AGR-o-RAMA  
Agricoltura di precisione con Robot Autonomi per il Monitoraggio Attivo

Deliverable D1.1  
Modulo di simulazione foto-realistico 3D per campi coltivati  
WP1

**Informazioni sul documento**

| **Codice Progetto** | |  | | **Acronimo** | | AGR-o-RAMA | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Titolo** | | | | Agricoltura di precisione con Robot Autonomi per il Monitoraggio Attivo | | | |
| **Bando** | | | |  | | | |
| **Data di avvio** | | 15/04/2021 | | **Durata** | | 24 mesi | |
| **URL di progetto** | | | | www.agrorama.it | | | |
| **Responsabile di Progetto** | | | | Vito TRIANNI | | | |
| **Deliverable** | | | | D1.1 | | | |
| **Work Package** | | | | WP1 | | | |
| **Data consegna** | **Contrattuale** | | M9 | | **Attuale** | | M9 |
| **Natura** | | Report | | **Disseminazione** | | Confidenziale | |
| **Beneficiario Responsabile** | | | | Sapienza | | | |
| **Autore principale** | | | | Carlos Carbone | | | |
| **Co-autore(i)/Revisore(i)** | | | | Si veda tabella seguente | | | |

**Cronologia delle revisioni**

| Versione | Data rilascio | Satus | Cambiamenti | Autore |
| --- | --- | --- | --- | --- |
| 0.1 | 10/01/2022 | Bozza |  | Vito Trianni |
| 1.0 | 15/02/22 | Finale |  | Carlos Carbone |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

## 

## Introduction

To ensure a realistic robot perception simulation in precision agriculture we require a photo-realistic environment that can procedurally spawn plants within an agricultural field. Furthermore, the generated synthetic imagery of the simulated field must be validated using a state of the art method for image classification including comparison with real imagery, i.e. a CNN.

This section serves to validate the photo-realistic capabilities of Unity to simulate plant imagery obtained with *UAVs*. Furthermore, additional experiments for image noise validations are included.

Potato and sugar beet plants are used for the experiments in this section which are available in images taken with a *UAV* in a sugar beet field and a potato field in Wageningen in the Netherlands. As the plants are presented separately in the fields, so they will be simulated to validate the photorealism of the simulation. Then, the plants will be combined inside Unity for the noise validation experiments. This simulates the scenario where sugar beets are grown in a field after the potato season, thus some undesired potatoes plants grow as weeds within the sugar beet field.

## Description of the simulator

The built photo-realistic simulation environment in this project is based on the video game engine Unity 3D. During the first stages of this research, some testing was done comparing the video game engine Unreal Engine 4, hereinafter referred to as "Unreal", and Unity 3D, hereinafter referred to as "Unity". We decided to use Unity because it presented better compatibility with Linux operative system, which is needed for integration with the rest of our developed tools based on the *Robotic Operating System (ROS)*. Furthermore, Unity provides a rendering engine with enough fidelity to achieve a photo-realistic result while having less demanding computational requirements than Unreal.

With Unity we are able to procedurally simulate multiple types of plants and generate synthetic imagery to train *CNNs* and achieve a performance similar to the one achieved with training on real images. We train *CNNs* with both types of data separately and then use each trained version to classify an allocated set of real images which are used only for classification and evaluation after training. Our main evaluation *Key Performance Indicator (KPI)* is the *Intersection Over Union (IoU)* which is mainly dependent on the true positives for detection of weeds, providing a clear comparison metric. Furthermore, the *CNN* classification methods are mainly divided into object detection (detection of entire weeds in images) and pixel semantic segmentation (pixels correctly classified).

The results provided a strong validation for Unity as a tool to reliably simulate realistic plants which serves to generate training imagery for *CNN* and *UAV* perception simulation.

#### **Potatoes**

A royalty free picture with high quality is used to extract the textures to simulate the potatoes in Unity. The image used for this is presented in Figure 1, which provides enough pixel data with minor shadow interference and leaf overlapping. Gimp is then used to extract the potato leaf textures in a similar way to the previous simulated plants. Texture samples are presented in Figure 2 which have higher pixel density compared to the leaves present in the real images taken from the *UAV* Figure 3.

