agricultural engineering.eu 81(1), 2026, 24–39

DOI:10.15150/ae.2026.3358

agricultural engineering.eu

# Using synthetic images to train neural networks for tramline detection

**Silko Schulpius, Jan Schattenberg, Ludger Frerichs**

In recent years, the ability to generate synthetic image data and the quality of such data have significantly improved. This advancement facilitates the training of neural networks, which require large amounts of training data. The ability to train neural networks more quickly and easily provides significant benefits for precision farming applications, particularly in auto-steering situations where Global Navigation Satellite System (GNSS) cannot be utilized. In these non-GNSS scenarios, a neural network combined with a single camera can act as an effective alternative to GNSS. This is especially useful in situations where GNSS signals are unavailable, e. g., because the initial seeding was performed without GNSS-based equipment, resulting in missing positional data, or due to signal obstruction caused by environmental factors, like overhead agrivoltaics systems. This paper explores the process of generating synthetic images using a game engine and a diffusion model. These synthetic images will be utilized to train a neural network for tramline detection. The neural network, when trained with real images, achieved a mean Intersection over Union (mIoU) of 81.7 %. Incorporating synthetic images into the training process increased the mIoU to 83.3 %, resulting in improved tramline detection.

**Keywords**

Synthetic images, game engines, machine learning, autonomous driving

In the realm of precision farming, innovative guidance systems are essential for enhancing efficiency and accuracy. An alternative to the widely used Global Navigation Satellite System (GNSS) are non-GNSS guidance systems. This option is particularly useful in situations where there is a loss of GNSS (Vélez et al. 2024), when GNSS is not available, or when the tramline coordinates in a field are unknown. This occurs when the initial seeding was performed wit hout GNSS-based equipment, resulting in missing positional data, or when GNSS signals are obstructed by environmental factors such as overhead agrivoltaics systems. In such cases, a non-GNSS system can serve as a reliable alternative.

For the tramline guidance, the combination of a mono camera and a neural network (NN) is a very useful tool to achieve a non-GNSS guidance system. To train such a network effectively, a large dataset of diverse images is necessary. Thus, a major challenge is to ensure that enough images containing corner cases are included in the training data. Moreover, it is important to incorporate various environmental factors, such as plant vegetation or weather conditions, to represent all scenarios in the field. Depicting all use cases with real images, as well as the mandatory subsequent image annotation, would be labor-intensive, as it requires substantial manual effort and human involvement, and time-consuming, due to the extensive duration needed to collect, process, and annotate the data. Referring to the Cityscape dataset, annotating and verifying a single image with 30 classes takes 1.5 hours on average, highlighting the complexity of the task (Cordts et al. 2016). Hence, to achieve the required objectives in a cost-effective and timely manner, using synthetic images is a more feasi-

received 15 April 2025 | accepted 27 February 2026 | published 12 March 2026

ble option for two main reasons. Firstly, synthetic images are suitable to represent different scenarios and various environmental conditions with little effort. Secondly, generating synthetic images automatically creates corresponding annotation masks, crucial for training neural networks.

In the present study, synthetic images are created using the game engine Unity (Unity Technologies, USA, https://unity.com/de). The engine generates images covering various scenarios and environmental conditions to create a diverse dataset for training purposes. Generating highly realistic plant and soil types, weather conditions, and other elements in a game engine usually requires a significant amount of time. To enhance the fidelity of synthetic images in a more time-saving way, neural networks, such as diffusion models, are used, resulting in modified synthetic images.

Considering real, synthetic, and modified synthetic images, the aim of the presented study is twofold: The first goal is to create and analyze different datasets, each for training a neural network that can detect tramlines on a field for localization purposes. Specifically, the type of images is varied between the training datasets. Thus, either real, synthetic, modified synthetic, or a combination of two types of images are used to train a neural network. To evaluate the similarity between each training dataset, the Fréchet Inception Distance (FID) score (Heusel et al. 2017) is determined. The second goal is to analyze the performance of the neural networks. For this purpose, the mean Intersection over Union (mIoU) (Rezatofighi et al. 2019) is calculated for each one of the neural networks and compared between them. Within the scope of this study, the mIoU measures the ability of every network to detect real tramlines in a field. The FID and the mIoU will be explained in more detail in a later section. Based on the results, best practices for training a neural network for tramline detection in an autonomous driving tractor are derived.

## Overview of the state of the art

The generation of synthetic images has become increasingly important in various sectors in recent years. Especially in the automotive industry and in agriculture, artificial images are gaining in importance. In the automotive industry, synthetic images are mainly used to depict scenarios on the road in various ways to enable robust, automated, and safe driving. Agricultural applications of synthetic data are very diverse. However, they are used particularly frequently for the detection of plant diseases. In principle, synthetic data can be generated in two different ways. On the one hand, the data can be generated using game engines and on the other hand, using artificial intelligence (AI). Both approaches will be presented in more detail later. A short overview of the pros and cons of both alternatives is shown in Table 1.

Table 1: Comparison of the synthetic image generation with game engines vs. machine learning algorithms

| **Game engines** | **Artificial Intelligence** |
|---|---|
| + High control over details and interactivity | + Fast generation of large datasets |
| + Realistic physics and lighting models | + Automated customization and variation |
| + Established tools and a large community support | + Ability to create creative and unexpected designs |
| − High effort and time required for creation in contrast to AI | − Dependence on large datasets for training |
| − Requires specialized knowledge and skills | − Potential quality and consistency issues |
| − High costs for licenses and hardware | − Limited control over specific details |

Frequently used 3D modeling software for creating synthetic data includes Unity (Unity Technologies, USA, https://unity.com), Unreal Engine (Epic Games, Inc., USA, https://www.unrealengine.com), or Blender (Blender Foundation, Netherlands, https://www.blender.org). Unity was used, for example, to create the virtual KITTI dataset (Gaidon et al. 2016) and the Synthia dataset (Ros et al. 2016) for representing real automotive scenarios. Based on Unreal Engine, for instance, CARLA was developed, which is a simulator for urban driving environments and is used for training and testing algorithms for autonomous driving (Dosovitskiy et al. 2017). CARLA has been extended to also simulate off-road driving (Matt Rowe 2023). Another simulation developed using Unreal Engine is AURELION (dSPACE Group SE & Co. KG, Paderborn, https://www.dspace.com). AURELION provides high-quality visualization and realistic sensor data for testing and validating driving functions in both urban and off-road environments. However, games such as GTA V are also utilized to create artificial datasets (Richter et al. 2016).

In the agricultural context, for example, Cieslak et al. (2024) used Blender to simulate different growth stages of plants, random field arrangements, and different soil conditions under different lighting conditions. Jung et al. (2024) implemented the combination of Blender and Unreal Engine to generate synthetic data for the detection of pine wilt disease.

