UNIVERSITY OF ICELAND

Design and Evaluation of Data Analysis and Augmentation Approaches for Functional-Structural Plant Models

Dirk Norbert Baker

2025

FACULTY OF INDUSTRIAL ENGINEERING, MECHANICAL ENGINEERING AND COMPUTER SCIENCE

Design and Evaluation of Data Analysis and Augmentation Approaches for Functional-Structural Plant Models

Dirk Norbert Baker

Dissertation submitted in partial fulfillment of a
Philosophiae Doctor degree in Computational Engineering

Supervisor
Prof. Dr.-Ing. Morris Riedel

Doctoral Committee
Prof. Dr.-Ing. Morris Riedel
Prof. Dr. Ebba Þóra Hvannberg
Dr. Hanno Scharr
Jens Henrik Göbbert

Opponents
Dr. Thomas Odaker
Prof. Dr. Michael Pound

Faculty of Industrial Engineering, Mechanical Engineering and Computer Science
School of Engineering and Natural Sciences
University of Iceland
Reykjavík, 5 2025

Design and Evaluation of Data Analysis and Augmentation Approaches for Functional-Structural Plant Models
(Virtual Worlds for Machine Learning)

Dissertation submitted in partial fulfillment of a *Philosophiae Doctor* degree in Computational Engineering

Faculty of Industrial Engineering, Mechanical Engineering and Computer Science
School of Engineering and Natural Sciences
University of Iceland
Tæknigarður - Dunhagi 5
107 Reykjavik
Iceland

Telephone: 525-4000

Bibliographic information:
Dirk Norbert Baker (2025) *Design and Evaluation of Data Analysis and Augmentation Approaches for Functional-Structural Plant Models*, PhD dissertation, Faculty of Industrial Engineering, Mechanical Engineering and Computer Science, University of Iceland, 134 pp.

ISBN 978-9935-9837-1-8

Printing: Háskólaprent
Reykjavik, Iceland, 5 2025

Abstract

Analyzing plant data for precision and robust agriculture is a key component of adapting to climate change. Plant data is multi-modal, experiments are directly restricted by plant growth, and the data analysis is often supported by image capturing. Particularly in the case of functional-structural plant models, the understanding of experimental data, the primary indicators for plant development and health, is crucial to calibrating robust and representative models of the real world. However, experimental data is also limited in its scope, which makes it more crucial that the data analysis pipeline is not only robust against noise but also yields as much value to researchers as possible. The focus of this thesis is the embedding of plant models into a virtual environment in such a way that it is informed by experimental data and compatible with both a more precise extraction pipeline and a more scalable data generation pipeline. Synthetic data generation is a key aspect of this thesis, as it is one of the promising ways to combat data scarcity for biological data analysis. This thesis contributes to the field by establishing a data generation framework that is compatible with modern high-performance computing systems and is based on a realtime communication standard. The virtual world embedding of the plant models also yields the possibility of measuring in-place, allowing the coupling of plant models to visualization to exhibit true digital twin behavior. Within the virtual embedding, data extraction is also more precise, allowing users to directly interact with challenging data sets to increase the precision and robustness compared to traditional methods. The output of this PhD thesis is a distributed synthetic data-based training framework called Synavis, full realistic virtual scenes that contain functional information and are computed scalably, and a data analysis pipeline called VRoot that directly assists researchers in their data analysis whenever automated approaches need human intervention.

Útdráttur

Lykilþáttur í aðlögun að loftslagsbreytingum, er að geta greint plöntugögn, sem fengin eru úr landbúnaði, nákvæmlega og áreiðanlega. Plöntugögn eru fjölhátta, tilraunir eru takmarkaðar af plöntuvexti og gagnagreining er oft studd af myndatöku. Sérstaklega í tilfelli virknibyggðra plöntulíkana er skilningur á tilraunagögnum, sem eru helstu vísbendingum um þróun og heilsufar planta, afar mikilvægur til að fínstilla áreiðanleg og raunhæf líkön af raunveruleikanum. Hins vegar eru tilraunagögn takmörkuð að umfangi og gæðum, sem gerir það enn mikilvægara að gagnaúrvinnsla sé ónæm fyrir gagnasuði og sé eins gagnleg vísindamönnum og hægt er. Áhersla þessarar ritgerðar er á innfellingu plöntulíkana í sýndarveruleika þannig að hann fái upplýsingar um tilraunagögn og sé samhæfður við nákvæmari útdráttarleið og stigfrjálsa gagnaúrvinnslukeðju. Myndun gervigagna er lykilatriði í þessari ritgerð, þar sem hún er ein af efnilegustu aðferðunum til að bæta upp skort á líffræðilegum gögnum. Þessi ritgerð leggur af mörkum til fræðasviðsins með því að þróa gagnasköpunarkerfi sem er samhæft við nútímalegar ofurtölvur og byggir á rauntíma samskiptastaðli. Innfelling plöntulíkana í sýndarveruleika opnar einnig möguleikann á mælingum á staðnum, sem gerir kleift að tengja plöntulíkön við sjónræna framsetningu og skapa raunverulega stafræna tvíburahegðun. Innan sýndarveruleikans verður gagnaútdráttur einnig nákvæmari, sem gerir notendum kleift að hafa bein samskipti við krefjandi gagnasöfn og auka þannig nákvæmni og áreiðanleika samanborið við hefðbundnar aðferðir. Afurð þessarar doktorsritgerðar er dreifður þjálfunarrammi fyrir gervigögn, kallaður Synavis, fullkomlega raunsæjar sýndarveruleikasenur sem innihalda virkniupplýsingar og eru reiknaðar á stigfrjálsan hátt, og gagnaúrvinnslukeðja nefnd VRoot, sem leyfir vísindamönnum að grípa inn í sjálfvirka gagnaúrvinnslu þegar að þörf er á.