While the approach to get the leaf textures ensures a high quality leaf, the ground textures needed to be extracted from the *UAV* image from Figure 3 which provided an image big enough to extract the ground textures. Figure 4 shows the ground texture cropped from the potato field ground and its tiled version used then in Unity to create the ground.

With the textures of both the leaves and the field ground, it is possible to spawn the plants and the simulated field in unity using a procedural code to build the structure of the potatoes. Figure 5 shows a sample of both a real potato plant and a potato plant generated in Unity. As the figure shows, the potato structure has multiple root stems from which the leaves spawn almost fully perpendicular in pairs, with a final leaf at the end of the stem. A simple polygon with a green color is bent to form the stem and possible leaf poses are saved across the stem to spawn all the leaves placing a bent polygon with the previously cropped textures on top. With the plant structure ready, the field is spawned with a row structure with some noise on plant positioning, rotation and size. Figure 6 shows a sample comparison of a real potato field and a Unity potato field. The real image is not used for training and only as reference to setup the spawning parameters in Unity to replicate the field structure. There is a difference in amount of plants per image since the Unity images had a different resolution. However, the selected resolution for the Unity images matches considerably the plant pixel density of the real images that are actually used for training. Furthermore, since the *Point of view (POV)* is a desired feature in the swarm simulation chapter, here it is included in the experiments. This means that the images taken from the *UAV* are cropped depending on the positioning of the *UAV* from the plant position of interest. Therefore, the full image from Unity here has a resolution of 6144x6144 (WxH) pixels, however if no *POV* noise is included then only the central portion of the image is cropped obtaining a final image of 2048x2048 (WxH) which is the one present in Figure 6b.



Figure 1: Reference image to extract the potato textures [1].

With the textures of both the leaves and the field ground, it is possible to spawn the plants and the simulated field in unity using a procedural code to build the structure of the potatoes. Figure 5 shows a sample of both a real potato plant and a potato plant generated in Unity. As the figure shows, the potato structure has multiple root stems from which the leaves spawn almost fully perpendicular in pairs, with a final leaf at the end of the stem. A simple polygon with a green color is bent to form the stem and possible leaf poses are saved across the stem to spawn all the leaves placing a bent polygon with the previously cropped textures on top. With the plant structure ready, the field is spawned with a row structure with some noise on plant positioning, rotation and size. Figure 6 shows a sample comparison of a real potato field and a Unity potato field. The real image is not used for training and only as reference to setup the spawning parameters in Unity to replicate the field structure. There is a difference in amount of plants per image since the Unity images had a different resolution. However, the selected resolution for the Unity images matches considerably the plant pixel density of the real images that are actually used for training. Furthermore, since the *Point of view (POV)* is a desired feature in the swarm simulation chapter, here it is included in the experiments. This means that the images taken from the *UAV* are cropped depending on the positioning of the *UAV* from the plant position of interest. Therefore, the full image from Unity here has a resolution of 6144x6144 (WxH) pixels, however if no *POV* noise is included then only the central portion of the image is cropped obtaining a final image of 2048x2048 (WxH) which is the one present in Figure 6b.



(a) Potato leaf texture sample 1. (b) Potato leaf texture sample 2. (c) Potato leaf texture sample 3.

Figure 2

#### **Sugar beets**

For the sugar beet generation an approach similar to the sunflower plants is taken to obtain the textures. Even if high quality sugar beet textures are already available it is difficult to replicate the reflecting lightning conditions observed in the real images from the *UAV* point of view. From the 20 m altitude in the real images, the sugar beets presented white reflective spots similar to the ones observed above water from high altitudes. While this is not impossible to simulate, it is not achieved with enough accuracy during the experiment setup. Therefore, cropped textures from a real image not used for the experiments is used for plant simulation. This is a reasonable approach since textures with enough variation are expected to be gathered before building the simulation environment. The plant structure is exactly the same as the previous sugar beet simulation experiments. Figure 7 presents a comparison between a sample from the sugar beet real data for training and a sugar beet generated in Unity with the extracted textures. Figure 7, shows then a sample sugar beet field comparison sample with also a row structure similar to the potato fields. The resolution conditions for the images are the same as in the potato case.