The most common artificial intelligence methods for generating synthetic models are Generative Adversarial Networks (GAN) algorithms or diffusion models. Huang et al. (2018) used Multimodal Unsupervised Image-to-image Translation (MUNIT), which is based on GAN algorithms, to generate multiple environmental conditions in road traffic based on a single image. Zhao et al. (2024) utilized both GAN algorithms and diffusion models to generate images of road traffic. The results have shown that the diffusion models can achieve higher mIoU scores.

Singh et al. (2024) and Zhang et al. (2024) implemented different GAN algorithms to artificially project a disease onto images of healthy leaves to enlarge the data set. Li et al. (2024) used GAN algorithms to generate images of weeds to improve the identification of real weeds.

A common problem in generating synthetic images is the domain gap between synthetic and real images. This means that synthetic images do not share the same characteristics as real images, for example, in terms of environmental appearance. To reduce the domain gap, two methods of synthetic image generation are combined in this paper.

## Theoretical background

In this study, the Unity game engine and the machine learning technique ControlNet (Zhang et al. 2023) were utilized to generate synthetic images. Unity is a widely used development platform for creating 2D and 3D games as well as interactive content. It supports multiple platforms, enabling developers to create applications for PCs, consoles, mobile devices, and even Virtual Reality (VR) and Augmented Reality (AR). In game engines like Unity, 3D objects are utilized to generate new environments. These 3D objects consist of multiple connected polygons, commonly referred to as a mesh. A polygon is a flat, two-dimensional shape consisting of multiple vertices, typically in the form of a triangle or a quadrilateral. The level of detail and smoothness of a 3D object increases with the number of polygons used per area. However, this also increases the computational load. To get fine details of an object, a texture is applied to the mesh. A texture is a bitmap image that is applied onto the surface of the 3D model. To apply textures to the 3D models, materials are utilized. Materials utilize specialized graphics programs to render a texture on the mesh surface. Those graphic programs are called

shaders. Shaders can create lighting and coloring effects to simulate various surface properties, such as shininess or bumpiness. Additionally, shaders can use multiple textures simultaneously, combining them to achieve greater flexibility and more complex visual effects. (Unity 2025b, Koulaxidis and Xinogalos 2022)

To generate images or frames from the 3D environment that can be displayed on a screen, rendering is essential. There are multiple types of rendering techniques, i.e., rasterization and raytracing. For this paper, only rasterization was used, because raytracing is a more time-consuming rendering method. Therefore, in the following, only rasterization will be explained.

At the rasterization for each vertex of a 3D model, which is visible by the camera, a pixel color is calculated using a model of shading (lighting). The whole rendering pipeline using rasterization is shown in Figure 1.

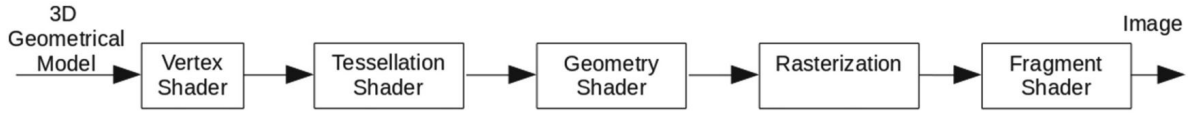


Figure 1: Rendering pipeline using rasterization

In the first step, the vertex shader receives the 3D model that is to be rendered and transforms it into a position relative to the camera. The next two steps, called tessellation and geometry shaders, are optional. To generate more polygons by vertex combinations, the tessellation shader can be used. The geometry shader, on the other hand, can manipulate the primitives (polygons, lines and vertices) in different ways, i.e., stretching or introducing holes, to create more complex models. During the rasterization process, polygons are converted into pixel fragments by projecting them onto the image using the camera model. This step also enables the interpolation of any output variable from the vertex shader, such as color and position. In the final step, the color of the pixels according to the lighting, textures, and materials is calculated by the fragment shader (Halmaoui and Haqiq 2022). In Unity, different render pipelines are available, such as the Universal Render Pipeline (URP), the High Definition Render Pipeline (HDRP) or the Built-In Render Pipeline (Unity 2025a).

The ControlNet is a neural network architecture that adds spatial conditioning controls to large, pretrained text-to-image diffusion models. These diffusion models normally generate images from text, but ControlNet allows guiding the generation using additional information, such as edges, depth maps, or segmentation, so the output matches specific spatial conditions. Instead of retraining the whole model, ControlNet reuses the strong feature extraction layers of existing models (trained on billions of images) and adds new control branches for these extra inputs. A key technique is zero convolution. This means the added layers start with all weights set to zero and gradually learn during training. Starting from zero prevents the new layers from introducing noise or damaging the original model.

Experiments show that ControlNet can be trained effectively with both small datasets (< 50,000 images) and large ones (> 1 million images), making it flexible for many applications. Figure 2 illustrates the comparison between a standard network and one augmented with ControlNet. Typically, a neural network block takes a feature map $x$ as input and produces another feature map $y$ as output (Figure 2a). When integrating ControlNet, the original block is frozen, and a trainable copy is created. Both are linked through zero-convolutional layers (Zhang et al. 2023).

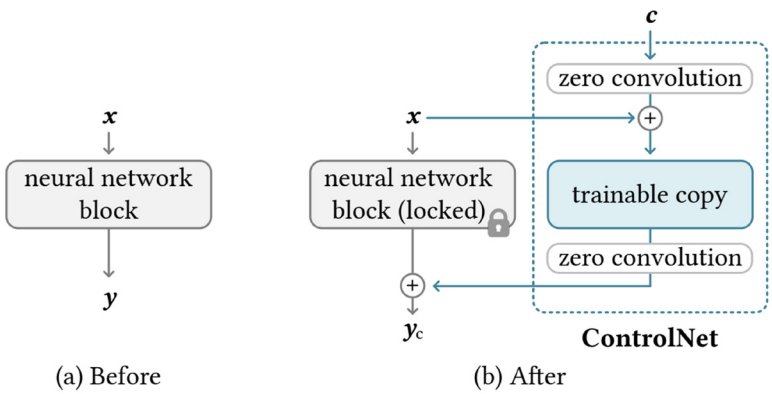


Figure 2: Network without (a) and with (b) ControlNet (Zhang et al. 2023)

For training neural networks, different parameters need to be set. These parameters include, among others, the batch size and the learning rate. In neural network training, multiple images are often used simultaneously as input. This collection of images is referred to as a batch. Therefore, the batch size indicates the number of images in that batch. A larger batch size can help achieve high algorithm accuracy in a shorter amount of time, but also increases computational effort (Smith et al. 2017). The learning rate, on the other hand, controls how much the parameters change in a single update (Fan et al. 2019).