Contents

List of Figures

List of Tables

List of Original Publications

Article I: D. N. **Baker**, F. M. Bauer, M. Giraud, A. Schnepf, J. H. Göbbert, H. Scharr, E. Þ. Hvannberg, and M. Riedel. "A scalable pipeline to create synthetic datasets from functional–structural plant models for deep learning." In: *in silico Plants* 6.1 (Dec. 2023), diad022. ISSN: 2517-5025 DOI: 10.1093/insilicoplants/diad022.

Article II: D. N. **Baker**, T. Selzner, J. H. Göbbert, H. Scharr, M. Riedel, E. þ. Hvannberg, A. Schnepf, and D. Zielasko. "VRoot: A VR-Based Application for Manual Root System Architecture Reconstruction." In: *Plant Phenomics* (2025), p. 100013. ISSN: 2643-6515 DOI: 10.1016/j.plaphe.2025.100013.

Article III: D. N. **Baker**, F. Bauer, A. Schnepf, H. Scharr, M. Riedel, J. H. Göbbert, and E. Hvannberg. "Adapting Agricultural Virtual Environments in Game Engines to Improve HPC Accessibility." In: *Nordic e-Infrastructure Tomorrow*. Ed. by A. Azab and T. Malkiewicz. Cham: Springer Nature Switzerland, 2025, pp. 152–167. ISBN: 978-3-031-86240-3 DOI: 10.1007/978-3-031-86240-3_11.

Article IV: D. N. **Baker**, M. Giraud, J. H. Goebbert, H. Scharr, M. Riedel, E. T. Hvannberg, and A. Schnepf. "Virtual World Coupling with Photosynthesis Evaluation for Synthetic Data Production." 2025 DOI: 10.1101/2025.02.06.633870.

List of Other Contributions

Note:	The PhD Candidate changed their name from Dirk Norbert Helmrich to Dirk Norbert Baker during the course of the PhD.
Contribution A:	D. N. **Helmrich**, J. H. Göbbert, M. Giraud, H. Scharr, A. Schnepf, and M. Riedel. "Towards Large-Scale Rendering of Simulated Crops for Synthetic Ground Truth Generation on Modular Supercomputers." In: *IEEE Symposium on Large Data Analysis and Visualization (LDAV)*. 2021 DOI: 10.48550/arXiv.2110.14946.
Contribution B:	A. C. Demiralp, D. N. **Helmrich**, J. Protze, T. W. Kuhlen, and T. Gerrits. "Performance Assessment of Diffusive Load Balancing for Distributed Particle Advection." In: *CSRN*. Západočeská univerzita, 2022 DOI: 10.24132/csrn.3201.2.
Contribution C:	T. Selzner, J. Horn, M. Landl, A. Pohlmeier, D. N. **Helmrich**, K. Huber, J. Vanderborght, H. Vereecken, S. Behnke, and A. Schnepf. "3D U-Net Segmentation Improves Root System Reconstruction from 3D MRI Images in Automated and Manual Virtual Reality Work Flows." In: *Plant Phenomics* 5 (2023), p. 0076 DOI: 10.34133/plantphenomics.0076.
Contribution D:	F. M. Bauer, D. N. **Baker**, M. Giraud, J. C. Baca Cabrera, J. Vanderborght, G. Lobet, and A. Schnepf. "Root system architecture reorganization under decreasing soil phosphorus lowers root system conductance of Zea mays." In: *Annals of Botany* (Nov. 2024), mcae198. ISSN: 0305-7364 DOI: 10.1093/aob/mcae198.
Contribution E:	D. N. **Baker**, T. Selzner, J. H. Göbbert, H. Scharr, M. Riedel, E. Þ. Hvannberg, A. Schnepf, and D. Zielasko. "Hands-On Plant Root System Reconstruction in Virtual Reality." In: *Proceedings of the 30th ACM Symposium on Virtual Reality Software and Technology*. VRST '24. Trier, Germany: Association for Computing Machinery, 2024. ISBN: 9798400705359 DOI: 10.1145/3641825.3689494.
Contribution F:	T. Selzner, D. N. **Baker**, M. Landl, D. Leitner, G. Lobet, and A. Schnepf. "Coupling of MRI and modelling." In: *NMR in Plants and Soils*. Ed. by A. Pohlmeier, S. Haber-Pohlmeier, and S. Stapf. Royal Society of Chemistry, 2025. Chap. 16, in press.