#### **Background polygon addition**

The real images available from the *UAV* data gathering are provided within the software Agisoft along with the box labeling of the plants. This labeled data was provided during collaborative work. The output images in the software provided real labeled images but also included some grey background cuts, Figure 9a. Therefore, a grey polygon is added on top of the Unity generated field images for training, Figure 9b. The labels from the plants covered by the generated grey polygon are discarded, and the intersecting boxes are discarded if any corner of a box with size 0.25 of the original box is inside the grey area. Additionally, the Unity images are generated with a high amount of pixels during these experiments to match the plant pixel density of the real images that included the grey background, the defined values are later presented in the Datasets section.

#### **Sugar beets combined with potatoes**

For the noise experiments the fields contained both sugar beets and potatoes simultaneously as this could be easily achieved in the simulated environment. The objective is to simulate a sugar beet field where potatoes would spawn as weeds close to the sugar beet rows. The potato plants are spawned with the same conditions as previously presented, and the sugar beets plants are spawned with the high quality textures this time. This would ensure that the pixel density per plant is considerably close, thus the *CNN* can not rely on distinguishing the plants base on texture quality but instead on their shape and visual features. Figure 10 presents a field sample image with both plants included in the disclosed field structure.



Figure 3: Sample image from *UAV* used to extract ground texture.

The objective of these experiments is also to demonstrate how including noise in the Unity images influences the *CNN*. The desired noises to study are:

• Motion blur: Pixel fuzziness. While Unity can simulate motion blur it only does it if the camera is in movement so to have further control over this noise it is added with a python script after the images are generated from Unity.

• Camera rotation: Rotation of field in image based on field and *UAV* orientation. This is simulated by rotating the camera in the simulated environment.

• *POV*: Points of view from the camera to the desired position to gather images. This is simulated by positioning the *CNN* in 8 additional possible locations around the center of the plant area of interest. These 8 locations have the same altitude and have an offset to form a 3x3 grid cell shape around the center cell position which is exactly above the area of interest. During simulation the camera is offset to each location and the gathered image is cropped to match the area of main plant interest in the center cell again.

• Altitude: *UAV* altitude. This is simulated by changing the camera altitude in the simulation environment.

• Light intensity: Solar illumination depending on the weather and day time. This is simulated by changing the intensity of the simulated directional light that acts as the sun in the simulation environment.

• Light rotation: Solar illumination orientation depending on time of day and earth positioning. This is simulated by changing the orientation of the simulated directional light that acts as the sun in the simulation environment.



(a) Potato ground texture cropped. (b) Potato ground texture tiled.

Figure 4: Potato ground field cropped from image 3



(a) Potato plant generated in Unity (a) Potato plant available from a real image.

Figure 5: Potato plant samples.



(a) Potato field available from real data (not used for *CNN* training).



(b) Potato field generated in unity.

Figure 6: Potato field samples at 20 m altitude.



(a) Sugar beet plant available from a real image. (b) Sugar beet plant generated in Unity.

Figure 7: Sugar beet plant samples.

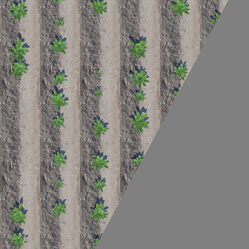
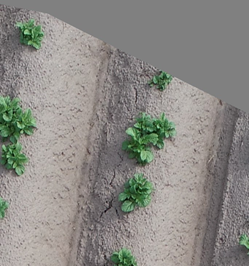


(a) Sugar beet field available from real data (not used for *CNN* training).



(b) Sugar beet field generated in unity.

Figure 8: Sugar beet field samples at 20 m altitude.



(a) Real labeled image of potato field. (b) Unity generated image with grey polygon background added.

Figure 9: Potato final field image samples.



Figure 10: Field image sample with both potato and sugar beet plants combined at 20 m altitude.

### **Datasets**

Multiple datasets for experiments are generated with the developed the code and obtained textures to spawn the all procedural fields. Two objectives are established dividing the experiments in two parts:

1. Realism: Set of experiments to validate the photo-realistic capabilities of Unity to simulate *UAV* imagery. This is achieved mainly by comparing the *CNN* performance on classifying real data after being trained on real data and then on data generated with Unity, similarly to the previous Unity validation experiments.

2. Noise: Set of experiments to demonstrate how imagery noises can influence *CNN* performance. This is achieved by training the *CNN* on Unity generated data and classifying Unity generated data with different types of noise.

