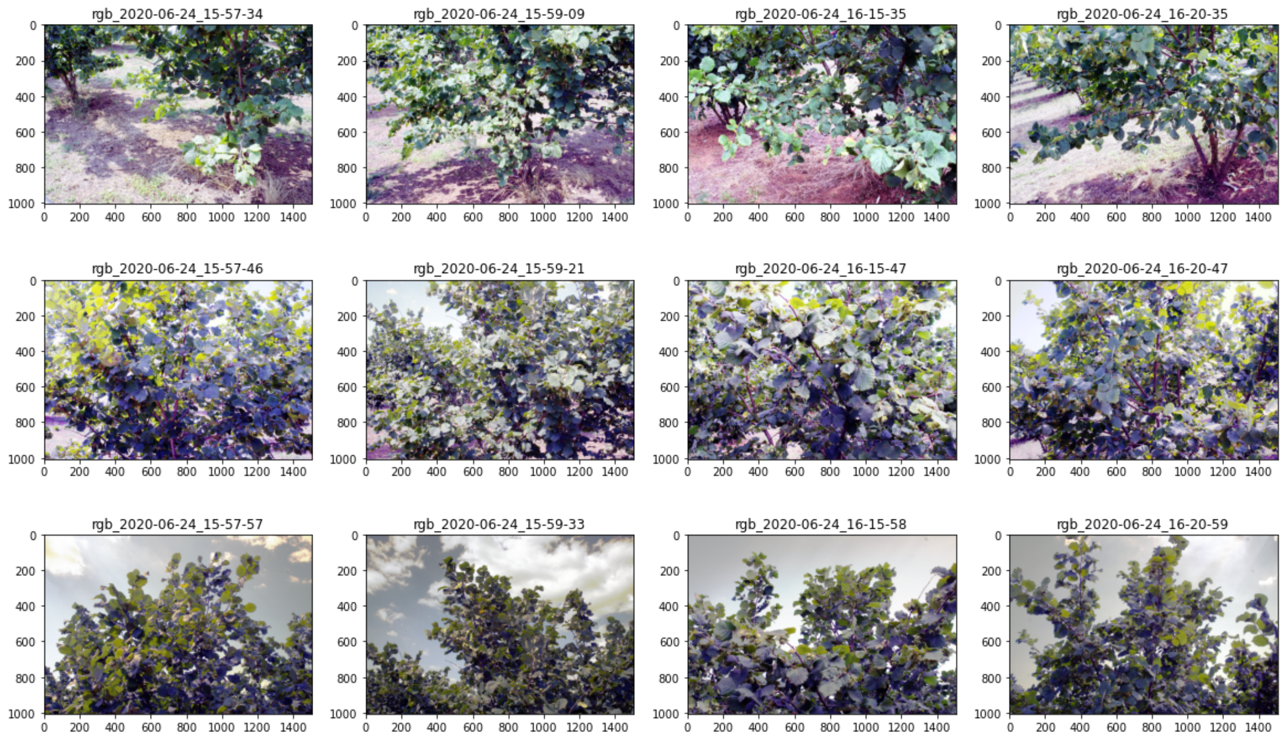

Figure 5: All Sony images of tree Yo_F3, histograms equalized for visibility

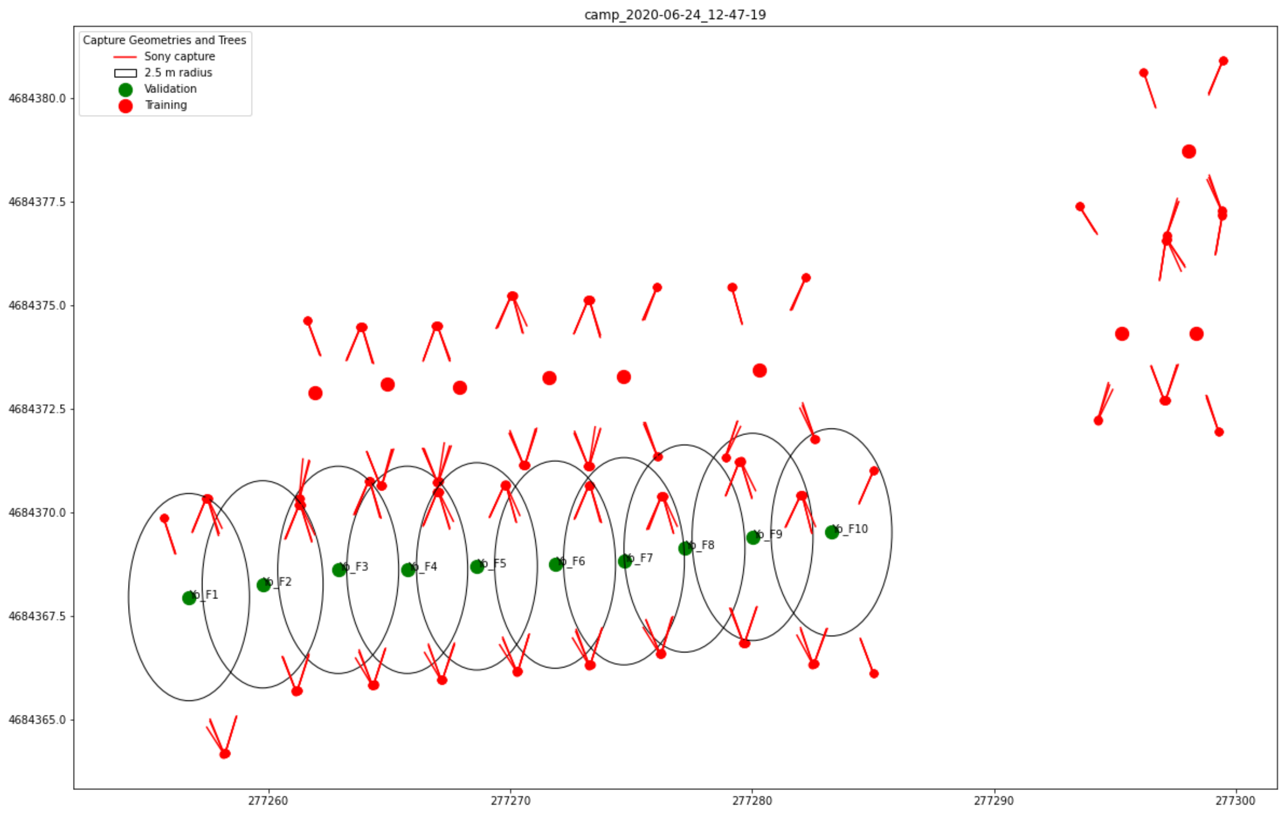

Figure 6: All capture geometries visualized

In addition to the Sony imagery, there are a in general 4 laser scans per tree. These scans were aligned, enriched with the RGB imagery (see section 7.7 in deliverable D4.1), and trees were extracted for the point cloud-based fruit detection. The laser scans were also used to verify the measured observer geometry. The UGV is equipped with a GPS which tracks the position of the Vehicle while the gimbal on the robot ensures a specific pitch, roll and yaw relative to each position. This observer geometry is essential for the point cloud enrichment and labelling process, since a very precise agreement between the real-world position and orientation between the laser scan and the Sony captures is necessary. While this agreement has been achieved in the lab calibration, field conditions are very different, and we used synthetic images of the laser scans to do plausibility checks. We used the extrinsic matrix of each Sony capture to project the point cloud of the scan, thus taking a virtual photo. In theory, these synthetic photos should show the same extent of the tree. While the colors are very different, the position of the branches and leaves relative to the image extent can be easily used to check the agreement of the observer geometries. Figure 7 shows these synthetic images for a single tree. A direct comparison with the images in Figure 5 shows a clear discrepancy of sometimes 1000 pixels which is much higher than what was achieved in the experimental calibration setup in the lab, which can be attributed to much more difficult field conditions in terms of variability of vehicle inclination, wind conditions.

