Adapting Agricultural Virtual Environments in Game Engines to Improve HPC Accessibility

Dirk Norbert Baker[1,2(✉)], Felix Bauer[3], Andrea Schnepf[3], Hanno Scharr[4], Morris Riedel[1,2], Jens Henrik Göbbert[2], and Ebba Hvannberg[1]

[1] School of Engineering and Natural Sciences, University of Iceland, Reykjavik, Iceland

[2] Jülich Supercomputing Centre, Forschungszentrum Jülich GmbH, Jülich, Germany
d.baker@fz-juelich.de

[3] Institute of Bio- and Geosciences 3: Agrosphere, Forschungszentrum Jülich GmbH, Jülich, Germany

[4] Institute for Advanced Simulation 8: Data Analytics and Machine Learning, Forschungszentrum Jülich GmbH, Jülich, Germany

Abstract. E-infrastructures deliver basic supercomputing and storage capabilities but can benefit from innovative higher-level services that enable use-cases in critical domains, such as environmental and agricultural science. This work describes methods to distribute virtual scenes to the GPU nodes of a modular supercomputer for data generation. High information density virtual scenes, containing $>$ 100k geometries, typically cannot be rendered in real-time without techniques that change the information content, such as level-of-detail or culling approaches. Our work enables the concurrent and partitioned coupling to the image analysis in such a way that the data generation is dynamic and can be allocated to GPU nodes on demand, resulting in the possibility of moving through a continuous virtual scene rendered on multiple nodes. Within agricultural data analysis, the approach is especially impactful as virtual fields contain many individual geometries that coexist in one continuous system. Our work facilitates the generation of high-quality image data sets, which has the potential to solve the challenge of scarcity of well-annotated data in agricultural science. We use real-time communication standards to couple the data production with the image analysis training. We demonstrate how the use-case rendering impacts effective use of the compute nodes and furthermore develop techniques to distribute the workload to improve the data production.

Keywords: Visualization · Computing Services · FSPM · Computer Vision

A. Azab and T. Malkiewicz (Eds.): NeIC 2024, CCIS 2398, pp. 152–167, 2025.
https://doi.org/10.1007/978-3-031-86240-3_11

1 Introduction

Analyzing camera images to extract agricultural plant information is a major bottleneck and one of the most challenging tasks in plant science [1–3]. Synthetic data generation is a strong contender to combat the scarcity of high-quality annotated data [4,5], especially with the tools available through modern graphics engines [6,7]. A scalable plant image generation pipeline could improve critical tasks such as training of statistical or Deep Learning (DL) models [8,9]. However, for virtual agricultural data to be useful, it must span realistic scales and contain functional information, such as root water uptake or photosynthesis. In this work we improve the scalability of rendering virtual environments in graphics engines leveraging current e-infrastructure capabilities, thus enabling a data-scarce domain to make more effective use of High-Performance Computing (HPC) resources.

Synthetic data generation is a method to model the input and output data in conjunction to allow for larger-scale training data sampling. A virtual environment involves the time-variant rendering of a scene, including objects, lighting, and movement. We refer to the *simulation* of a virtual environment as the computation of processes such as water flow or photosynthesis, which depend on plant structure. Synthetic data production for plant image analysis can be based on plant simulation models, though it might not depend on it, or not use a simulated plant model at all. Synthetic data provides a rigorous assessment of error margins as well as a scalable pipeline [6], as the employment of virtual ground-truth provides a definite error value that has no hidden effects, such as human labeling error.

Synthetic data can be used to pre-train DL-models to increase robustness towards new data. Scalability and robustness are valuable for decision making in agriculture, where multi-step algorithms impact assessments and actions [2,10]. One advantage of synthetic data is that it can be used to provide a baseline for evaluation [11], while it also provides high-quality annotated data [7]. To accommodate the need for state-of-the-art rendering pipelines, we are using the Unreal Engine (UE) as rendering framework, coupling it with plant simulation models to create virtual environments. To enable the coupling and concurrent computation of simulation model, virtual world, as well as DL framework, we are employing Synavis [12]. We developed Synavis specifically for the coupling of virtual environments with DL and simulation models.

A key aspect of virtual field simulation is the use of Functional-Structural Plant Models (FSPMs), such as CPlantBox [13,14]. FSPMs are statistical descriptions of plant traits (phenotypes) [13] and are calibrated using measurements and statistical optimization [14]. Because FSPM outputs are diverse [5,12,15], the generated scene configuration provides more diverse image data. CPlantBox in particular is useful since it was stochastically evaluated [16] and has proven applicability in replicating field experiments [17]. The functional simulation of plant systems is complex and requires coupling between all connecting systems to form a fully capable simulation model [14].