After training a neural network, the learned knowledge must be saved. This saved state of the model, including its weights and optimizer settings, is called a checkpoint. The checkpoint enables resuming training and restoring the best-performing version without restarting from scratch (PyTorch 2025). While training a neural network, overfitting can occur when the model learns both the underlying patterns and the noise in the training data, leading to high accuracy on the training set but poor generalization to unseen data (Montesinos López et al. 2022).

The mIoU score is a metric used to evaluate the accuracy of semantic segmentation models. It measures how well the predicted segmentation masks align with the ground truth masks. The mIoU score is calculated by taking the average Intersection over Union (IoU) across all classes in the dataset. The IoU for a single class is defined as the area of overlap between the predicted and ground truth masks divided by the area of their union. The formula is shown in equation 1. $A$ represents the prediction and $B$ the ground truth (Zhang et al. 2022).

$$IoU = \frac{A \cap B}{A \cup B} \quad \text{(Eq. 1)}$$

To further interpret the IoU metric in the context of pixel-wise classification, it can also be expressed in terms of classification outcomes. For a given class, the IoU is defined as:

$$IoU = \frac{TP}{TP + FP + FN} \quad \text{(Eq. 2)}$$

True positives (TP) are pixels correctly predicted as belonging to the target class. False positives (FP) are pixels incorrectly predicted as belonging to the target class, although they belong to a different class. Lastly, false negatives (FN) are pixels that belong to the target class but were misclassified as another class.

The mIoU score is the mean of these IoU values for all classes. A higher mIoU score indicates better performance, with a maximum value of 1 representing perfect segmentation.

The Fréchet Inception Distance (FID) score is useful for capturing the similarity of generated images to real images. To compute the FID score, the generated images and real images are passed through a pretrained Inception v3 network (Szegedy et al. 2016) to extract feature representations. The mean and covariance of these features are then calculated for both sets of images. The FID score is derived from the Fréchet distance between these two multivariate Gaussian distributions, considering both the difference in means and the trace of the covariance matrices. A lower FID score indicates that the generated images are closer in quality and diversity to the real images. This makes the FID score a valuable tool for researchers and developers working on improving generative models, as it provides a quantitative measure of progress (Heusel et al. 2017).

## Method

At the beginning of the study, a dataset of 1,000 real-world images was collected. After removing the corrupted images, 978 images remained. These images were used to compare synthetic images against real-world images to evaluate the results, but also to train the ControlNet network.

To generate synthetic images, the game engine Unity was utilized to construct a 3D representation of an agricultural field. The base scene was designed to achieve a high degree of visual realism, incorporating typical elements found in agricultural settings. These include crops, trees, a tractor, soil textures, and tramlines across the field. The required components were sourced as virtual 3D models and subsequently imported into Unity for scene composition. An overview of the development environment in Unity is shown in Figure 3. In the development environment, the 3D environment with tramlines, plants, and trees is displayed in the center. The red dots symbolize the path along which the tractor is supposed to move. The path can be generated in two different ways. On the one hand, a “standard field” with typical AB lines can be created in the simulation. On the other hand, Bézier curves can be used to create randomized paths. On the right side of Figure 3, the settings options are displayed, allowing the user to choose between the two path types, select plant species, and adjust other parameters. In this environment, it is also possible to adjust the lightning conditions.

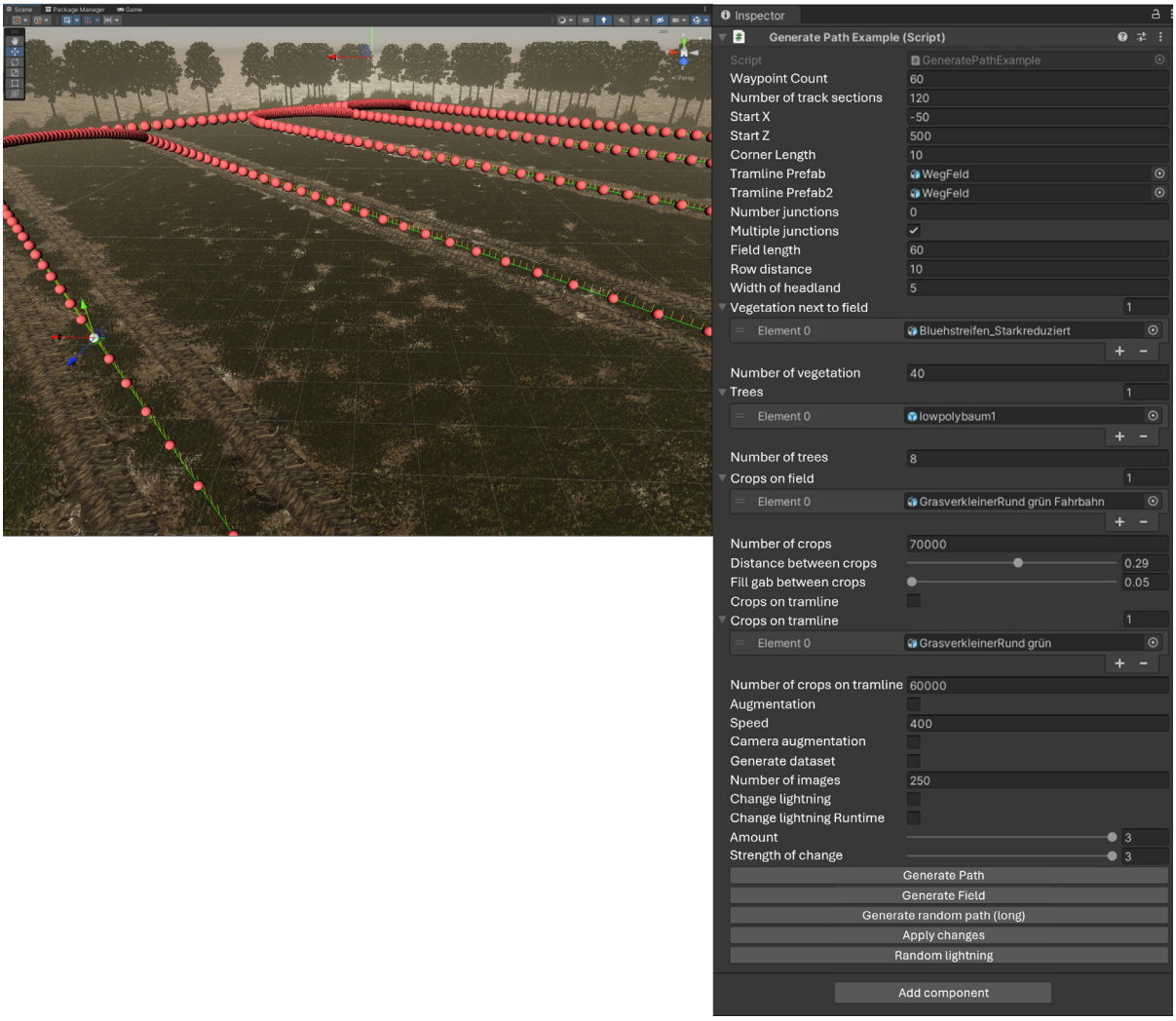


Figure 3: Development environment in Unity with the visual representation of the environment on the left side and the parameters to adjust the environment (e.g., change path or plant types) on the right side