The noises included for simulation are summarized Table 1. These include the most common noises that can be present during data collection with *UAVs*. Their maximum ranges are based also on possible conditions where the sensing equipment has low quality or an anomaly takes place during the mission. Furthermore, the columns presents how simulated noises are grouped into three categories:

• 0R: Means zero noise.

• ∼R: Means noise similar to the one observed in the real data available.

• +R: Means extreme noise that may occur with very low quality sensing.

The camera altitudes are selected considering that the real images available were taken at 20 m of altitude. Camera rotation covers the case that a field may be presented in any orientation depending on the drone flight path. Light intensity ranges cover the case with bad camera exposure. Light rotation considers solar lighting angle ranges that may occur during a day for any field orientation. Motion blur goes from no motion blur to the point that the plants are difficult to recognize by the human eye, it may take place with a bad camera or taking a snapshot during a quick sudden move by the *UAV*. Finally, the *POVs* cover imagery that is taken out of the center of the snapshot, meaning that the *UAV* taking the image is not directly above the plant.

The noise groups presented in Table 1 are used to generate training in Unity. Furthermore, the noise ranges in +R are used separately to generate evaluation datasets which for each noise type to present which makes the *CNN* performance to vary the most.

The data distribution for the photo-realism validation experiments is presented in Table 2 and

3. In these tables the label letters have the following meanings:

• Image type: R = real, U = unity image.

• Image purpose: T = images used for *CNN* training, E = images used for evaluation experiments.

• Plants in image: P = image contain only potatoes, S = image contain only sugar beets, C = image contain both potatoes and sugar beets.

• *CNN* role: T = training image, V = hyper parameter validation images, E = image for classification evaluation.

With the dataset image distribution of both Table 2 and 3, the *CNN* is trained separately using each of the training dataset image distribution per plant. The amount of Unity images generated is considerably higher than the real images because during the experiments this is necessary to obtain enough plants variation for the *CNN* to generalize enough to properly classify the real images during the Realism experiments. Then, during the Noise experiments less images are required to obtain a good performance classifying Unity images without noise to present a baseline.

The pixel dimensions per image are also an important factor to obtain a good *CNN* performance with a good pixel density per plant, thus they are distributed as follows:

• Real: 684x729 (WxH).

• Unity images for realism experiments: 2048x2048 (WxH).

• Unity images for noise experiments: 1024x1024 (WxH).

The Unity noise experiment images had a lower pixel density to simplify the noise experiments since the evaluation data is also generated in Unity and the pixel dimensions from the realism experiments took a long time to generate.

### **Object detection network**

The main task in this set of experiments is to evaluate and compare the generated dataset using plant detection. To perform the mentioned task, an existing convolution neural network architecture for object detection based on YOLO [2] was provided during the work collaboration. Therefore, this subsection briefly explains the architecture of the used *CNN* but it does not present it in detail as it is not part of the contribution in this project.

The architecture of the YOLO is considered as one of the state of the art for one stage detectors. The implemented *CNN* uses the latest version of YOLO which is considered the faster and more accurate version. Additionally, YOLOv5 has a compound scaling which was popularized in the EfficientNet research [3], where the depth is scaled based on the input size of the network and which also provides more accurate and flexible models.

The YOLO architecture consist of three main parts:

• Backbone - A convolutional neural network that collects and forms image features at different levels, and usually we use a pre-trained model on ImageNet dataset.

• Neck - A sequence of layers to merge image features and pass them to prediction.

• Head - use features coming from the neck and predict the bounding boxes and the classes of the objects.

Similarly to bonnet, the *KPI* used to evaluate and compare the performance between different datasets of sugar beet and potatoes on detection is the *IoU*. In this case the pixels classified can have the following result:

• True Positive (TP): bounding boxes correctly detected (*IoU > x* ).

• False Positive (FP): bounding boxes wrongly detected ( *IoU < x*).

• False Negative (FN): bounding boxes not detected where the plant are present.

Table 1: Parameters for synthetic training.

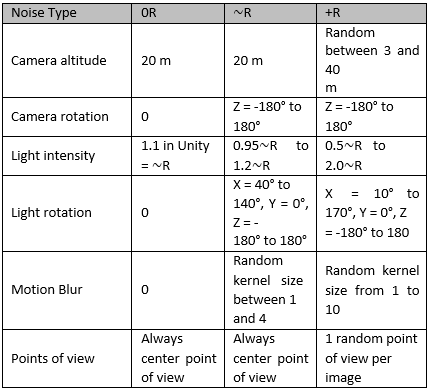


Table 2: Dataset image distribution for Unity photo-realism validation.