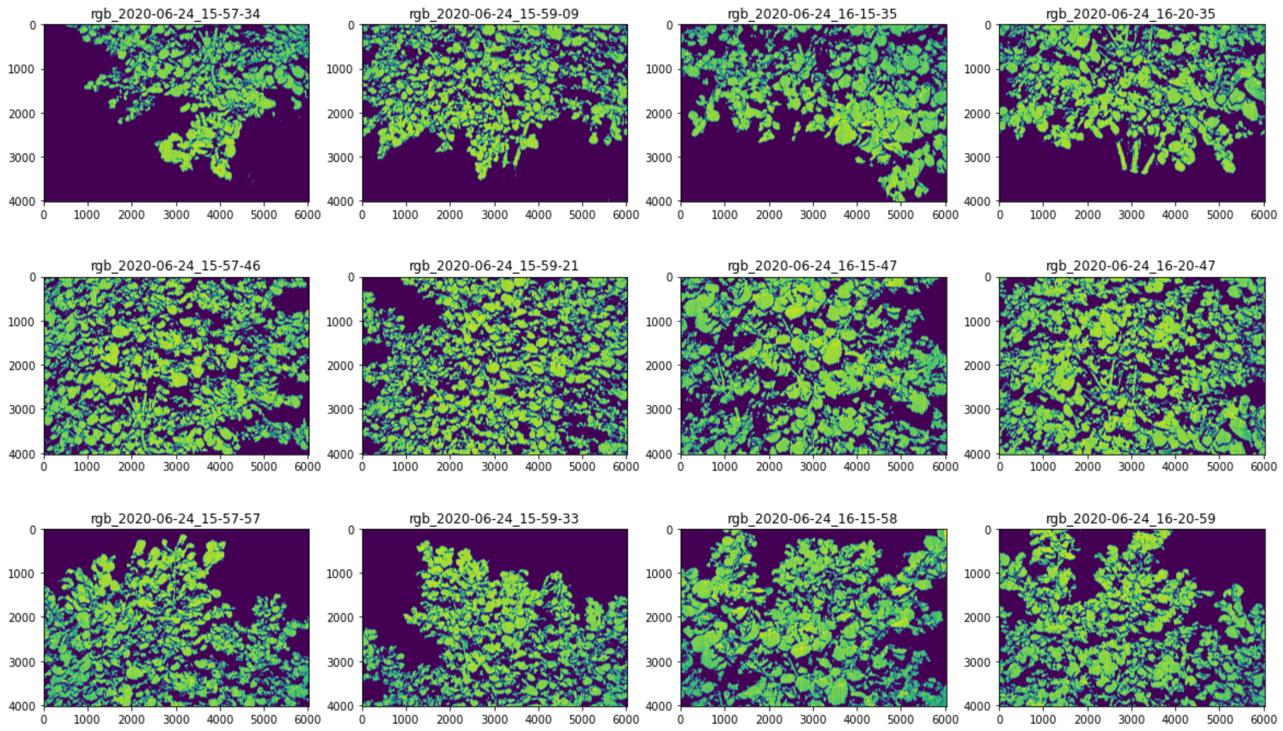

Figure 7: Synthetic images of tree Yo_F3

2.2 Ground Truth

During the growing season 2020, an accurate manual counting of clusters and nuts was carried out in the selected plants in the field 16. This counting was performed by a two-person team of the local Unit UNITUS, as shown in Figure 13, in which one person accurately counted the clusters per branch and one person recorded the data. This prevents omitting some clusters hidden in the canopy during counting. The counting was performed three times in July, August, and finally during harvesting time in September, while only the July counting is used in this experiment, because of the proximity to the UGV data acquisition by the end of June. The process of the field campaign is described in deliverable D5.5 “Fruit development and production monitoring”. In addition, a manual nut harvest was carried out, and plant yield was weighed after manually clearing it from defective and empty nuts.

Tree ID	Number of Clusters	Number of Nuts (July)	Nuts per Cluster	Yield (kg)
Yo_F1	459	1299	1.88	2.68
Yo_F2	559	1156	2.8	2.52
Yo_F3	948	1368	2.62	4.36
Yo_F4	1092	2349	2.21	4.83
Yo_F5	981	2756	2.53	4.36
Yo_F6	1564	4060	2.76	5.34
Yo_F7	1404	3491	2.54	4.55
Yo_F8	1463	3917	2.78	5.19
Yo_F9	1288	2758	2.75	5.1
Yo_F10	1516	5678	2.38	6.38
Mean ± SD	1127.4 ± 391.8	2883.2 ± 1443.1	2.49 ± 0.6	4.53± 1.17

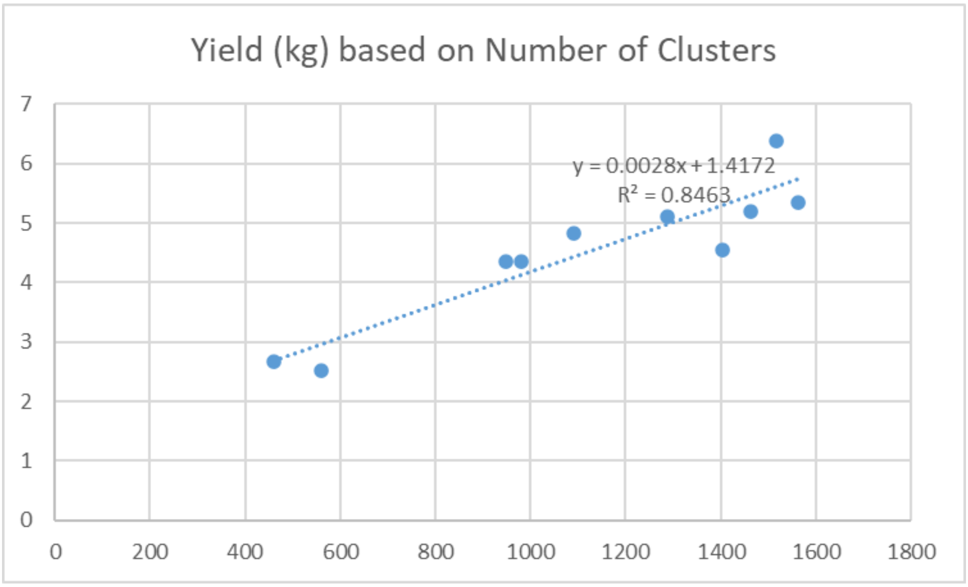

Figure 8: Relationship of nut count and yield

The counted nut clusters show a very strong positive linear relationship with the harvested nuts, as shown in Figure 8. As a result, if nuts could be counted effectively on a standing tree, these estimates could be used to predict the actual effective plant yield. All variables also show a relatively high variability within the 10 trees. In consequence, the idea to use counts of visible nuts based on images to predict potential hazelnut yield seems feasible and very promising.

2.3 Image Labelling

For the binary classification at pixel level, each pixel of the image has to be labeled. In general, deep learning requires a large number of annotated samples. In this experiment, 225 RGB-images were labeled using a custom interactive python-based tool. Using a 512 by 512 pixel patch size, this results in 17325 samples than can be used to train and validate the classifier. The script allows to draw polygons in an interactive widget which then flags all pixels inside the polygon as 1, while all others are left at 0 which indicates that they do not contain hazelnuts. Figure 9 shows one of the 225 images. The custom tool offered a couple of practical advantages as we could adjust it to our specific requirements regarding the input and output data types. The drawing of the polygons, however, did not work perfectly and in some cases the closing of the polygon left some small irregular artifacts. Drawing around the nuts was not always easy. Though the high resolution of the images certainly helped, there is sometimes a lot of overlap between the cluster and other leaves. In preparation of the labeling process, we made the conscious decision to rather overshoot the cluster area a bit, in order to simplify the already very labor-intensive workflow, as in general deep learning models are able to deal with labeling noise. On average, the labelling of an image took about 5-10 minutes.