Changing all those parameters, a dataset with a high variety was generated. Figure 4 to Figure 7 show different variations of the environment. Figures 4 and 5 show that the soil in the field can be altered, resulting in a different appearance of the tramline. The image in Figure 6 shows the effect when the sun is rising or setting. And lastly, it is also possible to put plants on the tramlines, shown in Figure 7, which is also a use case that has to be considered. During the extraction of the images, additional annotation masks are generated automatically for subsequent semantic segmentation. After creating the images using Unity and reviewing the dataset, redundant and corrupted images were removed. Since image generation in Unity is automated and requires less effort compared to collection real world images, a larger dataset was produced, resulting in a training dataset of 4,531 images. An example of an image with the corresponding annotation mask generated is shown in Figures 8 and 9. Figure 8 shows the Red-Green-Blue (RGB) image, while Figure 9 shows the annotation mask.

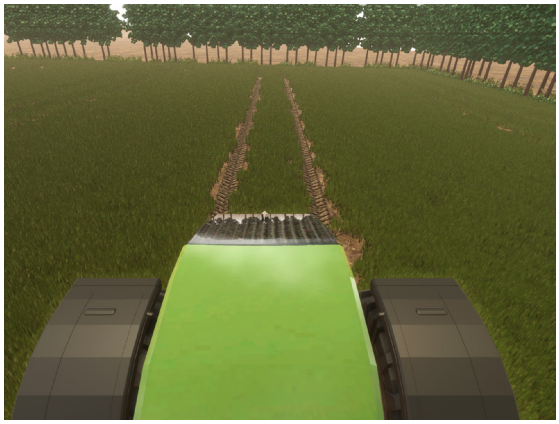

Figure 4: Synthetic image with tire tracks on dark soil

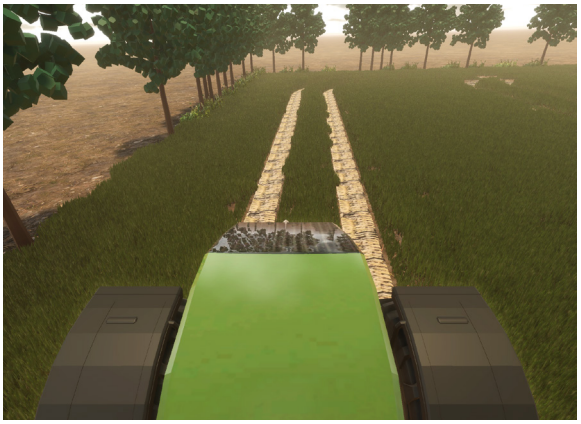

Figure 5: Synthetic image with tire tracks on dry soil

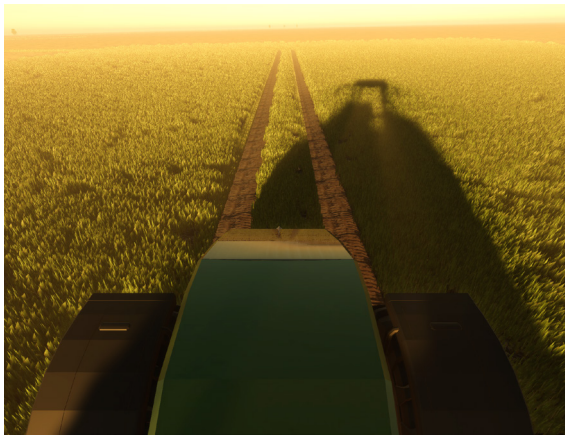

Figure 6: Synthetic image with sunset or sunrise

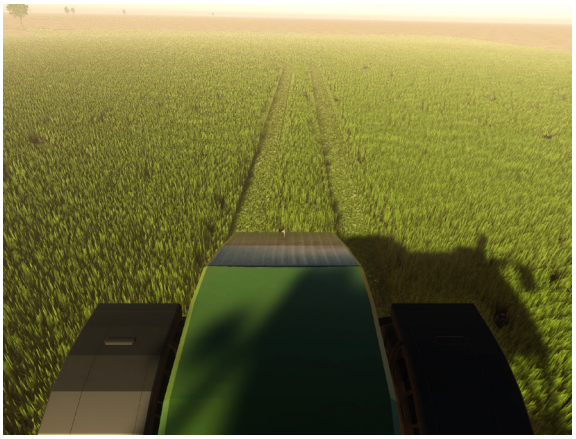

Figure 7: Synthetic image with plants on tramline

Figure 8: Synthetic RGB image generated with Unity

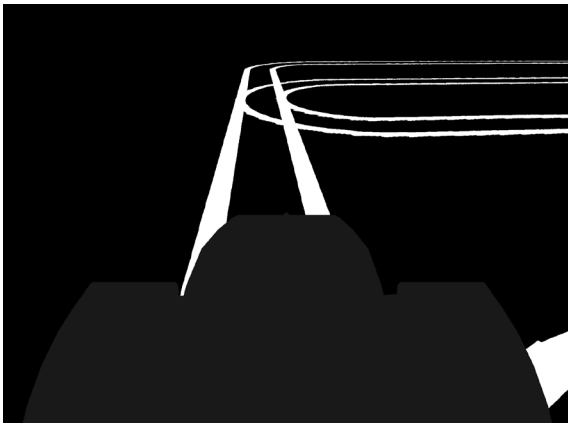

Figure 9: Synthetic annotation mask automatically generated with Unity

Table 2 provides an overview of the number of images for each condition. For every condition, the lightning was varied from bright daylight to sunset, as shown in Figure 4 and Figure 6.