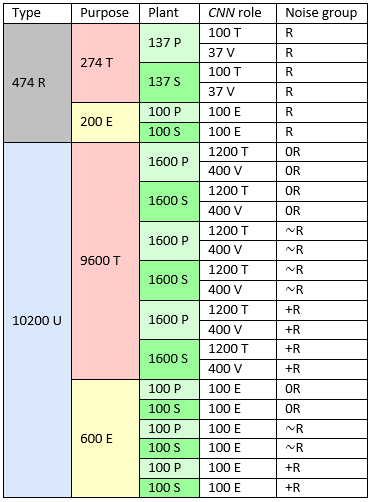
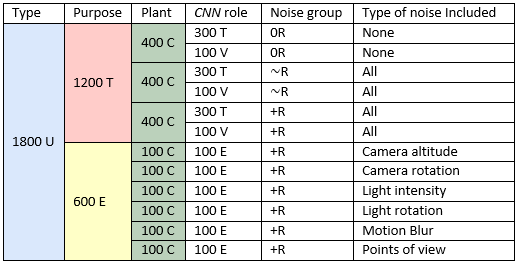


Table 3: Dataset image distribution for noise experiments.



### **Results**

Additionally, for the realism experiments in Table 2, two datasets are used with the already established data:

• Unity ∼R small: Dataset with images from the Unity ∼R section, but only taking 100 images for training and 37 for validation as the real data for training. This dataset presents the *CNN* performance using the same amount of noise and amount of images of the real data.

• Augmented: Dataset combining the training and validity images from the real and the Unity +R data for training. This dataset presents how the *CNN* performance of both best performing datasets combined could potentially present the best possible performance, similarly to previous Unity validation experiments.

The results for the realism experiments are presented in Figure 11, 12, 13, and 14. Each figure represents the result over classifying the evaluation images per noise group from Table 2, the evaluation dataset used is specified in the title of the graph. The result values are presented in the Y axis which represents the *IoU* of the classification over the current evaluation dataset in the graph. Results over both plants are presented simultaneously by color and each column group represents the training data from table 2 used to train the *CNN* to classify the current evaluation dataset in the graph. Each training dataset result in the graph is divided by best and last results, this means doing the classification over the evaluation dataset using the best *CNN* weights over the validation images and using the last *CNN* weights during training.

The results for the noise experiments are presented in Figure 15, 16, and 17, 18. The images contained both plants during training and classification, but here are presented in separate graphs for easier visualization on how the noise affected the *CNN* behavior, Figure 15 and 16 present the results for the potatoes and Figure 17 and 18 present the results for the sugar beets. The graphs that are labeled as "part 1" contain the first section of noise classification results from Table 3, while the graphs that are labeled as "part 2" contain the rest. Notice that the light intensity values are divided into 4 values going from the lowest to the highest value from Table 1 in an uniform manner.

#### **Additional noise experiments and results**

Before presenting the discussion for the results, additional tests for noise are made to demonstrate the change in performance by including each of these noises isolated during training. The purpose of these is to demonstrate how adding the most prominent noises from Figure 15 - 18 in an isolated manner during training can directly overcome each said noise during classification.

The amount of training images distribution is equivalent to the one present in Table 3, that is 400 images per training and the noise ranges used during training correspond to the +R noise range group from Table 1, with the exception of the altitude noise. The altitude training dataset is generated with a higher image count and the noise range is higher (3-50 m of altitude) in an attempt to overcome the clear trend in Figure 16 and 18, which is low performance on the high ends of the noise range.

The evaluation datasets per noise are exactly the same from the previous noise experiments in Table 3. The classification results are presented in Figure 19 and 20, presenting the results for the sugar beets and potatoes respectively in the same way as the previous noise experiments.