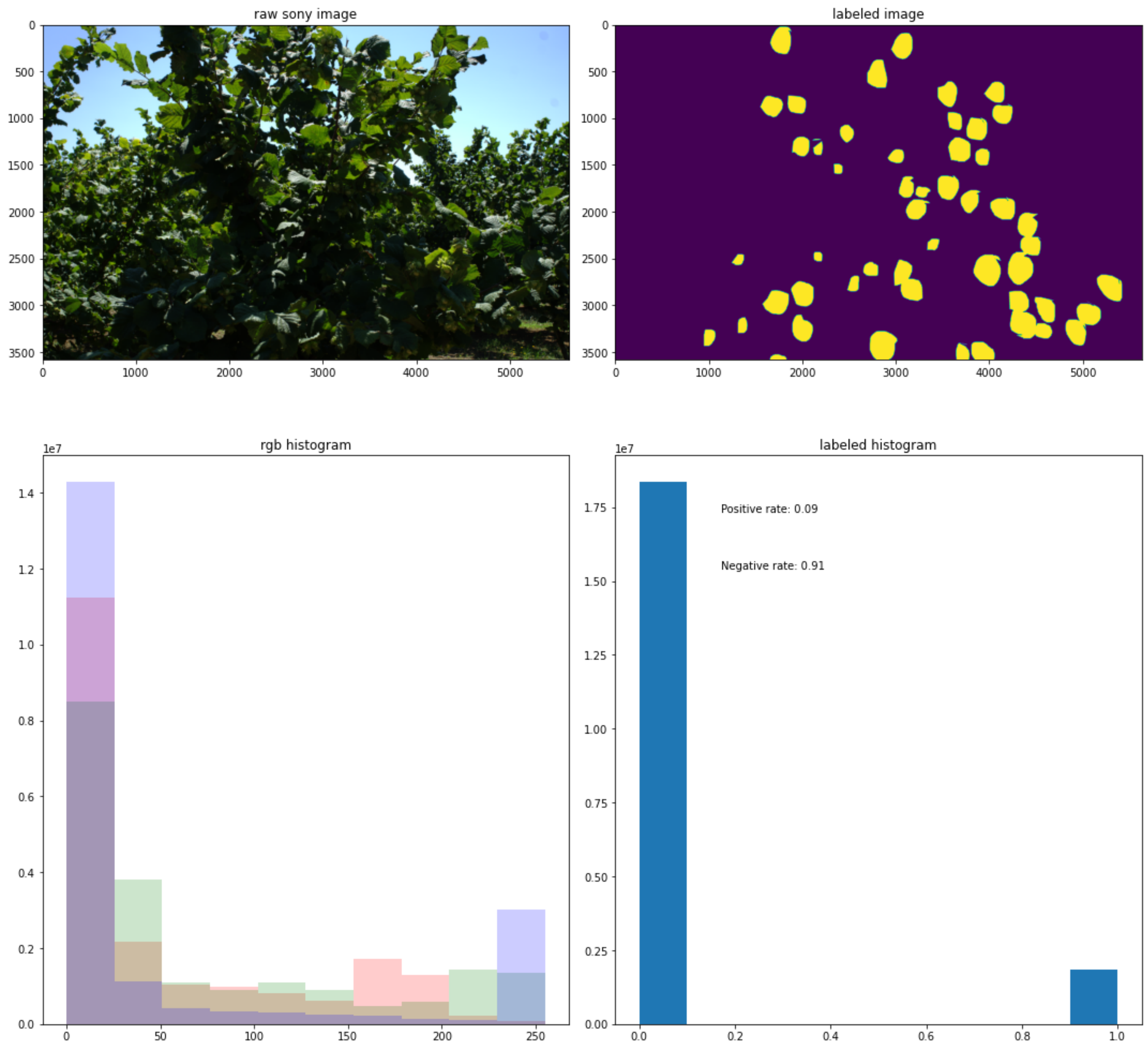

Figure 9: Fully labeled example

3 Methodology

3.1 Image Based Approach

3.1.1 Image Segmentation

Convolutional Neural Networks (CNN) are everywhere in computer vision, especially in image segmentation. They have their origin in the works of Yann Lecun during the Nineties (Lecun & Bengio, 1995), and use kernel based filters to extract spatial features from input images. They were discovered to have many different applications in image processing in computervision, being able to detect a large number of various objects in images (Krizhevsky et al., 2012). The specific architecture we employed in this experiment was U-Net, a state of the art image segmentation algorithm which is well established in the deep-learning and remote sensing communities (Gilcher & Udelhoven, 2021), and has been successfully used to detect apples in orchards (Häni et al., 2020). As Figure 10 shows, it works by nesting and reversing maxpooling and convolution layers, hence the eponymous "U" shape. The layers are interconnected in a way, such that spatial features are learned on several scales. This is very important in this experiment, since the nut clusters can be at varying distances from the tree, and consequently have varying size in the image. We used a custom implementation available within the keras/tensorflow environment for python (Zak, n.d.). The performance will be assed with the F-Score (also F1-Score, F-measure (Sasaki, 2007)), a common performance metric in the CNN community, which is a combination of recall and precision.

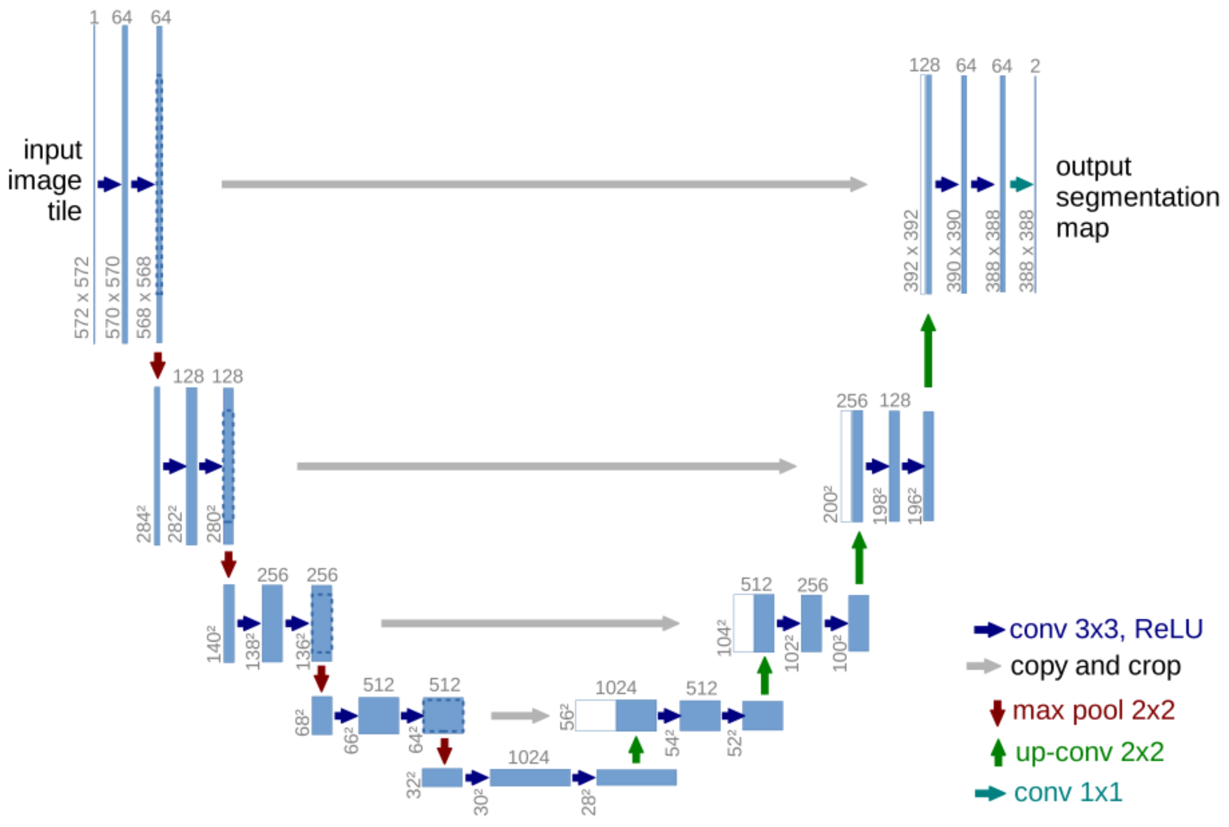

Figure 10: The U-NET architecture (Ronneberger et al., 2015)

The network expects square inputs with patch sizes divisible by two multiple times. We chose a patch size of 512 by 512 pixels, which allows the samples to easily contain one or two clusters consistently, if the sensor is within a distance between 2 and 10 meters of the tree, which is indeed a reasonable assumption considering the typical platting pattern of hazelnut orchards. Two examples are illustrated in Figure 11. In some samples, the nuts can be clearly seen from a human vision perspective, while in others there is context needed to distinguish the clusters. Most samples however do not contain any nuts. The samples were extracted systematically with no overlap. The raw images were cut to match exactly 11 by 7 patches per image. The raw output of the image segmentation algorithm are patches with a-posteriori probabilities. These patches need to be stitched together in order to reconstruct an equivalent to the original image.