Table 2: Variety of the Unity dataset; in batches 13 to 17, motion blur, depth of field, and exposure were additionally randomized, and the plants were consistently placed tightly along the tramlines, leaving no gaps

| Batch of the dataset | Crop | Tramline texture | Number of images |
|---|---|---|---|
| 1 | Wheat (BBCH: 21–34) | Loam soil | 810 |
| 2 | Wheat (BBCH: 21–34) | Dry sandy loam | 251 |
| 3 | Wheat (BBCH: 21–34) | Loam soil with wheat on the tramlines | 251 |
| 4 | Wheat (BBCH: 21–34) | Partially compacted sandy loam | 251 |
| 5 | Wheat (BBCH: 21–34) but with a higher crop–density than batch 4 | Partially compacted sandy loam | 251 |
| 6 | Dry Grass | Partially compacted sandy loam | 251 |
| 7 | Barley (BBCH: 21–34) | Partially compacted sandy loam | 251 |
| 8 | Barley (BBCH: 37–39) | Partially compacted sandy loam | 251 |
| 9 | Barley (BBCH: 72–75) | Partially compacted sandy loam | 251 |
| 10 | Barley (BBCH: 72–75) | Partially compacted sandy loam | 251 |
| 11 | Barley (BBCH: 55–83) | Loam Soil | 251 |
| 12 | Barley (BBCH: 55–83) | Dry sandy loam | 251 |
| 13 | Barley (BBCH: 55–83) | Loam soil | 160 |
| 14 | Wheat (BBCH: 21–34) | Dry sandy loam | 160 |
| 15 | Wheat (BBCH: 21–34) | Partially compacted sandy loam | 320 |
| 16 | Dry Grass | Loam Soil | 160 |
| 17 | Wheat (BBCH: 35–19) | Dry sandy loam | 160 |

In the next step, the domain gap between real and synthetic images was reduced using a diffusion model. For this purpose, the ControlNet was used. The workflow for the usage of the ControlNet is shown in Figure 10. In the first step, the ControlNet was trained using 978 real images, the corresponding annotation masks, and a text prompt. In this case, the same text prompt was used for each image. The text prompt was: “A tractor is driving through a field of grass with light brown tracks”. In the second step, the synthetic images are finally generated using the ControlNet. Therefore, the annotation masks generated in Unity are used. Additionally, a text prompt must be included, describing what should be visible on the output image. For simplicity, the text prompt was not changed and is the same used for training: “A tractor is driving through a field of grass with light brown tracks”. An example of an output is shown in Figure 10 on the right side. The process generated 4,531 images with the ControlNet.

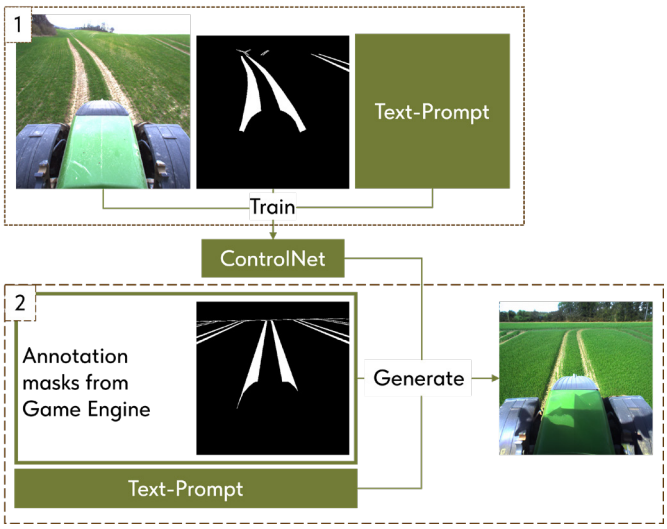


Figure 10: Pipeline to train a ControlNet and generate new images with the ControlNet based on annotation masks of the synthetic images

The parameters for training and using the ControlNet are shown in Table 3. Because of the high memory requirements for the training, a batch size of 1 was used. Furthermore, the standard learning rate of 0.00005 was used. During the training of the model, sample images were saved for each epoch, which were created with the respective checkpoints of each epoch. In total, the network was trained for 1,000 epochs. The base model used was StableDiffusion (SD) version 1.5. For the generation process, an unconditional guidance scale of 9 and a Denoising Diffusion Implicit Model (DDIM) of 50 were used.

Table 3: Parameters for the training of the ControlNet and generation of new images

| **Parameter** | **Wert** |
|---|---|
| Batch-Size | 1 |
| Learning Rate | 0.00005 |
| Epochs | 365 |
| SD locked | True |
| Only mid control | False |
| SD version | 1.5 |
| Unconditional guidance scale | 9.0 |
| DDIM steps | 50 |

Afterward, all datasets were used to train a neural network for the semantic segmentation of real tramlines. The segmentation network, which was created with the output images of ControlNet, was also fine-tuned with the real images after training. This process involved taking the best saved checkpoint obtained during the training on the ControlNet dataset and then continuing to train the network with real images. In addition, the 978 real images used to train ControlNet were also utilized to train a model for semantic segmentation, which serves as a reference. For validation, 300 real images were used, and the metric mIoU was calculated. Furthermore, the FID score was determined to measure the similarity of the synthetic images to the real images.

## Results

In this section, the test results are presented and validated. During the training of the ControlNet model, the loss function is the only metric reported. As shown in Figure 11, the loss decreases up to epoch 1000, indicating steady optimization progress.

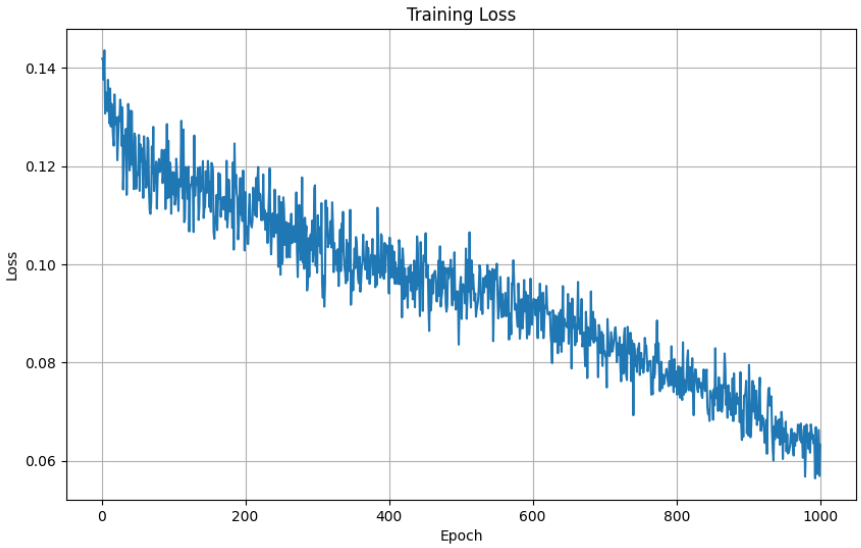


Figure 11: Loss over epoch while training the ControlNet

However, the loss function alone does not provide a meaningful measure of perceptual image quality or semantic fidelity. This limitation becomes evident in Figure 12, which presents an example output image generated at epoch 1000. Despite the low loss value, the resulting image displays noticeably unrealistic color distributions, illustrating that convergence in the loss space does not necessarily correspond to improved visual realism.