Abbreviations

AI Artificial Intelligence

CNN Convolutional Neural Network

CPU Central Processing Unit

DNN Deep Neural Network

FSPM Functional-Structural Plant Model

GPU Graphical Processing Unit

HMD Head-Mounted Display

HPC High-Performance Computing

HTTP Hypertext Transfer Protocol

IA Immersive Analytics

JSON JavaScript Object Notation

MPI Message Passing Interface

RSA Root System Architecture

RTC Real-Time Communication

RTP Real-Time Protocol

RSML Root System Markup Language

SSH Secure Shell

UDP User Datagram Protocol

UE Unreal Engine

VR Virtual Reality

Acknowledgments

This thesis would not have been possible without the support of the staff at the Jülich Helmholtz Research Centre, particularly the institutes Jülich Supercomputing Centre and Institute for Bio- and Geosciences 3: Agrosphere. Their continuous support in the face of my questions and requests was invaluable.

My committee, Prof. Dr.-Ing. Morris Riedel, Prof. Dr. Ebba Þóra Hvannberg, Dr. Hanno Scharr, and Jens Henrik Göbbert were a great source of feedback and support. The collaboration between Germany and Iceland was a great opportunity to not only collaborate on scientific and technical aspects but also broaden my horizons.

I want to thank my colleagues and friends, particularly Dr. Tobias Selzner, Dr. Felix Bauer, Mona Giraud, Dr. Magdalena Landl, and Dr. Daniel Leitner, for their continuous support and feedback.

I am also very grateful to my colleagues Dr. Juan Baca Cabrera, Jonathan Windgassen, Dr. Herwig Zilken, Dr. Helmut Schumacher, Sonja Habbinga, Prof. Dr. Jan Vanderborght, Samuel Le Gall, Dr. Andreas Pohlmeier, Thomas Schuster, Dr. Marcel Aach, Prof. Dr.-Ing. Gabriele Cavallaro, Amer Delilbasic, Tim Kreuzer, Dr.-Ing. Bernd Mohr, Thomas Plaga, Dr. Rocco Sedona, Surbhi Sharma, Sandra Strick, Dr. Kay Thust, Dr. Daniel Pflugfelder, and Dr. Dagmar van Dusschoten.

I would like to express my gratitude to the staff of the University of Iceland, particularly Toby Erik Wikström, Prof. Dr. Matthias Book, Liang Tian, Sigríður Daney Sigurðardóttir, and John Denver Rafael Florentino.

My friend Dr. Daniel Zielasko was instrumental in the realization of the Virtual Reality application and even more so in the realization of the user study that was conducted to evaluate the application. The experience of rigorously going through the scientific process on such a project was one of the quintessential experiences of my PhD.

Prof. Andrea Schnepf of the Institute for Bio- and Geosciences 3: Agrosphere was a constant source of inspiration and a well of knowledge (and patience regarding my initial lack of biological knowledge). Especially the ability to collaborate with her on numerous points of this thesis that particularly intersected multiple domains was a great opportunity.

My family, particularly my wife Yarê, was a well of support, inspiration, and strength. Without her, this thesis would not have been possible. We embarked on this journey together, and I am grateful for her support and love.

1 Introduction

Facing climate change and changing conditions in agriculture, there is a large push towards revolutionizing land use and crop production. Particularly in a global effort to eliminate hunger and malnutrition by organizations such as the World Health Organization, the need for precision agriculture is growing [FAO24]. These sweeping changes depend on a solid and in-depth understanding of crop systems and plant species, a large technological advance in agricultural practices, and a sustainable view and concept for land use. One important cornerstone of this understanding is the use of models and simulations, informed by digitized experiments, that fill the gaps in experimental data [Jon+20]. These gaps might be caused by the limitations of the experimental setup, such as not being able to excavate the root system of a plant without destroying it or by the ambition to portray a larger scale while retaining the same amount of detail. However, a good basis for the models through the careful use of experimental data is necessary to be able to describe the complex effects we are seeing in the field.