Modern agricultural fields contain tens of thousands of plants, optimizing for high density and yield. A digitized version of realistic crop fields requires resource management as well as an adaptive pipeline to suit different use-cases that have conflicting requirements. To allow for increased scalability, we have developed techniques that facilitate the distribution of large-scale generation of virtual scenes of digital crop fields on to multiple GPU nodes of an HPC system. Our technique requires little user-based alteration to the virtual environment created in UE. The need for both scalable virtual field simulation as well as cohesive data generation is met through the distribution of FSPMs across graphics engine instances. We enable comprehensive workflows to generate synthetic data on HPC systems. This paper provides relevant insight into previous work in all supporting domains and techniques in Sect. 2, before describing specific HPC scenarios to support plant science domains in Sect. 3. We provide a description of our experiments in Sect. 3.4 before we describe the measurement results in Sect. 4 and discuss them in Sect. 5.

2 Related Work

In context of our use-cases, there are key aspects that are embedded in partly disjunct domains that need to work together to form a fully formed pipeline. This section highlights important work in all parts of the pipeline, starting with synthetic data generation in UE. A well-known framework for a data generation framework with UE was developed by Qiu et al. [7], called UnrealCV, which uses filters and image-based commands to provide the ground-truth for these filters, such as object contours or scene depth. It uses a direct Python binding of UE to allow for the expedient generation of data sets. Using UnrealCV, both Zhang et al. [18] and McCormac et al. [19] produced effective pipelines for RGB camera depth estimation. Especially Zhang et al. highlight the interplay between algorithm performance and UE scene generation, showcasing that the data generation needs to be dynamically adapted for validation purposes, which would need to have special accommodation from the e-infrastructure provider. There are a number of DL applications[1] that have a baseline evaluation or also data set generation through UE, particularly surrounding agent-environment interaction, like trajectory optimization by Roberts et al. [20]. The visualization of virtual scenes to train agents has seen an overall increase in use, especially in industry, as described by Nassif et al. [21]. Particularly, Bondi et al. [22] developed a separate UE-based approach to train unmanned aerial vehicles in a controlled setting.

From general synthetic data generation, we are now highlighting work that focuses on generating plant synthetic data. Certain algorithms, such as leaf segmentation, can be trained using data that is generated through image augmentation, as shown by Ward et al. [4], who generate top-view leaf data with semantic segmentation. However, there are classes of problems where image-based synthetic data is not sufficient. One challenge in plant science is the measurement of small-scale features. The estimation of poses of animals is object-centered

[1] Publications based on UnrealCV can be found here.

detection of small-scale features, and Mu et al. [23] developed an approach to use UE to generate the appropriate training data. The recent adaption of pose estimation to plant science by Berrigan et al. [24] shows that the transposition of certain methods to plant science depend on large-effort acquisitions of data sets. To combat this, synthetic data using plant models can be used, but these models need biological validation. Thus, Morandage et al. [17] show a use-case for simulation model evaluation - by providing a synthetic field example and analyzing how well, from a parameter estimation view, the FSPM CPlantBox can describe the field data and how accurate the estimations are. Lobet et al. [15] created a generation pipeline for virtual root system images, showing another use of simulation model-based synthetic data generation. A generalized modeling framework such as Helios [5] can encompass virtual fields that include plants, as well as their surface structure and reaction to light influx. While CPlantBox itself is a stochastic description of the plant structure, as introduced by Zhou et al. [13], more recent advances in the coupling of FSPMs for functional processes developed by Giraud et al. [14] illustrate the descriptive power of these systems.

The generation of synthetic data, and the above mentioned methods, do not make efficient use of cutting-edge HPC resources of e-infrastructures, even though they overlap domains that individually have seen innovation regarding scalability. One particular approach is data-parallelism, which is the partitioning of a data set across nodes. A visualization service-based example of data parallelism has been developed by Aunmüller et al. [25] in their framework *Vistle*. Data parallel approaches require rethinking some approaches to rendering, but will yield a better efficiency of using Graphics Processing Unit (GPU) nodes. Data parallelism is expedient for some visualization techniques, such as isosurface visualization, but more difficult to adapt to others. For example, Larsen et al. [26] showcase a raytracing approach that is compatible with the paradigm of data parallelism and have adapted this image rendering approach, which commonly requires the whole data set, to function in parallel. Moreland et al. showcase their design concepts for highly-threaded data-parallel visualization approaches for the Visualization Toolkit (VTK) [27].

From an e-infrastructur point-of-view, our challenge is accommodating the use of synthetic data generation methods using UE, which have been proven to increase robustness, on HPC systems to allow these techniques to scale with the increasing computational demand of DL frameworks.

3 Methods

Our aim is to improve the accessibility of HPC systems, providing plant scientists with methods that enhance their data augmentation. To this end, here we showcase two experiments (E1 & E2) focusing on the GPU performance when generating increasing number of plants (E1) and the ability to optimize field partitioning for visualization of large fields with HPC resources (E2).