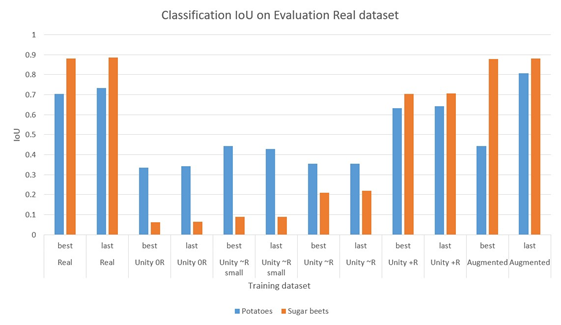


Figure 11: Results for the realism experiments after evaluating real images available for evaluation in table 2. The X axis represents which training dataset is used for training of the *CNN*.

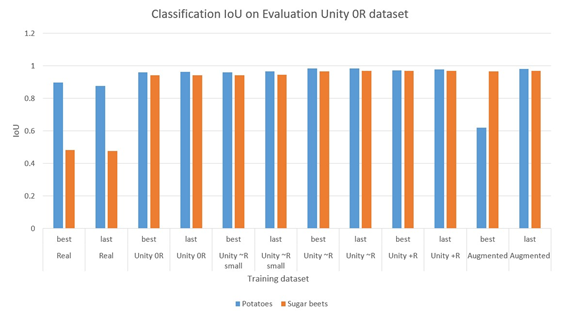


Figure 12: Results for the realism experiments after evaluating Unity images with noise 0R available for evaluation in table 2. The X axis represents which training dataset is used for training of the *CNN*.

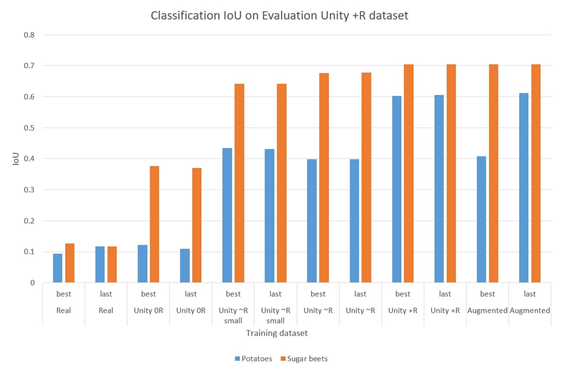


Figure 14: Results for the realism experiments after evaluating Unity images with noise +R available for evaluation in table 2. The X axis represents which training dataset is used for training of the *CNN*.

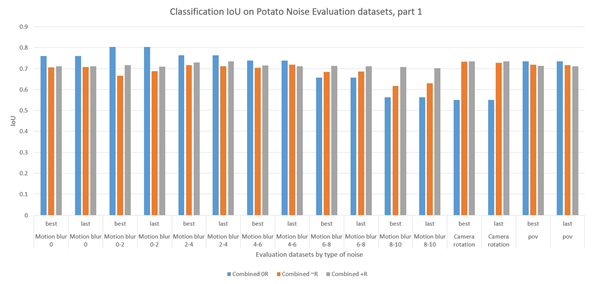


Figure 15: Part 1 of results for the noise experiments after evaluating potato images available for evaluation for each type of noise in Table 3. The X axis represents which type of noise is included in the evaluation.

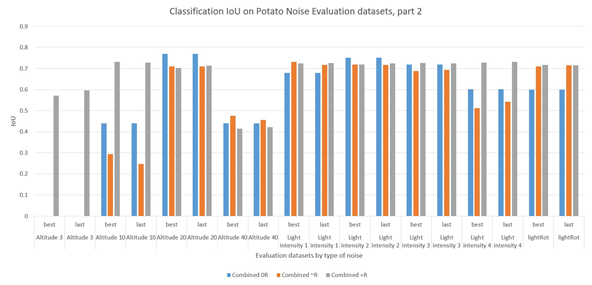


Figure 16: Part 2 of results for the noise experiments after evaluating potato images available for evaluation for each type of noise in Table 3. The X axis represents which type of noise is included in the evaluation

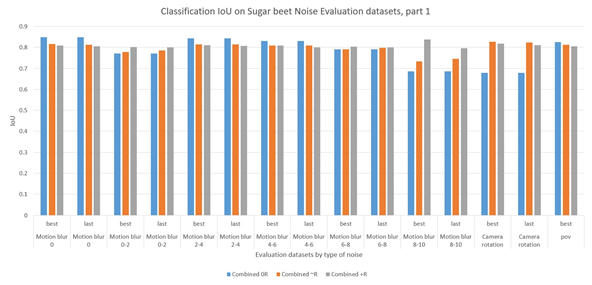


Figure 17: Results for the noise experiments after evaluating sugar beet images available for evaluation for each type of noise in Table 3. The X axis represents which type of noise is included in the evaluation.