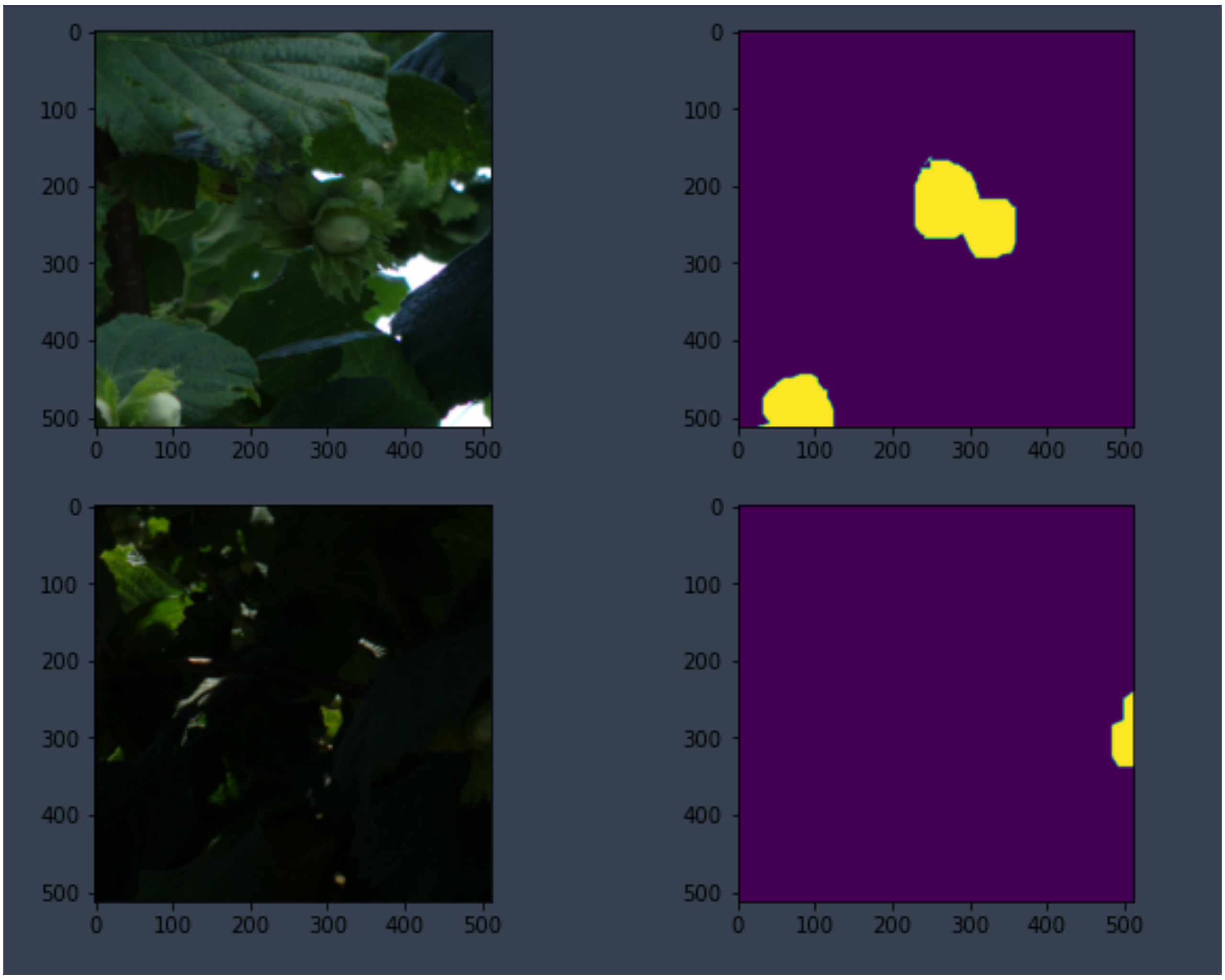

Figure 11: Two samples of the image segmentation training data

3.1.2 Cluster Count

The image segmentation itself does not produce a number of objects, but rather a number of connected pixels flagged as 1 or 0 (after thresholding of a posteriori probabilities). This is a conceptual problem, since we do not want the absolute number of detected hazelnut pixel, but rather the number of connected clusters as an estimate of visible clusters. Consequently, some postprocessing needs to be employed co cluster pixels together, and also to remove some smaller clusters related to prediction noise. In the experiments we used opencv (Bradski, 2000) to merge connected clusters, and filter based on pixel count. Figure 12 illustrates the clusters above the threshold of 1500 pixels in green for both reference and prediction dataset. It shows that a lot of the labelled clusters are identified correctly, but also a substantial number of false positives specifically at the neighboring trees. The false positive and false negative rates seem to be consistent across all images, and do not seem to have an effect on overall prediction performance.

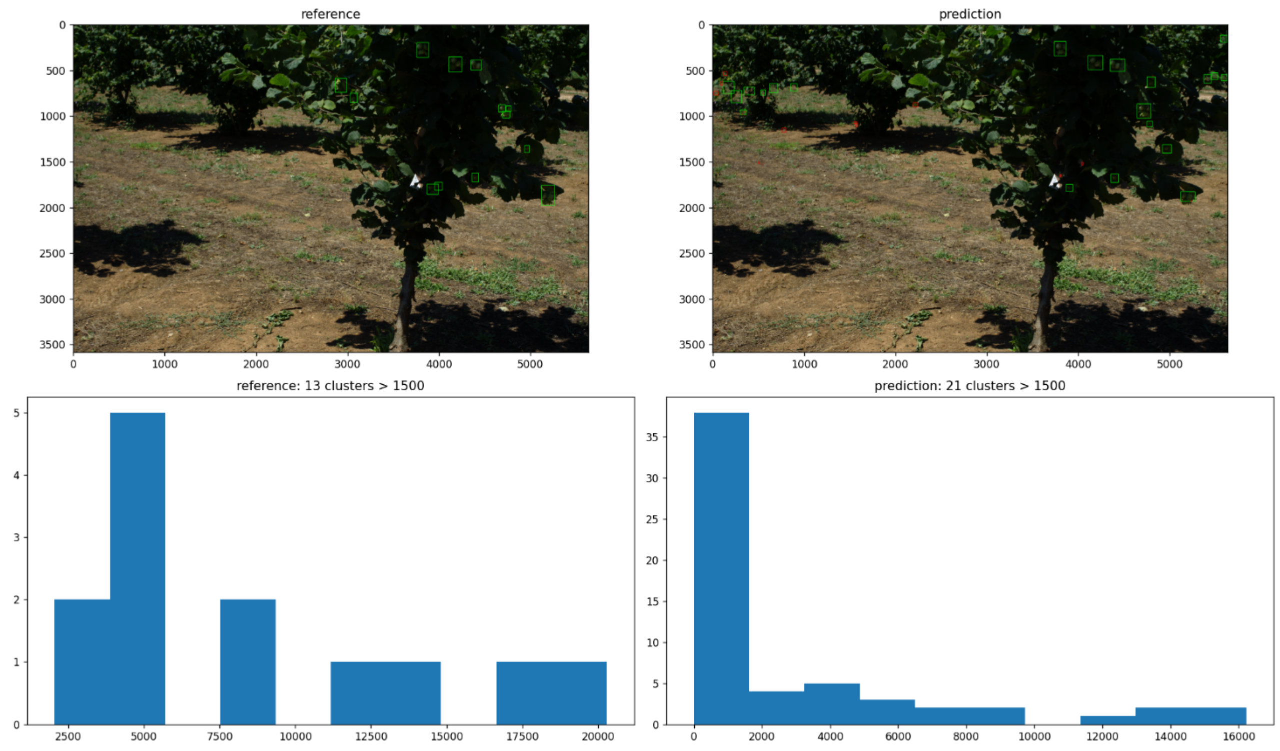

Figure 12: Cluster count based on image segmentation

3.1.3 Masking

In order to distinguish between the nuts detected on the tree in focus, and nuts on a neighboring tree, the segmented images are then postprocessed with masks derived from the synthetic laser scan images. In an image, we know the position and orientation, but we do not know the position of each pixel in 3D space. This is different for the laser scan point clouds. We can use the position of a tree to filter the point cloud based on a given buffer around that position. Then we project the point cloud onto the theoretical image plane of the Sony capture, and we get a synthetic image with only the given tree. We then need to use gaussian blur on the image to increase the area of the laser scan on the projected plane. Afterwards, a threshold is used to produce a binary mask. Figure 13 illustrates this, though in case of tree Yo_F3 the effect is rather subtle. It is most visible in the top left caption, where there is a neighboring tree in the original image (see Figure 5).

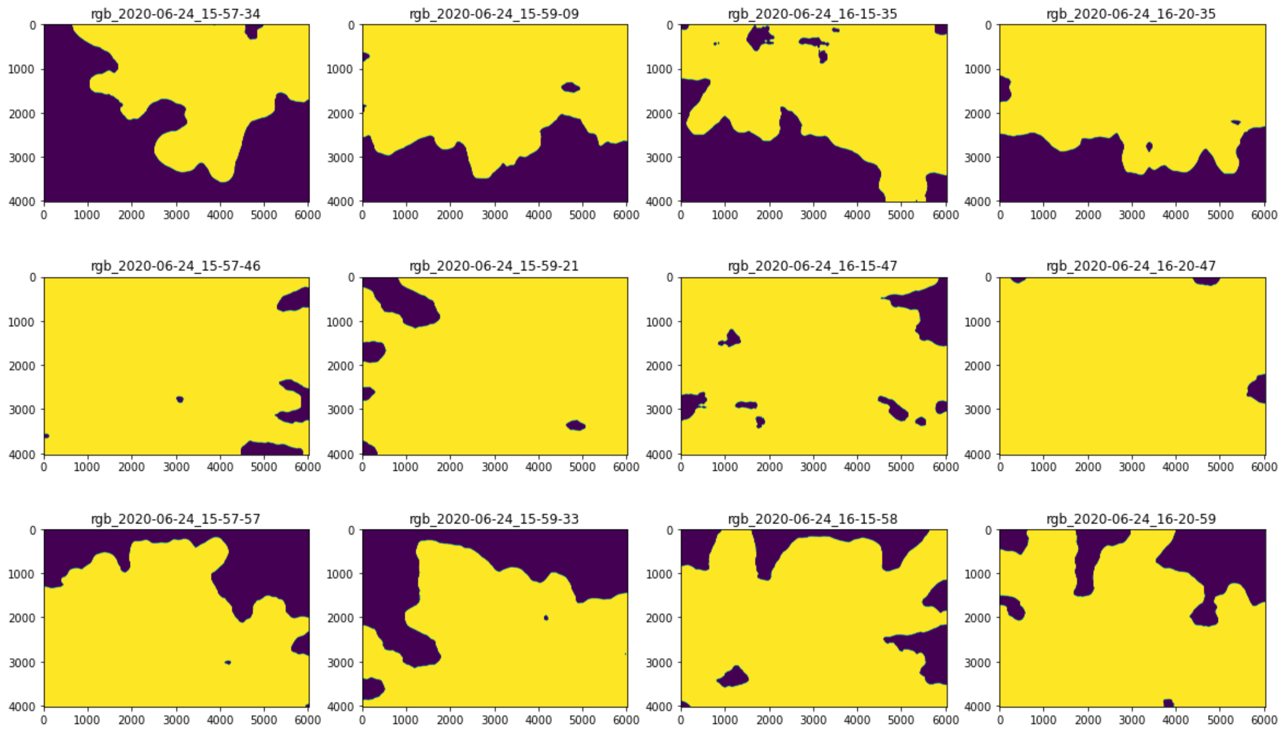

Figure 13: Tree masks for tree Yo_F3

3.1.4 Tree Matching

As stated in section 2.2, the initial approach to match trees with images did not work, because of unpredictable vehicle-tree distances. Instead, we employed a more sophisticated observer geometry approach to find out if a given capture is to be associated with a given tree. In theory, if the Sony image is supposed to observe a tree, a focal point defined by a multiple of its normal, should lie within a certain distance threshold around a tree. Therefore, we multiplied each normal to a theoretical length of 3 meters and checked if it is within a circle of 2 meters around a tree. If so, a match was found, as illustrated in Figure 14. Out of the 10 trees in question, 12 matches were found in 7 cases. Tree Yo_F1 is missing one capture, and observer geometry around Yo_F9 and Yo_F10 is much more inconstant, so the cameras are not pointing directly at the tree.