3.1 Software and Data Generation

For our analysis of large-scale field generation, we are using an application that was built using UE on a user system. To couple UE with other data providers as well as with the training framework, we are using the Synavis framework to ease the setup of connecting the individual services. In Synavis, we can connect the FSPM CPlantBox, which outputs geometries of its plant simulation, to the virtual scene for rendering. Information that is generated by the simulation can be used as reference or label data, allowing the training of a variety of scenarios, especially the validation of DL-models that estimate plant traits. This data is being rendered within UE on-demand and the coupling is not synchronous, meaning that most changes to the virtual environment are just-in-time, which also applies to the image generation. The coupling used in our approach is based on real-time communication and no data is being written to disc. Most of the setup is done in user Python scripts, as the virtual scenes can be fully dynamic. However, it might be preferable to introduce pre-designed environments of publicly available project files, which is possible by including the Synavis plugin in an already existing UE application.

We are using a virtual field setup that uses stochastically parameterized plants [28]. These plants are discrete node-link descriptions that each have transition probabilities assigned to their structure. The leaf calibration as well as the geometrization methods for CPlantBox have been described by Helmrich et al. [12]. Plant surface geometry is inferred from centerline splines evaluated in fixed resolutions. Image rendering is largely dependent on what is feasible to render in one instance of UE. Essentially, the more plants can be rendered per instance, the larger field of view can be rendered, resulting in reduced need for stitching for a wider view. Simplification of the individual geometries is done by scaling the geometry resolution. Due to simulation coupling employed in our pipeline, there is a mix of base geometries being rendered in the scene along with geometries that are being treated with dynamic hierarchical culling (such as by UE-Nanite [29]). The visualization module for CPlantBox generates geometry buffers that can almost immediately be written into GPU memory. UE uses procedural meshes for this task, which have a very simple base layer of transformation and collision support and mesh sections for geometry buffers. In some cases, geometry buffers are filled separately using individual organs to allow for instance-based segmentation. This is especially important for leaf counting tasks, which are indicators for leaf development [4] and thus plant growth stage.

Measurements from UE use virtual textures that act as rendering targets for scene capture cameras. Direct scene rendering, processing image information such as object distance or velocity (i.e. pixel movement relative to the camera), or object property quantification, use this proxy to allow for immediate dynamic measurements. We use Synavis to handle measurement prompts, such that the controller is able to extract arbitrary measurements. This is especially important in cases where there is a mix of different data sources, such as image data from the renderer as well as plant data from the FSPM. HPC infrastructure allows the scalability of the DL-model training, but the other components of the workflow

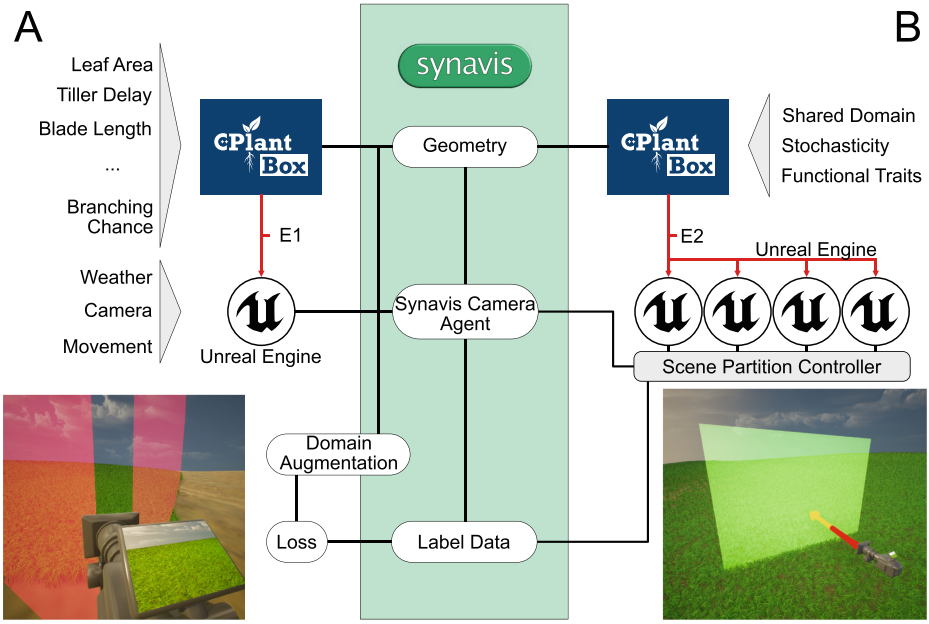

Fig. 1. Setup of pipelines in different distribution techniques. Measurements are highlighted and are also implemented in Synavis. The listed parameters are the primary steering parameters for the individual scenario. A. Continuous update with the intended use-case of adaptive data generation. B. Scene partition using Synavis and UE.

also need to be scalable to be on-par with the DL-model data requirements. To render the virtual scenes, UE requires the use of a Vulkan-compatible GPU, as is present in visualization or data analysis modules. Using Synavis, DL methods gain access to the domain-specific augmentation that is otherwise not accessible, which directly improves the pipelines that are also increasingly domain specific, with the potential for more impact.

3.2 HPC Scenario: Image Generation for Computer Vision

Computer Vision (CV) algorithms need labeled image data but often do not need to interact with the virtual environment. As such, compute infrastructure services for these systems should focus on enabling a responsive data generation to optimize DL training results. In our service implementation, Synavis keeps sending new FSPM realizations to UE, and parameter adaption directly influences the information content of rendered images. This is illustrated in Fig. 1.A, showing sample parameters that can be adapted in CPlantBox as well as in UE.