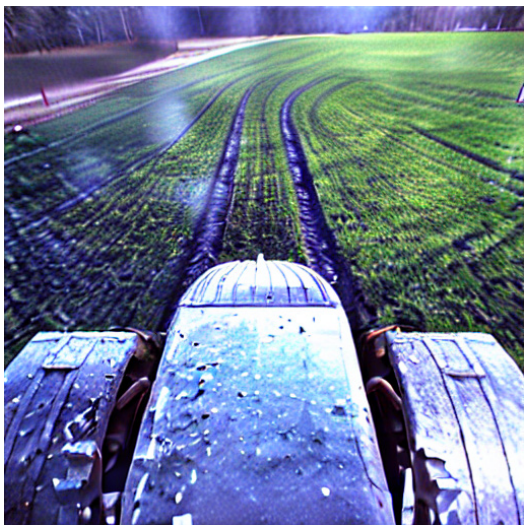

Figure 12: Output of the ControlNet after 1000 Epochs

To qualitatively assess the progress of model performance throughout training, a representative image was generated after each epoch. This procedure enabled a direct visual comparison across checkpoints. Among the inspected epochs, epoch 365 produced outputs that appeared more realistic than those generated at later stages. Consequently, the checkpoint corresponding to epoch 365 was selected for subsequent experiments. In future work, it would be desirable to generate a dedicated evaluation dataset at every epoch checkpoint and to subject these datasets to a systematic quantitative analysis. Metrics such as the Fréchet Inception Distance (FID) could be employed to objectively assess image realism and distributional similarity across the entire training trajectory. This would allow a more reproducible comparison between checkpoints and reduce the subjective bias introduced by visual inspection alone. However, such a comprehensive quantitative evaluation was beyond the scope of the present study and was therefore not conducted here.

Figure 13 displays the mIoU scores achieved by the segmentation networks. Based on the real data, an mIoU of 81.7 % was achieved.

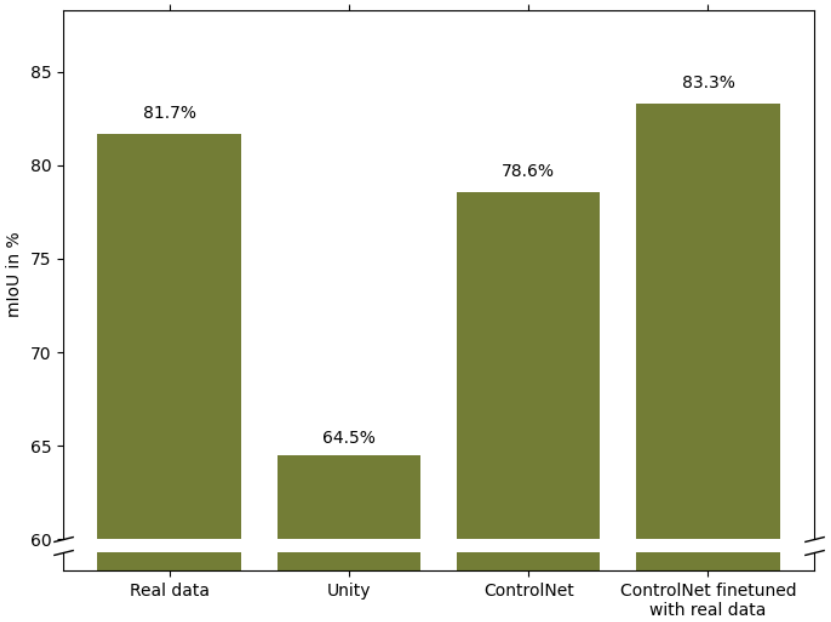


Figure 13: mIoU – mean Intersection over Union scores in % of the semantic segmentation networks based on the real data (81,7 % ), the Unity data (64.5 % ), the ControlNet output (78.6 % ), and the ControlNet output and afterwards finetuned with real data (83,3 % )

Using the unmodified Unity dataset resulted in an mIoU of 64.5 %. It can be concluded from the results that the recognition of real tramlines based on synthetic data does not work as well as based on real data yet. In Figure 13, it is also shown that an mIoU of 78.6 % was achieved through the ControlNet output. This means that the mIoU is increased by 14.1 % when incorporating the ControlNet compared to the simple output from Unity. As mentioned before, the segmentation network based on the ControlNet images was finally fine-tuned with the real images. Thereby, an optimized mIoU of 83.3 % was achieved. This means that the mIoU was increased by 1.6 % when using synthetic data. It should be mentioned that all results are validated solely on the same set of 300 validation images. Changing these images or varying the number of validation images could yield different outcomes. The results derived from the ControlNet images demonstrate that synthetic images can yield good

outcomes on their own, with only a 3.1 % difference. The current goal is to achieve a mean Intersection over Union (mIoU) with solely synthetic images that matches the performance of real images. Furthermore, the outcomes from the fine-tuned neural network demonstrate an even higher detection accuracy, indicating that synthetic images have great potential. Normally, a crossvalidation procedure would be applied to assess the robustness and statistical significance of model performance. However, in this study, it is not feasible because cross-validation would require mixing synthetic and real data across training and validation folds. This would undermine the primary goal of our study, which is to evaluate models solely on real data to ensure comparability between networks trained on real data and those trained on synthetic data.

To visually represent the results, an example image was annotated using the trained segmentation networks visible in Figure 14 to Figure 17.

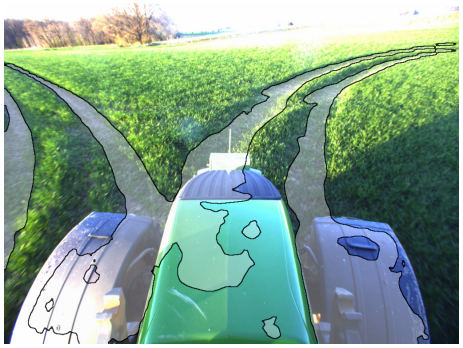

Figure 14: Annotated image based on the real images

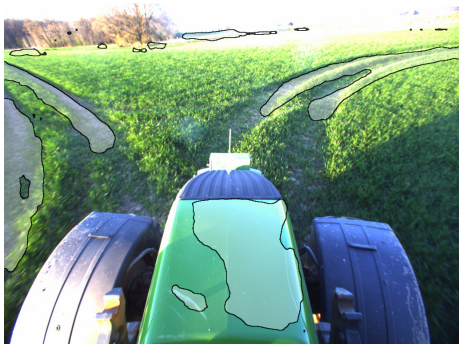

Figure 15: Annotated image based on the synthetic images (Unity)