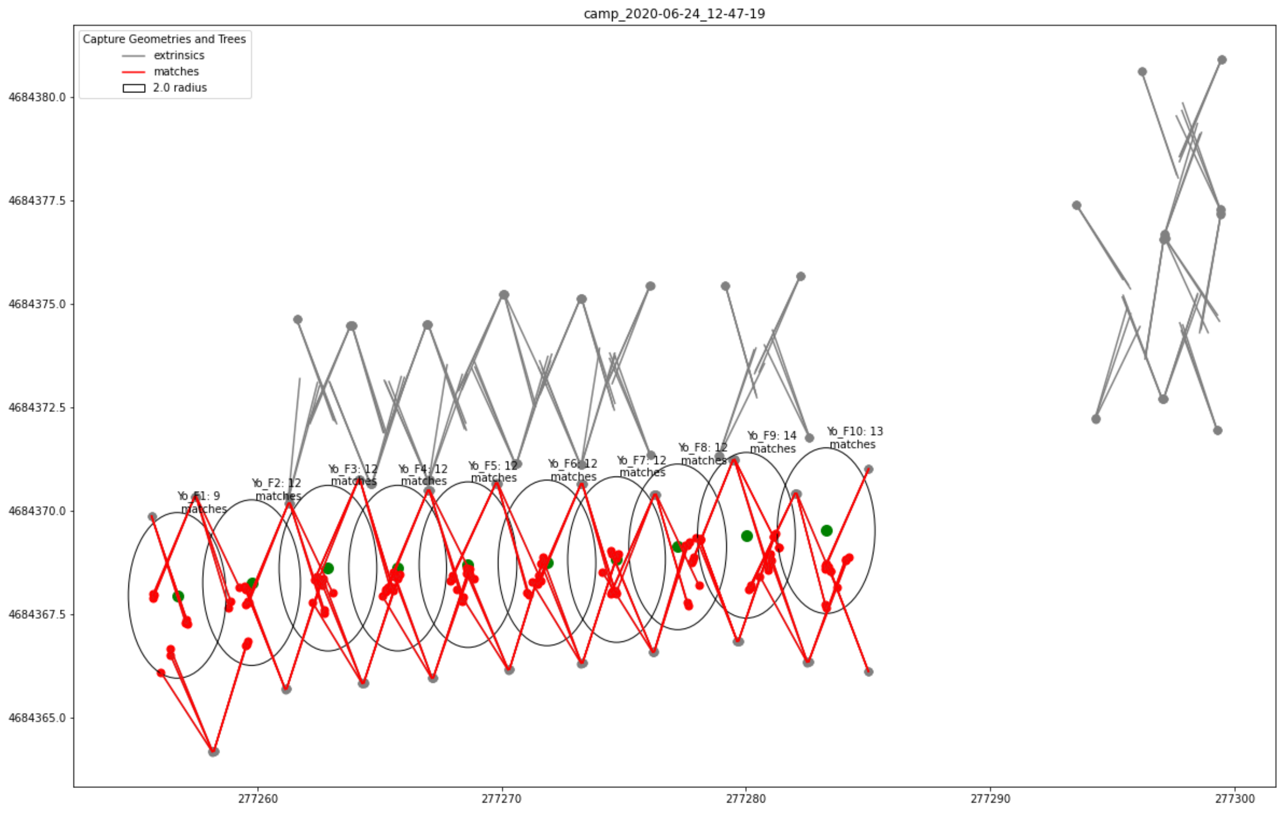

Figure 14: Image/tree matching illustrated

3.2 Point Cloud Based Approach

3.2.1 Spectral Enrichment

In order to produce point clouds with color values or multispectral reflectances we need to know the exact observer geometry. Using the observer extrinsics given by the IMU and the gimbal settings, we can project synthetic images based on the enriched point clouds onto the theoretical image plane. The premise here, is that we assume that the point cloud geometry is relatively precise, since it is the result of data driven and iterative fine tuning (see deliverable D4.1). The geometry of the images however is just raw values extracted from the robot, and prone to lots of different error sources. In Figure 15 we see the result of the spectral enrichment based on the original geometries. The synthetic images are blurred and histogram equalized to enhance visibility of the otherwise very small points derived from the point cloud. They can be directly compared to Figure 5 and should in theory show very similar color values. However, it is evident that there are a lot of very bright color values from the ground projected on the point cloud. At the same time, a lot of blue pixels are also projected onto the canopy. This is a clear indication that the spectral enrichment, calibrated in the laboratory, does not work under field conditions.

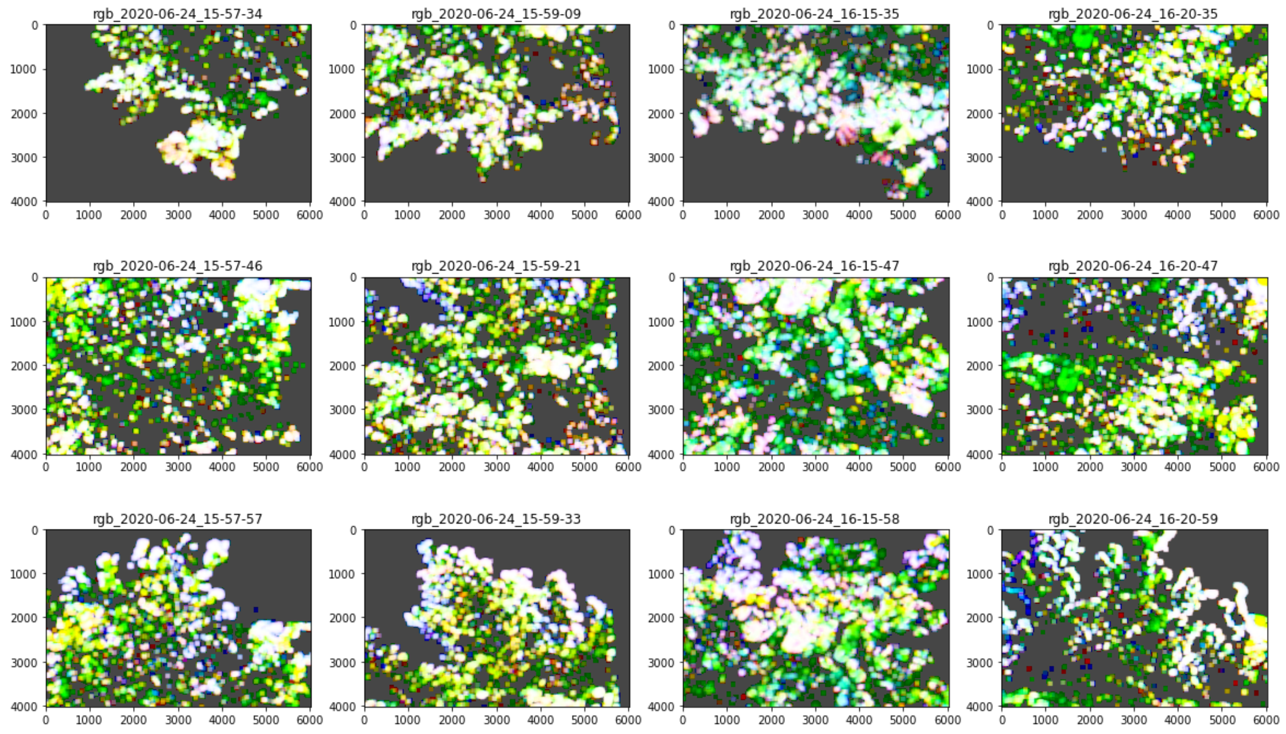

Figure 15: Synthetic images based on the enriched point clouds for the tree Yo_F3