Ceaselessly updating the scene during image capture is a process that only functions if the relative speed of the camera agent compared to the simulation time is sufficiently slow. This limitation is largely dependent on how many plants need to be rendered, and how fast the FSPM computes a time step. In this

setup, the DL framework typically has full authoritative control over the field generation, and all data as well as images being generated depend on initial parameters or direct steering. This method depends on the live coupling of the FSPM with the environment. Evaluation of the FSPM is done on-demand, but as the virtual world is fully dynamic, we input a ceaseless stream of plant geometries to place in the scene, relative to the camera agent.

Continuous evaluation might also be used to capture images of plant fields that have to be consistent but with no functional simulation, such as nutrient fluxes and photosynthesis [14], which have inherently competing elements. Absence of functional simulation, however, does not imply that the training data lacks functional information, as structural parameters can be fitted to experimental conditions like phosphorus availability, as done by Bauer et al. [28].

3.3 HPC Scenario: Virtual Worlds for Multi-agent Systems

Field partitioning becomes necessary at the scales that we see in agriculture, as rendering an average field size of 36.4 ha [30] in high detail requires distributed rendering. Infrastructurally, there needs to be an expedient pathway towards this partitioning that does not disturb the user-centered setup for these virtual environments, a challenge we meet through both explicit and implicit scene partitioning. We illustrate this approach in Fig. 1.B, which shows the setup with the partition controller that knows the partitioning boundaries and will connect the camera agent to a specific renderer when it enters its assigned area. The areas also define uniquely what plant structure is assigned to a specific location. An instance of the FSPM is assigned a seed at the start, and all organs are stochastic realizations of the input distributions of the parameter space [13]. The upscaling of the individual FSPMs to field level yields information on between-plant competition, for example to absorb sunlight. The field partitioning is stochastic seed based, which means that there would be structural (and thus functional) consistency between the individual compute nodes. The result is a fully informed virtual field that contains agricultural information, such as plant age, health, or leaf areas. This distribution is most effective if cameras are evenly distributed to nodes.

The partitioning, as seen in Fig. 1.B, is dependent on the preemptive assignment of regions to nodes. For a specific instance of UE, the simulation only generates a subset of the field, which in turn depends on how boundary conditions are being handles. For the purposes of the transition between rendering back-ends for a continuous camera path, we include a buffer region that is shared between neighboring nodes. This region is generated in addition and does not need to be communicated, as CPlantBox can generate identical structures on demand. Which rendering node is used is defined based on position on the field, which makes it necessary for the user to set the field partition in the run script or environment variables. Using Synavis for the scene partitioning allows the change of the camera source, depending on the position of the camera agent relative to the boundaries of the scene partitioning, as illustrated in Fig. 1.B. In our setups, we pre-register the streaming connection between endpoints on

Table 1. Node configuration for our tests. Instances of UE are run on dedicated nodes.

Module	JURECA-DC
CPU	2x AMD EPYC 7742, 2 × 64 Cores, 2.25 GHz
Memory	512 (16 × 32) GB DDR4, 3200 MHz
GPU	4 × NVIDIA A100 GPU, 4 × 40 GB HBM2e
Network	2 × InfiniBand HDR (NVIDIA Mellanox Connect-X6)

the infiniband [31] network, which is dedicated to HPC users. This means that the initialization phase is being skipped and the receiving application already allocated communication ports.

3.4 Experimental Setup

We tested two configurations, which are also highlighted in Fig. 1.A. These tests were performed on nodes of the JURECA-DC GPU module [32], with the node configuration shown in Table 1. We evaluated the rendering performance through the use of Synavis, which induces the transfer of a video stream. To measure the use-case of a continuous update that is applicable to CV as described in Sect. 3.2, we tested the rendering of N instances of the FSPM CPlantBox in Experiment 1 (E1). We measured the frame time as reported by UE through Synavis and the GPU utilization via the graphics vendor driver software. Details on the software and data setup are described in Sect. 3.1. We ran UE on 4000 × 3000 pixel resolution, with VP9 encoding on CPU. The partitioning of the virtual field into instances of UE has a specific worst case, which is concurrently running the instances of UE on one GPU node. We evaluated this in E2, which is highlighted in Fig. 1.B, using four concurrent instances of UE, running with a constant stream of new geometries similar to E1, but on the same node. Here, we evaluated the frame time performance for $4N$ instances of the simulation model. In this case, we tested the framework within one module, which means that setups that need to bridge via Ethernet will be slower than our setup using Infiniband. The framework has a baseline workload resulting from rendering an empty sunlit scene, and a minimum duration for the handling of commands.