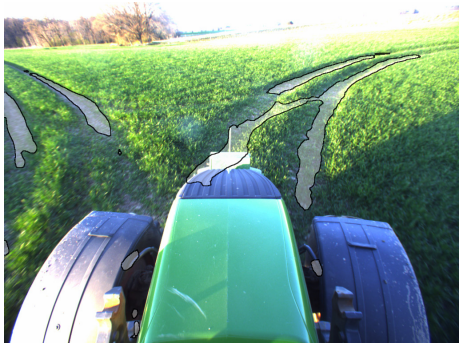

Figure 16: Annotated image based on the modified synthetic images (ControlNet)

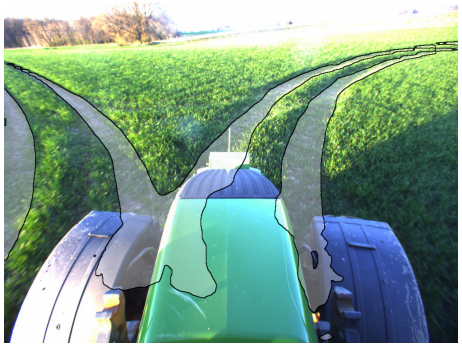

Figure 17: Annotated image based on the modified synthetic images (ControlNet) finetuned with real images

In Figure 14, the semantic segmentation network, which is based on real data, effectively identifies the tramlines. However, a significant portion of the tractor is also being identified as a tramline. This explains why the mean Intersection over Union (mIoU) is only 81.7 %, rather than a higher value. Figure 15 shows the output based on the NN with synthetic data. Near the tractor, it is particularly noticeable that the tramlines are not well recognized. Additionally, the tractor is partially misclassified as a tramline. Those misclassifications explain the lower mIoU based on synthetic images. Figure 16 visualizes the output of the NN based on ControlNet images. It is visible that tramlines are better recognized compared to Figure 15. Furthermore, the false positive (FP) values decrease as the tractor is less frequently misclassified as a tramline. Finally, Figure 17 displays the output of the finetuned

ControlNet network. There are fewer misclassifications of the tractor compared to Figure 14. Additionally, the tramlines are fully recognized, resulting in a high mIoU. In the post-processing used for calculating the trajectories, a mask will be laid over the region of the tractor in the image. Therefore, the misclassification of the tractor doesn't need to be considered. Only if the mask of a tractor is not available or the position of the camera is changing, fewer misclassifications are of importance. To get a further understanding of why the segmentation results are as different as they are, the FID scores are presented in Table 4. The score between two different real datasets is 29.3, while the FID score between the synthetic dataset and the real dataset is 170.5. This is due to the domain gap between the real and synthetic data, which is the result of, for example, the different plant and tramline textures. Comparing the ControlNet output with real images results in an FID score of 47.4. Thus, the FID decreased by 123.1, showing that the ControlNet effectively reduces the domain gap.

Table 4: List of the FID (Fréchet Inception Distance) scores for the different datasets compared to a real dataset

| Dataset comparison | Fréchet Inception Distance |
|---|---|
| Real to Real | 29.3 |
| Unity to Real | 170.5 |
| ControlNet to Real | 47.4 |

## Conclusions

The experiments conducted in this study have demonstrated the effectiveness of using synthetic images for tramline detection in precision farming. By employing a combination of real, synthetic, and modified synthetic images, the neural network achieved a mean Intersection over Union (mIoU) of 83.3 % after fine-tuning with real data. This result surpasses the mIoU of 81.7 % obtained using only real images, highlighting the potential of synthetic data to enhance neural network performance.

The use of synthetic images offers significant advantages, particularly in terms of cost and time efficiency. Generating synthetic images allows the creation of diverse datasets that include various environmental conditions and corner cases, which would otherwise be labor-intensive and time-consuming to collect and annotate manually. This approach ensures that the neural network is trained on a comprehensive dataset, improving its robustness and accuracy in real-world applications.

Moreover, the integration of ControlNet to reduce the domain gap between synthetic and real images has proven to be effective. The Fréchet Inception Distance (FID) score of 47.35 indicates a substantial improvement in the similarity between synthetic and real images, further validating the use of synthetic data in training neural networks.

The experiments demonstrate that synthetic data is effective for tramline detection. However, this study focuses solely on detection accuracy (mIoU) and image domain adaptation. Other important metrics, such as cross-track error, overall guidance accuracy, and reliability compared to GNSS-based systems, have not yet been assessed. Evaluating these factors will require dedicated field tests in real operating conditions. Consequently, the current findings should be viewed as an initial step towards developing a non-GNSS guidance system. Future work will aim to determine whether the proposed approach meets industry standards for accuracy and reliability.

In conclusion, this study has made a significant contribution to the field of precision farming by demonstrating the viability of using synthetic images for tramline detection. The findings suggest that synthetic data can not only complement but also enhance the performance of neural networks trained on real data. Future research should focus on further improving the accuracy of tramline detection by increasing the variation in synthetic images and exploring advanced techniques for domain adaptation. Additionally, using image-specific text prompts when training ControlNet could further reduce the domain gap and improve detection accuracy. By continuing to refine these methods, the agricultural industry can benefit from more reliable and efficient non-GNSS guidance systems, ultimately contributing to the advancement of precision farming technologies.

## References

Cieslak, M.; Govindarajan, U.; Garcia, A.; Chandrashekar, A.; Hädrich, T.; Mendoza-Drosik, A.; Michels, D.L.; Pirk, S.; Fu, C.-C.; Palubicki, W. (2024): Generating Diverse Agricultural Data for Vision-Based Farming Applications. 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pp. 5422–5431, https://doi.org/10.48550/arXiv.2403.18351

Cordts, M.; Omran, M.; Ramos, S.; Rehfeld, T.; Enzweiler, M.; Benenson, R.; Franke, U.; Roth, S.; Schiele, B. (2016): The Cityscapes Dataset for Semantic Urban Scene Understanding. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, pp. 3213-3223, https://doi.org/10.1109/CVPR.2016.350

Dosovitskiy, A.; Ros, G.; Codevilla, F.; Lopez, A.; Koltun, V. (2017): CARLA: An Open Urban Driving Simulator. PMLR Proceedings of Machine Learning Research, pp. 1–16, https://doi.org/10.48550/arXiv.1711.03938

Fan, J.; Ma, C.; Zhong, Y. (2021): A Selective Overview of Deep Learning. Statistical science 36(2), pp. 264–290, https://doi.org/10.1214/20-STS783

Gaidon, A.; Wang, Q.; Cabon, Y.; Vig, E. (2016): VirtualWorlds as Proxy for Multi-object Tracking Analysis. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, pp. 4340–4349, https://doi.org/10.1109/CVPR.2016.470