3.2.2 Fixing extrinsic matrices with keypoints

The only way to fix this in postprocessing is to find ways to tie the point cloud geometry to the image geometry and adjust the extrinsic matrices of the images based on these tie points. We tested several feature detection algorithms, such as SIFT and SURF, but they did not provide useful results, since the matching between the synthetic laser scan images and the Sony RGB images is very difficult to achieve with traditional computer vision computer vision algorithms (see also (Sima & Buckley, 2013)). Therefore, we digitized key points manually, for each individual image, and calculated estimated matrices with a RANSAC approach. We were able to find keypoints in most of the images, but it was very labor intensive. The result however was not satisfactory. Figure 16 shows the fixed synthetic images for a capture position with three different pitches. The top row illustrates the problem very well. The synthetic image in the middle shows that the gimbal is set up to look directly at the tree, but the original image shows that it does not, it looks much further to the left. The fixed image on the right that after the key point-based adjustment it looks much better. In other examples however, it does not look as good. In the bottom row, which looks at the top of the tree the image geometry is adjusted too far to the right. In general, this approach is rather inconsistent. Figure 17 shows the normal visualized in 3D space. The original observations in red have always the exact same origin, which is most likely not true since the robot arm does not rotate exactly around the image sensor. The origins of the fixed extrinsics on the other hand, are very different, which also cannot possibly be correct. In general, basing the adjustment of the image extrinsics on a low amount of manually assigned tie points does not seem a robust approach to solve the geometry issue. An algorithmic approach is necessary, which, in contrast to SIFT, is able to deal with different image sensors, e.g., RGB and synthetic laser scan images. One approach could be to do some additional image processing steps, to make the synthetic and real images look more similar. These could be simple edge detection kernels, which would transform both images into more similar domains. Alternatives could be more robust deep learning-based techniques, which could potentially detect unique features by overfitting one sample artificially augmented with rotation, scaling, shearing and signal amplification techniques.

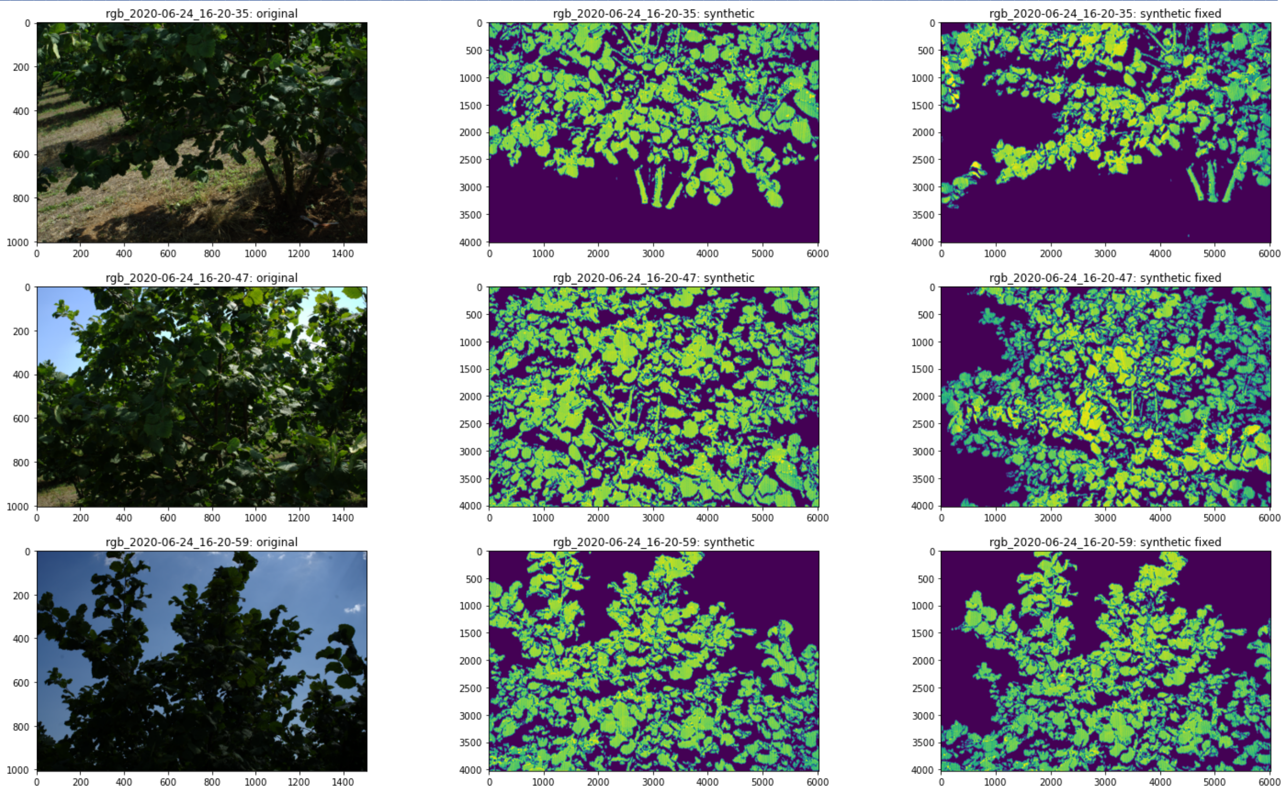
rgb_2020-06-24_16-20-35: original
rgb_2020-06-24_16-20-35: synthetic
rgb_2020-06-24_16-20-35: synthetic fixed
rgb_2020-06-24_16-20-47: original
rgb_2020-06-24_16-20-47: synthetic
rgb_2020-06-24_16-20-47: synthetic fixed
rgb_2020-06-24_16-20-59: original
rgb_2020-06-24_16-20-59: synthetic
rgb_2020-06-24_16-20-59: synthetic fixed

Figure 16: First capture position of tree Yo_F3, original and fixed extrinsics

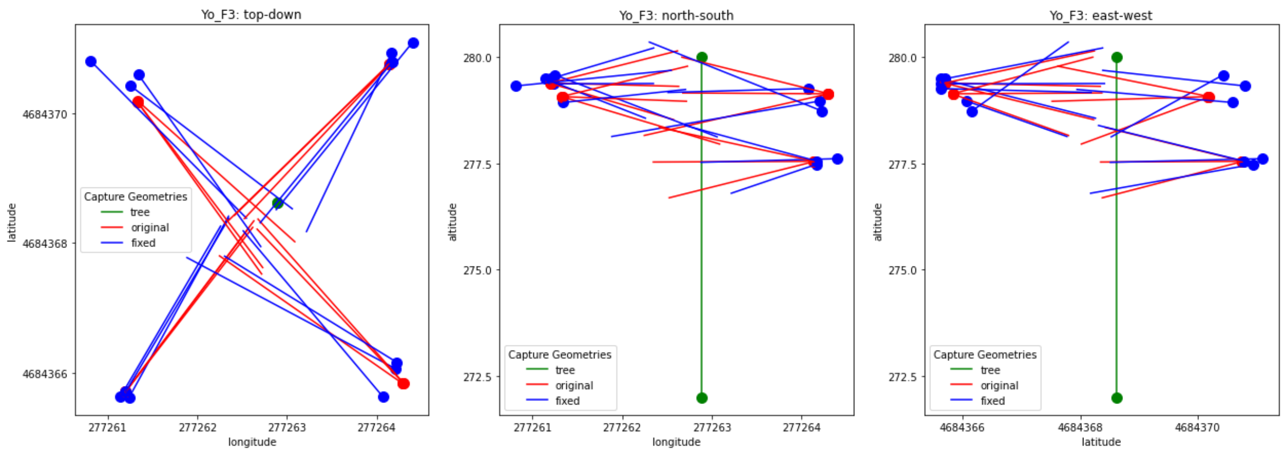

Figure 17: Yo_F3 capture normals, original and fixed

4 Results

4.1.1 Sample Level Performance

For the model building process, the 224 available images were split into training and validation data. For the validation we used the 117 images, 77 samples per image, that were matched to any of the fruit detection trees Yo_F1 – Yo_F10, which resulted in a total of 9009 sample patches. For model training, we used the 107 images that were not associated with any of these trees, to guarantee that the predictions are completely independent from the training data. The training process is visualized in Figure 18. Both training and validation accuracies quickly saturated around 99.5 %, which is mostly because of the class imbalance, i.e., the fact that the majority of pixels are not hazelnuts. The F-score is a much more suitable metric to asses model performance in image segmentation models. The training shows a much slower development of F-Scores compared to the total accuracies, but a rather steady increase which is still improving after 240 epochs. While the final result of 0.6915 in the training, and 0.5515 in the validation samples is not perfect, it clearly shows that the algorithm is well suited to detect these nuts. Furthermore, it should be noticed that there is still some potential for improving the model performance with: i) data augmentation, ii) a denser patch extraction, and iii) more training epochs. Future work will be focused on these directions.