4 Results

The measurement of E1, seen in Fig. 2 yielded no fixed maximum field size, but a decreased efficiency with increasing plant number. We measured a slightly superlinear increase in frame time for more complex scenes while the efficiency of using the GPU node decreased. Figure 2 shows the relative rendering performance depending on the amount of plants rendered in the scene, each with 28 d of growth time. Frame time average increased to 0.09 s which is roughly 10.7 frames per second at about 10k plant geometries. Notably, we observe a decreased GPU utilization, which is due to GPU memory exhaustion, resulting

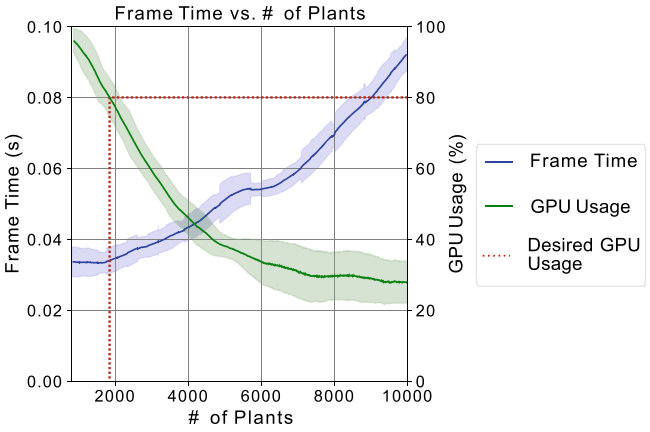

Fig. 2. Frame time measurement average for a field of N plants that are being continuously updated.

in more loading operations from the main memory. This results in a reduced efficiency of using the GPU node, which indirectly results in a decreased energy efficiency. While GPU usage is not the primary performance metric of the whole pipeline, we would like to keep this value above 80%, which referring to Fig. 2 yields an instance count of just below 2000 plants. As the simulation model and all new instances continue to be evaluated over time, Synavis constantly updates the scene and changes parameters. The highest impact to the rendering performance is entity creation, which refers to the spawning of a plant simulation model in the scene, along with registering of the object and handling of geometry information.

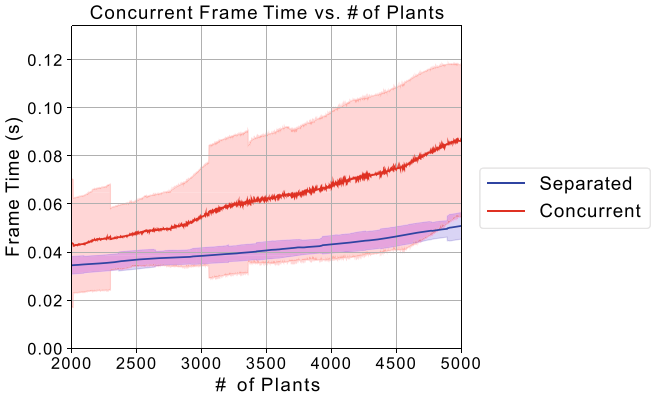

Fig. 3. Impact of rendering $N * 4$ geometries in UE concurrently on the same node as opposed to different nodes.

For E2, we show measurements of concurrent and non-concurrent rendering in UE on GPU nodes in Fig. 3. We observe, in the case of concurrently rendering four UE instances, a higher frame time on average as well as higher variance in frame time. The communication and geometric operations directly compete in this setting, for the Infiniband [31] network latency/throughput as well as in terms of memory operations. We note that we could not exclude effects from CPU usage competition between the instance of UE, because the base UE implementation does not respond to environmental variables and the parallel execution of UE threads might be suboptimal. The update time of individual plants in E1 is short enough that we are achieving a rendering time of more than 25 frames per second (0.04 s) for up to 3200 plants per GPU. An important note is that for distributed data generation, the image and environmental conditions such as weather and light need to be randomized to facilitate scalable training. Changing the settings of the virtual environment is typically done within 0.1 s from the time of prompting.

5 Discussion

The combination of simulation models, UE, and DL to enhance understanding and algorithms in plant science is a challenge both on a software level as well as infrastructural, as the necessary components have different requirements and best practices. Generally, synthetic data training is most effective with a foundational DL-model as basis, and subsequent real-world data fine-tuning. Synthetic data provides a scalable [5,12] approach to generate a diverse [6,18] data set but might not replicate all potential artifacts. In plant science, it is more impactful if rendered images contain biologically relevant information through plant simulation. Large field sizes that exceed GPU memory need to be partitioned once there are simultaneous observations in distant areas. This is partly because proper field upscaling should lift the plant model to the field scale at which agricultural decisions can be made. Our setup also allows multi-agent systems to communicate status information and the virtual scene can replicate visual information for each camera. The two main considerations that need to be discussed are how well our service-based infrastructural support performs with UE, and what we need to do to provide these pipelines to plant scientists.