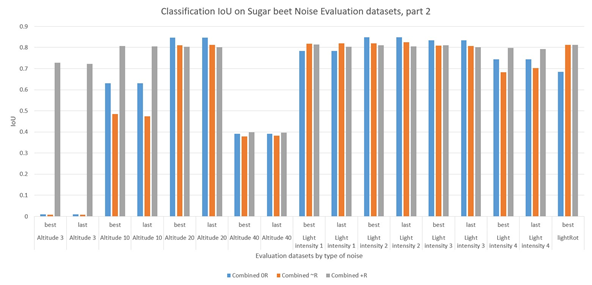


Figure 18: Results for the noise experiments after evaluating sugar beet images available for evaluation for each type of noise in Table 3. The X axis represents which type of noise is included in the evaluation.

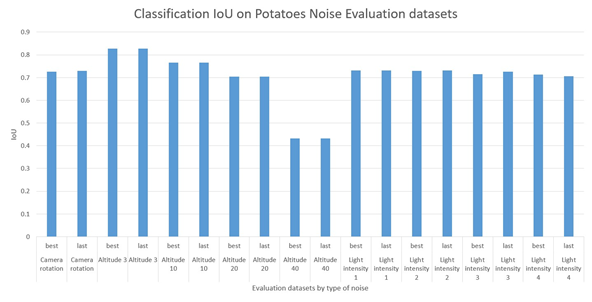


Figure 19: Results for the noise experiments evaluating potato images available for evaluation for each type of noise in Table 3 after including that type of noise in training. The X axis represents which type of noise is included in the evaluation.

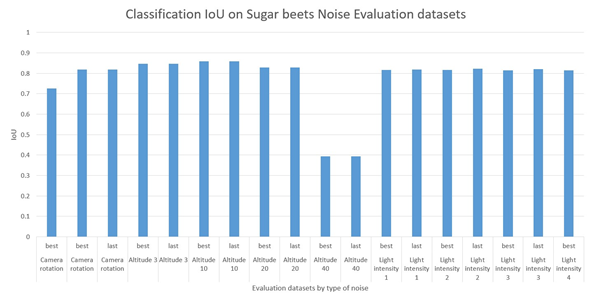
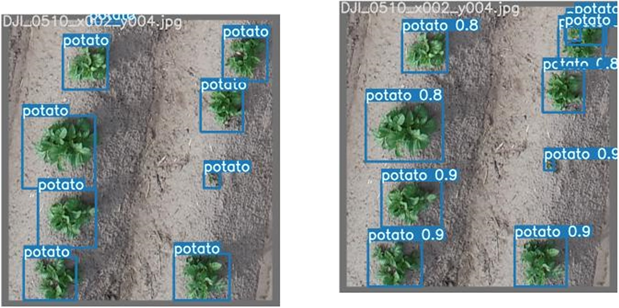


Figure 20: Results for the noise experiments evaluating sugar beet images available for evaluation for each type of noise in Table 3 after including that type of noise in training. The X axis represents which type of noise is included in the evaluation.



(a) Potato real image label sample. noise Unity images. (b) Potato *CNN* prediction after training with +R

Figure 21: Potato final field image samples.



Figure 22: Close up view of image sample from the 40 altitude evaluation images from Table 3.

### **Discussion**

The results from Figure 11 demonstrate that Unity is capable of generating the *UAV* camera imagery that can serve to reliably train a *CNN* for real plant detection and achieve a performance similar to the previous Unity validation experiments. However, it is also very clear that high amounts of simulated noise are required including during training with Unity data to achieve a reliable performance, even if noises are mildly present in the real data. Furthermore, it is clear that there is a difference in performance between the potatoes and the sugar beets. This is due the imprecise labeling of real data, which is the network does not consider after training. The available amount of labeled real potato images is higher than the sugar beet ones, so the best labeled ones are selected to try to achieve the most realistic scenario possible, achieving the previously presented results. Figure 21 shows a sample of an image that is not included in any of the experiments but it is classified for and added here for demonstration purposes. It can be clearly seen that the label boxes are not fully precise and in some cases it multiple potatoes as a single one (shown in the upper right corner of the image), which are further identified by the *CNN*, but some false positives are present in this case.