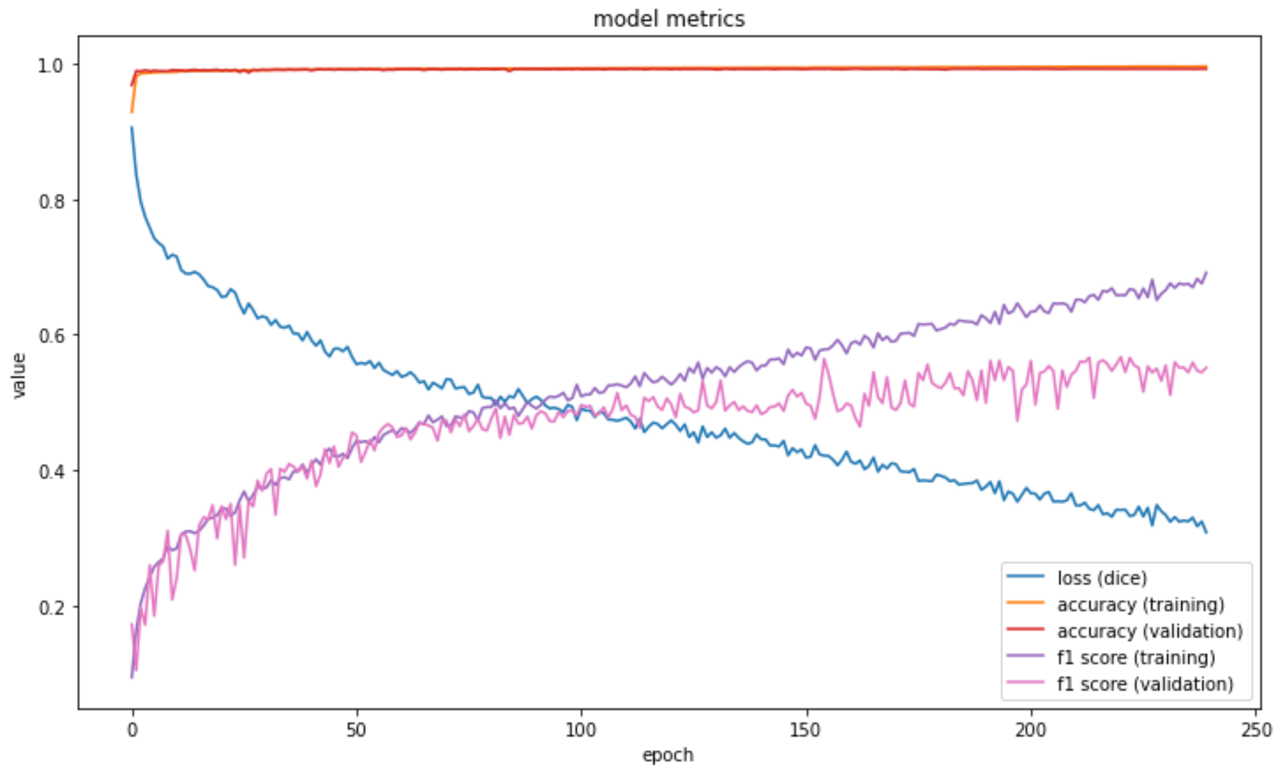

Figure 18: U-NET training

4.1.2 Image Level Performance

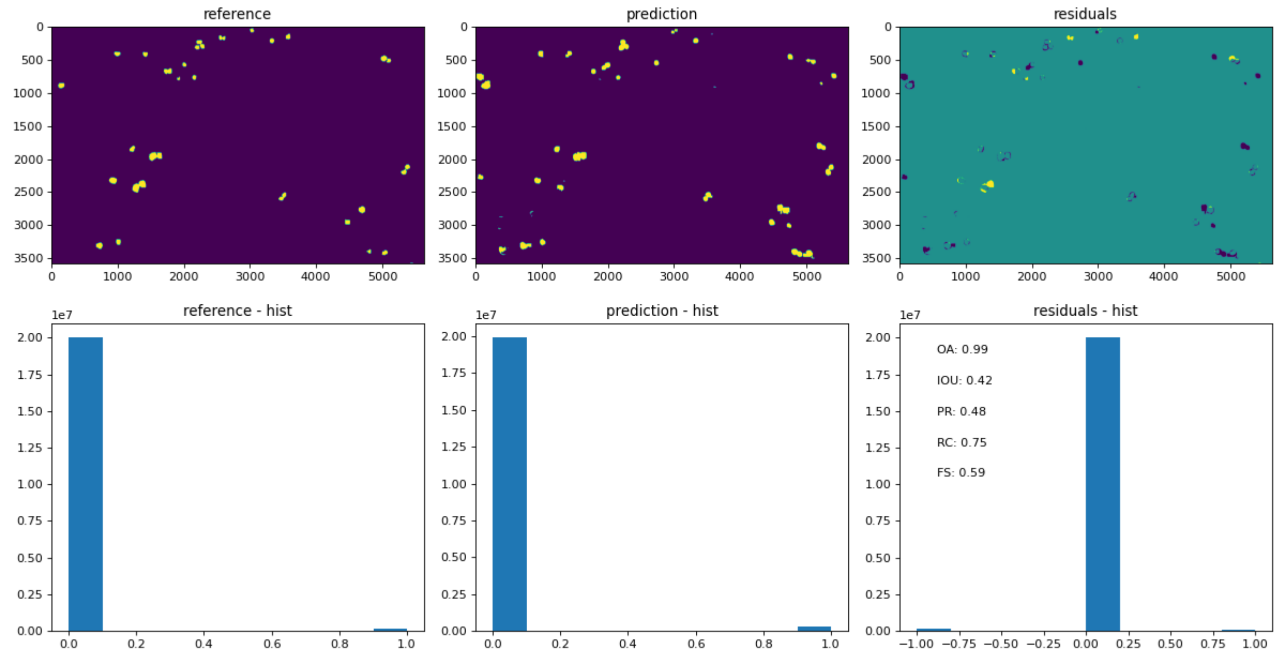

Figure 19: Image level Prediction: example with flagged pixels

Figure 19 and Figure 20 show the results of the prediction of one image on a pixel level, and on a cluster level. Here the masking step is not applied yet, since the clusters do not need to be associated with a tree yet, and it is more desirable to have as many clusters as possible in the images. We can see that the majority of clusters are detected, in this example there is a pixel-level recall of 75%. There is also some residual halo around the predicted nuts. This is not completely unavoidable because of the desirable overshooting of the manual labeling polygons. We see some error on both sides, false positives as well as false negatives, but they are very consistent in our data. As a result, the F-Score also shows a very promising consistency, thus numerically demonstrating that the overall approach works very well.

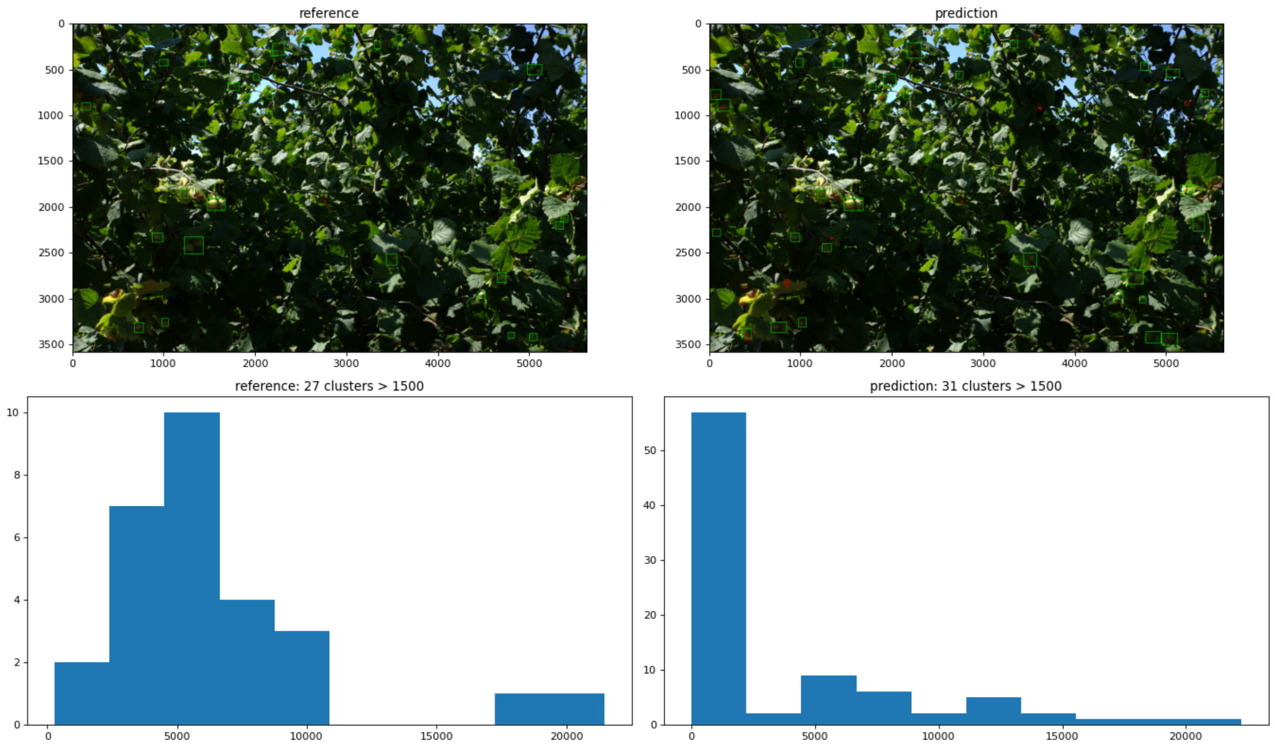

Figure 20: Image level prediction: example with detected clusters

While the image level performance measures are quite good, the real question is how well the image segmentation algorithm is able to predict the visible nuts on and average basis. For this reason, pixel and nut counts have been performed for labeled images of the campaign, and then again for all binary predictions. Figure 21 shows the correlation analysis of predicted and labeled images. It shows an overall very good agreement for both counted clusters and fraction of maize pixels. Both correlations show a substantially better fit with the training data of 95% and 97% of explained variance. There is a considerable drop in the correlation for the training images, but that is expected, and could be further improved with image augmentation and more training data in general. Still, it can be concluded that the U-Net image segmentation matches the human visual detection of hazelnuts very well.