5.1 Distribution Performance

The performance measurements were done purely from the standpoint of *rendering* performance. However, for a large-scale training use-case, the rendering performance might be secondary to an efficient use of tensor cores for the neural network training. Particularly if there need to be images with many plants in one view, users might forego the need for stitching and just assign all plants to one UE instance, resulting in a lower GPU utilization, which might not be critical in some cases. Furthermore, individual operations will reflect in the image data with a slight delay. Previous measurements in this framework have yielded

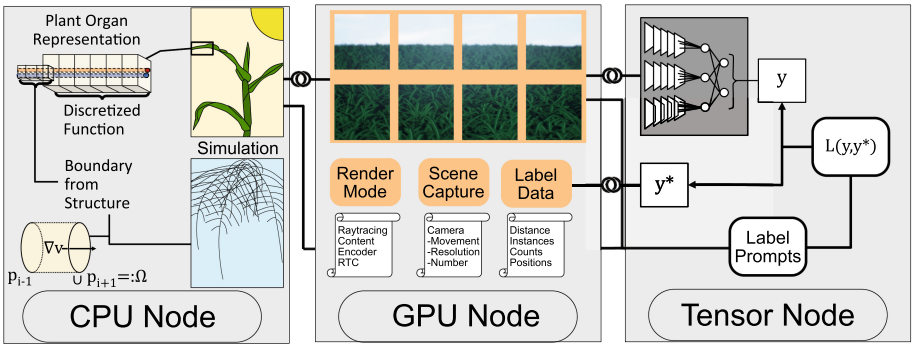

Fig. 4. Overview of technical components and data flows. The illustrated assignment to specific nodes is a performance recommendation, though individual components can share resources. A linked line indicates concurrent coupling.

a time difference between change request and confirmation of 0.1 s [12], but the execution of individual message workloads is done in bulk for certain operations (such as geometry data loading).

Streamlike data generation for CV is a suitable tool to create responsive and dynamic data that can change to the needs for the training framework. This is similar to active training frameworks, but the "rendering-in-the-loop" tuning of the virtual environment allows for a much more precise optimization of training impact. We achieve fairly good performance regarding the rendering of a large number of complex geometries. Our approach accommodates dynamic coupling such that we can partition large scales into individual instances through Synavis. As we are fully simulating the FSPMs, there are discrete updates to the scene geometry, which is not the case for pre-built geometries, but the approach is more scalable and directly visualizes biological information. The feedback loop between simulation and rendering allows for an exact quantification of either classification error thresholds (in domain-specific measures) or an analysis of robustness against scene conditions.

Dynamic camera coupling is a powerful tool to allow for a coherent field partitioning with multiple camera agents. We can render a much larger field by distributing the scene. Based on the desire to render efficiently with at least 80% GPU usage, we need to split an average of 50k plants per ha [33] into 25 GPUs per ha otherwise sacrificing GPU node efficiency as measured in E1 in Fig. 2. Transitions between individual nodes is reasonably fast with pre-registered connections (within 0.2 s). Drawbacks arise from the fact that there needs to be overlap between the areas of the field in cases of rendering simulation models to avoid information conflict, reducing the partitioning efficiency. Reconstructing a full video from this technique requires a few steps, e.g. a "fade" transition between streaming sources. Field partitioning allows for a better use of GPU resources, as we see that with more geometries on a single node (Fig. 2), the GPU utilization decreases, leading to less efficient use of compute time.

5.2 Infrastructural Support

Our example workflows consist of multiple components, that can be programmed together using Synavis. We acknowledge that the level of complexity involved in this system is high. However, due to the large amount of previous work in both the FSPM CPlantBox [13,14,16,17] as well as the compatibility provided by UE, new users will have an easier time adapting to each component individually. Due to the standardized method of communication between the frameworks, we are furthermore robust towards software version changes, which increases the inherent longevity of the framework in an everchanging HPC infrastructure landscape.

It is generally recommended that, for training purposes, multiple nodes are allocated, at least one of which would be a data generation node with UE. Figure 4 shows the components and what primary computing resource they utilize. Notably, the plant simulation produces *geometries* quite fast, while the *functional* coupling (particularly in the soil domain) requires fixed-point iteration solvers and thus might delay the digital plant evolution in cases of coupled growth. A functional coupling generally is most efficiently calculated on a shared memory system using all resources, while a purely structural simulation might run on the same node as UE. Our recommendation is to separate the neural network training and the data production, though it is possible to run these components on the same machine provided it has multiple GPUs. Particularly, this is one of the use-cases in which it is imperative to provide exclusive-use visualization or data analysis nodes to users. This is because shared GPU nodes which provide the only rendering capabilities in the system will inhibit certain scientific use-cases of HPC systems, especially in times of dedicated technologies such as tensor cores or massively-parallel GPUs as present in LUMI [34] or the planned JUPITER system [35].

6 Conclusion

Distributed rendering of plant fields for data generation is pertinent, especially in instances where large data sets, that would be cumbersome to store, are needed for the training. We enabled rendering virtual fields from plant simulations on nodes of the supercomputer JURECA-DC, highlighting continuous scene updates as well as field distribution as potential use-cases. With a strong basis of representative synthetic data, we have established techniques for distributed remote virtual environment rendering, and aim for large-scale use cases, such as light simulation [36], in the future. Furthermore, we want to extend our work on improving rendering performance and load balancing between nodes, depending on use-cases. We have shown that the parallel rendering and simulation of virtual environments is a valuable tool to establish a scalable data production pipeline and synthetic training environments for plant data analysis models, which is one of the most affected domains in terms of data scarcity and under-use of HPC infrastructure.