For additional realism demonstrations, Figure 12, 13 and 14 present the classification results over classifying synthetic data which demonstrate how the *CNN* interprets the synthetic data also as plants. This includes the cases where the *CNN* is trained with real data which achieves a good detection performance when no noise is included. When noise is added it is more difficult for the *CNN* to achieve reliable classification performance as it would be expected. Furthermore, all cases demonstrate that the best possible performance overall is achieved by combining best performing Unity training data with the real data, similarly to the previous validation experiments.

Once that Unity photo-realism is validated, the noise experiments demonstrate how each noise individually influences the *CNN* performance the most. The most noticeable cases are the altitude, camera rotation and light intensity where the altitude and light intensity have the most impact on the values of the +R noise range. Similarly to the realism experiment, the best performance overall from the *CNN* is achieved by including the +R noise during the training with a few exceptions when the noise in the evaluation is low which sometimes gets higher classification performance by the *CNN* trained with 0R noise data.

To further demonstrate Unity noise generation utility, the short experiment presented in Figure 19 and 20, attempts to overcome each respective noise by including the isolated noise in the training data. The camera rotation and light intensity results are easily enhanced with a similar performance to the one presented by the network trained with all +R noises. However, the altitude noise data included is not able to overcome low performance when 40 m altitude images are presented in the evaluation data. This is mostly due to the plant quality that is presented at this altitude in Unity, where not every part of the plant is visible anymore, thus it is difficult for *CNN* the *CNN* to detect some of the small plants, so many are missed as false negatives. Figure 22 shows a zoomed sample of a Unity generated image at 40 m altitude which clearly shows how the plant features quality is highly deteriorated because of the low plant pixel density with the current camera settings that are used to match the real data pixel density data at 20 m altitude.

## Conclusions

This chapter presents several validation experiments for agriculture visual photo realism provided by Unity. It is clear after all the results presented demonstrate that Unity can serve to reliably simulate photo-realistic imagery for precision agriculture. The data provided can serve to train a *CNN* and achieve very reliable performance compared to training with real data, even in cases with high noise at high altitudes which are present in *UAV* mission cases. Furthermore, the experiments demonstrate how Unity can be used to generate plant imagery with controlled noises. This is highly useful as noises in real world are unpredictable and difficult to replicate. Undesired bad equipment or environment conditions might be required to replicate these noises, for example by having a bad camera exposure under specific sun light intensity conditions which is a prominent noise in the results.

It is important to note that the *CNN* performance from the ground robot perspective does not necessarily translate to the performance with *UAV*. Moreover, a network trained for ground robots might not perform well on a *UAV* and vice versa, since the camera altitude noise is one the biggest factors that influence the *CNN* performance. However, high noise inclusion during training shows promising resilience to image variation that may come from deploying the *CNN* in different robots or type of missions. Therefore, additional noise type and ranges could be further included during simulation to ensure a *CNN* is capable of performing in more than one type of robot or mission. This resembles a case when a heterogeneous multi-robot system is deployed for multiple types of precision agriculture missions. Furthermore, additional type of crops can also be included within Unity to simulate perception in more complex situations like plants and weeds occluded by bigger plants, remnant leaves, mud or rubble from other activities within the field and its surrounding areas.

Unity presents a better ease of use in Linux where *ROS* is mostly used to develop robotic applications. With the results demonstrated, we considered Unity as the most appropriate candidate to handle our perception simulation of *UAVs* in precision agriculture research.

Further research can still be done for less general but common situations like fields with varying and unpredictable levels of water, moisture and background objects like tree leaves and rocks. However, these cases are beyond the scope of this research as only validation of aerial swarm robotics strategies in common precision agriculture setups are tested and validated.

1. W. Commons. File:potato leaves.jpg — wikimedia commons, the free media repository, 2020. [Online; accessed 28-December-2021].
2. M. Rahnemoonfar and C. Sheppard. Deep Count: Fruit Counting Based on Deep Simulated Learning. Sensors, 17(4):905, Apr. 2017.
3. A. Tahir, J. Böling, M.-H. Haghbayan, H. T. Toivonen, and J. Plosila. Swarms of unmanned aerial vehicles — a survey. Journal of Industrial Information Integration, 16:100106, Dec. 2019.