Halmaoui, H.; Haqiq, A. (2022): Computer Graphics Rendering Survey: From Rasterization and Ray Tracing to Deep Learning. In: Innovations in Bio-Inspired Computing and Applications, eds. Abraham, A., et al. , IBICA 2021, Lecture Notes in Networks and Systems, vol 419, Springer, Cham, https://doi.org/10.1007/978-3-030-96299-9_51

Heusel, M.; Ramsauer, H.; Unterthiner, T.; Nessler, B.; Hochreiter, S. (2017): GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA, https://doi.org/10.48550/arXiv.1706.08500

Huang, X.; Liu, M.-Y.; Belongie, S.; Kautz, J. (2018): Multimodal Unsupervised Image-to-Image Translation. In: Computer Vision – ECCV 2018, eds. Ferrari, V.; Hebert, M.; Sminchisescu, C.; Weiss, Y., Lecture Notes in Computer Science, vol 11207. Springer, Cham, https://doi.org/10.1007/978-3-030-01219-9_11

Jung, Y.; Byun, S.; Kim, B.; Amin, S.U.; Seo, S. (2024): Harnessing synthetic data for enhanced detection of Pine Wilt Disease: An image classification approach. Computers and Electronics in Agriculture 218, p. 108690, https://doi.org/10.1016/j.compag.2024.108690

Koulaxidis, G.; Xinogalos, S. (2022): Improving Mobile Game Performance with Basic Optimization Techniques in Unity. Modelling 3(2), pp. 201–223, https://doi.org/10.3390/modelling3020014

Li, T.; Asai, M.; Kato, Y.; Fukano, Y.; Guo, W. (2024): Channel Attention GAN-Based Synthetic Weed Generation for Precise Weed Identification. Plant Phenomics 6, https://doi.org/10.34133/plantphenomics.0122

Matt Rowe (2023): Lets' go off-road. https://carla.org/2023/04/21/avl-off-road-simulation/, accessed on 7 Oct 2025

Montesinos López, O.A.; Montesinos López, A.; Crossa, J. (2022): Overfitting, Model Tuning, and Evaluation of Prediction Performance. In: Multivariate Statistical Machine Learning Methods for Genomic Prediction, Cham, Springer International Publishing, pp. 109–139, https://doi.org/10.1007/978-3-030-89010-0_4

PyTorch (2025): Saving and Loading Models. https://docs.pytorch.org/tutorials/beginner/saving_loading_models.html, accessed on 27 Feb 2026

Rezatofighi, H.; Tsoi, N.; Gwak, J.; Sadeghian, A.; Reid, I.; Savarese, S. (2019): Generalized Intersection Over Union: A Metric and a Loss for Bounding Box Regression. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, pp. 658–666, https://doi.org/10.1109/CVPR.2019.00075

Richter, S.R.; Vineet, V.; Roth, S.; Koltun, V. (2016): Playing for Data: Ground Truth from Computer Games In: Computer Vision – ECCV 2016, eds. Leibe, B., Matas, J., Sebe, N., Welling, M., Lecture Notes in Computer Science, vol 9906, Springer, Cham, https://doi.org/10.1007/978-3-319-46475-6_7

Ros, G.; Sellart, L.; Materzynska, J.; Vazquez, D.; Lopez, A.M. (2016): The SYNTHIA Dataset: A Large Collection of Synthetic Images for Semantic Segmentation of Urban Scenes, pp. 3234–3243, https://doi.org/10.1109/CVPR.2016.352

Singh, A.K.; Rao, A.; Chattopadhyay, P.; Maurya, R.; Singh, L. (2024): Effective plant disease diagnosis using Vision Transformer trained with leafy-generative adversarial network-generated images. Expert Syst. Appl. 254, p. 124387, https://doi.org/10.1016/j.eswa.2024.124387

Smith, S.; Kindermans, P.-J.; Le, Q. (2018): Don't Decay the Learning Rate, Increase the Batch Size. 6th International Conference on Learning Representations, Vancouver, BC, Canada, https://doi.org/10.48550/arXiv.1711.00489

Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; Wojna, Z. (2016): Rethinking the Inception Architecture for Computer Vision. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, pp. 2818–2826, https://doi.org/10.1109/CVPR.2016.308

Unity (2025a): Introduction to render pipelines. https://docs.unity3d.com/6000.0/Documentation/Manual/render-pipelines-overview.html, accessed on 15 Apr 2025

Unity (2025b): Introduction to textures. https://docs.unity3d.com/6000.0/Documentation/Manual/Textures.html, accessed on 15 Apr 2025

Vélez, S.; Valente, J.; Bretzel, T.; Trommsdorff, M. (2024): Assessing the impact of overhead agrivoltaic systems on GNSS signal performance for precision agriculture. Smart Agricultural Technology 9, p. 100664, https://doi.org/10.1016/j.atech.2024.100664

Zhang, L.; Rao, A.; Agrawala, M. (2023): Adding Conditional Control to Text-to-Image Diffusion Models. 2023 IEEE/CVF International Conference on Computer Vision (ICCV), pp. 3813–3824, https://doi.org/10.48550/arXiv.2302.05543

Zhang, Y.; Shi, N.; Zhang, H.; Zhang, J.; Fan, X.; Suo, X. (2022): Appearance quality classification method of Huangguan pear under complex background based on instance segmentation and semantic segmentation. Frontiers in Plant Science 13, https://doi.org/10.3389/fpls.2022.914829

Zhang, Z.; Zhan, W.; Sun, Y.; Peng, J.; Zhang, Y.; Guo, Y.; Sun, K.; Gui, L. (2024): Mask-guided dual-perception generative adversarial network for synthesizing complex maize diseased leaves to augment datasets. Engineering Applications of Artificial Intelligence 136, p. 108875, https://doi.org/10.1016/j.engappai.2024.108875

Zhao, H.; Wang, Y.; Bashford-Rogers, T.; Donzella, V.; Debattista, K. (2024): Exploring Generative AI for Sim2Real in Driving Data Synthesis. 2024 IEEE Intelligent Vehicles Symposium (IV), Jeju Island, Republic of Korea, pp. 3071–3077, https://doi.org/10.1109/IV55156.2024.10588493

## Authors

**M. Sc. Silko Schulpius** is a project engineer and PhD student at the AGCO GmbH, Gebrüder-Welger-Str. 3, 38304 Wolfenbüttel. E-Mail: silko.schulpius@agcocorp.com

**Dr.-Ing. Jan Schattenberg** is the head of automation and robot systems, and **Prof. Dr. Ludger Frerichs** is the Director of the Institute of Mobile Machines and Commercial Vehicles (IMN) at the Technische Universität Braunschweig, Langer Kamp 19a, 38106 Braunschweig.

## Acknowledgements

This research project is funded by the German Federal Ministry for Economic Affairs and Climate Action based on a resolution of the German Bundestag.