Acknowledgments. The authors would like to acknowledge funding provided by the German government to the Gauss Centre for Supercomputing via the InHPC-DE project (01-H17001).

This work has partly been funded by the EUROCC2 project funded by the European High-Performance Computing Joint Undertaking (JU) and EU/EEA states under grant agreement No 101101903.

This work has partly been funded by the German Research Foundation under Germany's Excellence Strategy, EXC-2070 - 390732324 - PhenoRob and by the German Federal Ministry of Education and Research (BMBF) in the framework of the funding initiative "Plant roots and soil ecosystems, significance of the rhizosphere for the bioeconomy" (Rhizo4Bio), subproject CROP (ref. FKZ 031B0909A).

The authors would like to acknowledge the compute time on the supercomputer JURECA [37] at Forschungszentrum Jülich GmbH, grant *VIS4AI*.

Data Availability Statement. Synavis is an open source repository and SynavisUE is its associated plugin for Unreal Engine. For new projects, we provide a template called SynavisUEexample. The CPlantBox official code can be found in the Plant-Root-Soil modeling group GitHub. For ensured compatibility and reproducibility, the branch associated with this paper has been forked separately. We have uploaded a video description of the method, seen in 10.6084/m9.figshare.25723773.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Minervini, M., Scharr, H., Tsaftaris, S.A.: Image analysis: the new bottleneck in plant phenotyping. IEEE Signal Process. Mag. **32**(4), 126–131 (2015). https://doi.org/10.1109/MSP.2015.2405111
2. Taiz, L.: Agriculture, plant physiology, and human population growth: past, present, and future. Theor. Exp. Plant Physiol. **25**(3), 167–181 (2013). https://doi.org/10.1590/S2197-00252013000300001
3. Tsaftaris, S.A., Minervini, M., Scharr, H.: Machine learning for plant phenotyping needs image processing. Trends Plant Sci. **21**(12), 989–991 (2016). https://doi.org/10.1016/j.tplants.2016.10.002
4. Ward, D., Moghadam, P., Hudson, N.: Deep leaf segmentation using synthetic data. In: Proceedings of the British Machine Vision Conference (2018). https://doi.org/10.48550/arXiv.1807.10931
5. Bailey, B.N.: Helios: A scalable 3D plant and environmental biophysical modeling framework. Front. Plant Sci. **10** (2019). https://doi.org/10.3389/fpls.2019.01185
6. Zhang, T., et al.: UnrealPerson: an adaptive pipeline towards costless person re-identification (2020). arXiv:2012.04268
7. Qiu, W., et al.: UnrealLCV: virtual worlds for computer vision. In: Proceedings of the 25th ACM international conference on Multimedia, pp. 1221–1224 (2017). https://doi.org/10.48550/arXiv.1609.01326
8. Pound, M.P., et al.: Deep machine learning provides state-of-the-art performance in image-based plant phenotyping. GigaScience **6**(10) (2017). https://doi.org/10.1093/gigascience/gix083