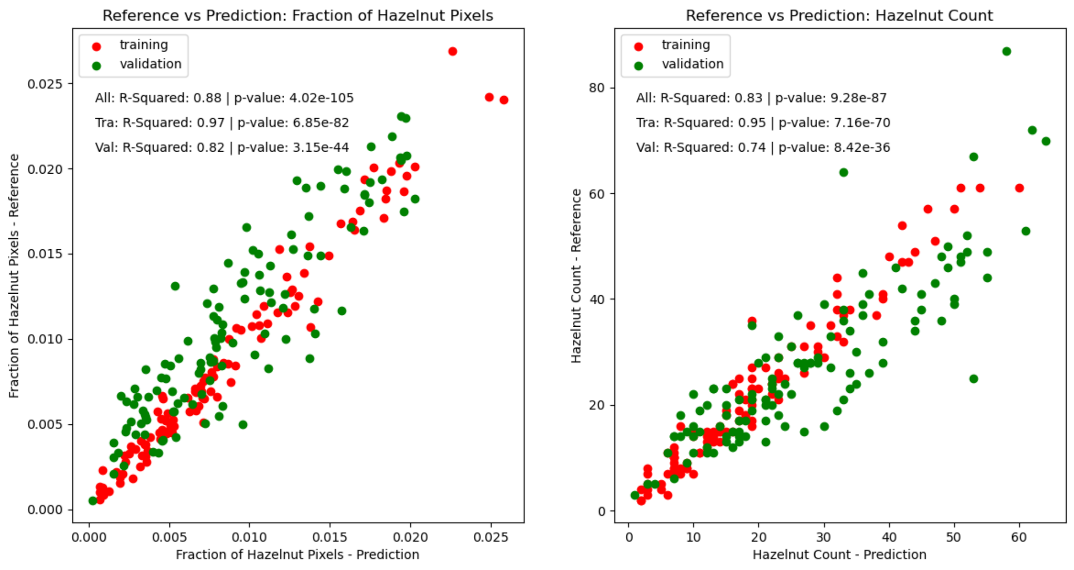

Figure 21: Image based Hazelnut Fraction and Number of clusters

4.1.3 Tree Level Performance

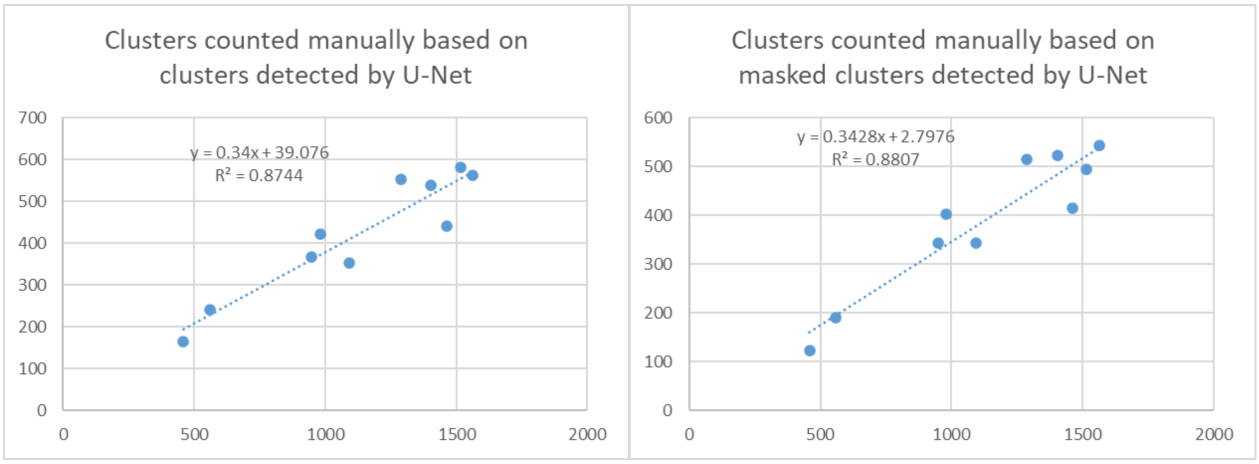

Figure 22: Relationship of counted nuts and detected visible nuts

The final step of the validation includes the matching of the now masked predicted images and nut counts to specific trees. In most cases these were 12 images per tree. In two cases it was more matched images, but this is not going to be a problem, since the tree in question will be mostly out of the image frame. The only real problem is if there are less capture positions. This has been the case for tree Yo_F1 which has only 9 observations from 3 capture positions. In this case, both the number of detected clusters in the images, and the number of counted clusters on the tree were overall the lowest, so this problem does not have a big influence on the overall model. Figure 22 depicts the end result of the analysis. In particular, it shows that the image-based fruit detection underestimates the number of clusters by a relatively constant factor of 3.4. Indeed, this underestimation is expected, since a large number of clusters is covered by leaves and branches, and therefore not visible. Nonetheless, being this constant, it can be easily taken into account within a linear regression modeling, i.e., by considering a multiplicative coefficient of 3.4. The overall fit of the cluster prediction model is very good, while the tree masking improves it only marginally. This is likely because out of focus trees were only present to a very small extent. Overall, the number of trees analyzed is relatively small. More data from 2021 is available and is currently analyzed, to tell how well this factor would translate to a different year and different growing conditions.

5 Conclusions and comments

The image segmentation algorithm worked perfectly well, and it could be improved with data augmentation and more samples to make in more robust to structural changes in the state of the tree phenotype, such as different time of image acquisition, different tree varieties, or visible damage due to frost or other weather extremes. The segmentation performance transferred very well to out of training samples from the same orchard, with a substantial but relatively small loss of the ability to detect visible hazelnut clusters. In addition, the transfer to tree level works reasonably well, and the detection of visible clusters underestimates the number of overall clusters on a tree by a constant factor, which as pointed out before can be easily taken into account within a linear regression modeling.

Unfortunately, the point-cloud based detection could not be implemented. We will continue to find a robust solution to the image alignment problem, since a point-cloud segmentation would be a much more generalizable approach to fruit detection. It would directly solve the problem of underestimation due to feature occlusion, and would also be very future proof, as RGB laser scanners become more and more important for precision agriculture. We will try alternatives to traditional feature extraction techniques like SIFT and SURF, to find a large number of robust tie points, to increase the precision of the observer geometry up until a point where spectral enrichment is feasible.

6 References

Bradski, G. (2000). The openCV library. *Dr. Dobb's Journal: Software Tools for the Professional Programmer*, *25*(11), 120–123.

Gilcher, M., & Udelhoven, T. (2021). Field geometry and the spatial and temporal generalization of crop classification algorithms—a randomized approach to compare pixel based and convolution based methods. *Remote Sensing*, *13*(4), 1–20. https://doi.org/10.3390/rs13040775

Häni, N., Roy, P., & Isler, V. (2020). A comparative study of fruit detection and counting methods for yield mapping in apple orchards. *Journal of Field Robotics*, *37*(2), 263–282. https://doi.org/10.1002/rob.21902

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems*, *25*, 1097–1105. https://doi.org/10.1201/9781420010749

Lecun, Y., & Bengio, Y. (1995). Convolutional Networks for Images, Speech and Time-Series. In *The handbook of brain theory and neural networks* (Vol. 1, Issue 1). https://doi.org/10.1177/016555157900100111

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *9351*, 234–241. https://doi.org/10.1007/978-3-319-24574-4_28

Sasaki, Y. (2007). The truth of the F-measure. *Teach Tutor Mater*, 1–5. http://www.cs.odu.edu/~mukka/cs795sum09dm/Lecturenotes/Day3/F-measure-YS-26Oct07.pdf

Sima, A. A., & Buckley, S. J. (2013). Optimizing SIFT for matching of short wave infrared and visible wavelength images. *Remote Sensing*, *5*(5), 2037–2056. https://doi.org/10.3390/rs5052037

Zak, K. (n.d.). *Keras Unet*. https://github.com/karolzak/keras-unet