9. Scharr, H., et al.: Leaf segmentation in plant phenotyping: a collation study. Mach. Vis. Appl. **27**(4), 585–606 (2016). https://doi.org/10.1007/s00138-015-0737-3
10. Kamilaris, A., Prenafeta-Boldú, F.X.: Deep learning in agriculture: a survey. Comput. Electron. Agric. **147**, 70–90 (2018). https://doi.org/10.1016/j.compag.2018.02.016
11. Mildenhall, B., et al.: Local light field fusion: practical view synthesis with prescriptive sampling guidelines. ACM Trans. Graph. (TOG) (2019). https://doi.org/10.1145/3306346.3322980
12. Helmrich, D.N., et al.: A scalable pipeline to create synthetic datasets from functional-structural plant models for deep learning. Silico Plants **6**(1), diad022 (2023). https://doi.org/10.1093/insilicoplants/diad022
13. Zhou, X.R., et al.: CPlantBox, a whole-plant modelling framework for the simulation of water- and carbon-related processes. Silico Plants **2**(1) (2020). https://doi.org/10.1093/insilicoplants/diaa001
14. Giraud, M., et al.: CPlantBox: a fully coupled modelling platform for the water and carbon fluxes in the soil-plant-atmosphere continuum. Silico Plants **5**(2) (2023). https://doi.org/10.1093/insilicoplants/diad009
15. Lobet, G., Koevoets, I.T., Noll, M., Meyer, P.E., Tocquin, P., Pagès, L., Périlleux, C.: Using a structural root system model to evaluate and improve the accuracy of root image analysis pipelines. Front. Plant Sci. **8** (2017). https://doi.org/10.3389/fpls.2017.00447
16. Schnepf, A., Huber, K., Landl, M., Meunier, F., Petrich, L., Schmidt, V.: Statistical characterization of the root system architecture model crootbox. Vadose Zone Journal **17**(1) (2018). https://doi.org/10.2136/vzj2017.12.0212
17. Morandage, S., Laloy, E., Schnepf, A., Vereecken, H., Vanderborght, J.: Bayesian inference of root architectural model parameters from synthetic field data. Plant Soil **467**(1), 67–89 (2021). https://doi.org/10.1007/s11104-021-05026-4
18. Zhang, Y., Qiu, W., Chen, Q., Hu, X., Yuille, A.: Unrealstereo: controlling hazardous factors to analyze stereo vision (2018)
19. McCormac, J., Handa, A., Leutenegger, S., Davison, A.J.: Scenenet RGB-D: can 5M synthetic images beat generic ImageNet pre-training on indoor segmentation? In: Proceedings of the IEEE International Conference on Computer Vision (ICCV) (2017). https://doi.org/10.1109/ICCV.2017.292
20. Roberts, M., et al.: Submodular trajectory optimization for aerial 3D scanning. CoRR abs/1705.00703 (2017)
21. Nassif, J., Tekli, J., Kamradt, M.: Digital Images – The Bread and Butter of Computer Vision, pp. 89–106. Springer Nature Switzerland, Cham (2024). https://doi.org/10.1007/978-3-031-47560-3_5
22. Bondi, E., et al.: AirSim-W: a simulation environment for wildlife conservation with UAVs. COMPASS 2018, Association for Computing Machinery, New York, NY, USA (2018). https://doi.org/10.1145/3209811.3209880
23. Mu, J., Qiu, W., Hager, G.D., Yuille, A.L.: Learning from synthetic animals. CoRR abs/1912.08265 (2019)
24. Berrigan, E.M., et al.: Fast and efficient root phenotyping via pose estimation. Plant Phenomics **6**, 0175 (2024). https://doi.org/10.34133/plantphenomics.0175
25. Aumüller, M.: The architecture of vistle, a scalable distributed visualization system. In: Markidis, S., Laure, E. (eds.) Solving Software Challenges for Exascale, pp. 141–147. Springer International Publishing, Cham (2015). https://doi.org/10.1007/978-3-319-15976-8_11

26. Larsen, M., Meredith, J.S., Navrátil, P.A., Childs, H.: Ray tracing within a data parallel framework. In: 2015 IEEE Pacific Visualization Symposium (PacificVis), pp. 279–286 (2015). https://doi.org/10.1109/PACIFICVIS.2015.7156388
27. Moreland, K., et al.: VTK-M: accelerating the visualization toolkit for massively threaded architectures. IEEE Comput. Graph. Appl. **36**(3), 48–58 (2016). https://doi.org/10.1109/MCG.2016.48
28. Bauer, F.M., et al.: In silico investigation on phosphorus efficiency of ZEA mays: an experimental whole plant model parametrization approach (2023). https://doi.org/10.34734/FZJ-2023-04031
29. Karis, B., Stubbe, R., Wihlidal, G.: A deep dive into unreal engine 5's nanite. In: SIGGRAPH (2021)
30. White, E.V., Roy, D.P.: A contemporary decennial examination of changing agricultural field sizes using landsat time series data. Geo: Geography Environ. **2**(1), 33–54 (2015). https://doi.org/10.1002/geo2.4
31. Pentakalos, O.I.: An introduction to the infiniband architecture. In: International CMG Conference (2002). https://doi.org/10.1109/9780470544839.ch42
32. Thörnig, P.: JURECA: Data centric and booster modules implementing the modular supercomputing architecture at Jülich supercomputing centre. J. Large-Scale Res. Facil. JLSRF (2021). https://doi.org/10.17815/jlsrf-7-182
33. Zhang, Y., Xu, Z., Li, J., Wang, R.: Optimum planting density improves resource use efficiency and yield stability of rainfed maize in semiarid climate. Front. Plant Sci. **12** (2021). https://doi.org/10.3389/fpls.2021.752606
34. Markomanolis, G.S., et al.: Evaluating GPU programming models for the Lumi supercomputer. In: Panda, D.K., Sullivan, M. (eds.) Supercomputing Frontiers. Springer International Publishing (2022). https://doi.org/10.1007/978-3-031-10419-0_6
35. Shapiro, A.: Nvidia grace hopper superchip powers Jupiter, defining a new class of supercomputers to propel AI for scientific discovery. NVIDIA Enterprise Networking Press Release (2023). https://nvidianews.nvidia.com/news/
36. Malenovský, Z., et al.: Discrete anisotropic radiative transfer modelling of solar-induced chlorophyll fluorescence: structural impacts in geometrically explicit vegetation canopies. Remote Sens. Environ. **263** (2021). https://doi.org/10.1016/j.rse.2021.112564
37. Krause, D., Thörnig, P.: JURECA: modular supercomputer at Jülich supercomputing centre. J. Large-Scale Res. Facil. JLSRF (2018). https://doi.org/10.17815/jlsrf-4-121-1