1.1 Motivation

Diverse measurements can be performed on a single plant experiment, but these measurements are often supported in large parts by captured images [MST15]. Many aspects of plant development need understanding, which rests on a full tracking of the plant development in experimental settings. This includes growth rate, time points of organ emergence, senescence, and more. Captured images are very useful since plants exhibit structural changes when their conditions change, such as changes in root elongation due to phosphorus deficiency [Bau+24]. This emerging behavior might change the structure of captured images, changes that might be uncovered if enough discriminative examples are provided. This introduces the primary limitations of many data analysis pipelines. High-effort annotation pipelines need to be established to uncover such effects. Particularly in cases as implemented by Bauer et al. [Bau+24] who require segmented images or by Ward et al. [WMH18] who require leaf identification, these annotations are required for the data analysis, but themselves do not carry domain knowledge.

The central research areas of this thesis are therefore how virtual world workflows can assist in these use cases, particularly to alleviate the burden of measuring auxiliary data and to increase extraction accuracy. Moreover, how can we make the best use of the present experiments that have been carried out to limit unnecessary sources of uncertainty in data extraction? What steps can we take to fill the gaps between our understanding of the models and our data analysis in the field? How can our models help us scale our workflows to provide a greater database of auxiliary measurements

that are needed but not domain-specific?

The research in this thesis aims to make better use of the experimental data to inform models and to use these models to generate data that can assist in the analysis of plant images by Deep Neural Networks (DNNs). Experimental data that is needed to calibrate models is a focus of the research, yielding pipelines to generate auxiliary data measurements and to enable a more precise data extraction. There is a great deal of possibility regarding the employment of synthetic data. Taiz [Tai13] detailed a potential future of agriculture in which automated remote sensing systems are being used to quantitatively assess crop state based on models of the real world. These models have limited expressiveness unless supplied with a large and representative data set. There are cornerstone measurements that can be used to reduce the uncertainty in the data [Mor+21a], but these measurements are often destructive, e.g. require excavation, and cannot be performed on a larger data set.

Three key aspects of data extraction and analysis motivate this thesis. First, this thesis aims to enable a direct but scalable coupling between models and virtual worlds when creating synthetic data. While generative networks have been proposed for this task, this thesis focuses on model-based data generation. As Hartley et al. [Har+21] based on Zhu et al. [Zhu+17] found, generative networks have failure cases in which they do not preserve the intrinsic logic of the scene. Nonetheless, generative networks have a place in synthetic data generation in many circumstances, as they generalize some aspects of the model space and might increase generalizability [Har+21]. A model-based approach, while requiring more understanding of processes being simulated, is free of such issues if the parametrization is grounded in experimental data and the visualization is faithful to the domain knowledge included in such a model. Furthermore, the thesis motivation is the fact that the absolute difference between the DNN output and the synthetic label data is helpful and can provide insights into exactly which cases need revisiting in the course of the training [Hel+21].

Second, as reported recently by Li et al. [Li+24], the data pipelines that are used for training are often non-reactive and unidirectional. However, the primary advantage of synthetic data, aside from enabling domain adaptation [Zha+20], is that it is not restricted to a sequential process and can be used concurrently with the DNN training. Both Li et al. [Li+24] and Zhang et al. [Zha+20] establish synthetic data pipelines using a popular graphics engine, Unreal Engine (UE)[1], developed by Epic Games Inc. This allows for the use of common techniques to establish scene variety and a baseline of representability, as UE is an engine developed for natural and realistic scenes. Particularly the built-in support for outdoor virtual environments makes it a strong choice for such workflows. Training and visualization might require separate hardware, therefore requiring the orchestration of distributed data streams. This is particularly impactful if active learning frameworks are taken into account, as the selection of data based on previous training performance [Agh+19] can help the downstream tasks of increased performance, particularly in important underperforming cases [Cao+20]. Early community feedback on synthetic data gathered in contribution A (**C-A**) [Hel+21] also yielded that model embedding is necessary for successful and robust data generation pipelines. In general, the generation of synthetic data is focused on large-scale systems such as High-Performance Computing (HPC) systems, which enable greater scalability

[1] https://www.unrealengine.com/

of training approaches for DNNs [Sed+19].

Third, when DNNs are employed, the actual extraction quality strongly depends on the input data. This is especially true when implementing pipelines using Convolutional Neural Network (CNN) models, which are a subclass of DNN approaches. Horn et al. [Hor+21] uncovered cases of their automated tracing pipeline, supported by CNN segmentation, in which the automated algorithm failed to identify the data true to the expert annotation used as a baseline. The reason for this is the fact that there are certain domain knowledge aspects not presented in the data that, if a network was trained to detect such effects, could lead to a higher rate of mislabeling. The case provided in Horn et al. [Hor+21] was a case of a larger gap in the root system which even human but untrained evaluators would have trouble identifying, as certain expert knowledge is required to know how a specific plant would behave in the context of the data. As our synthetic data ultimately depends on well-callibrated models, the stage of extracting the data necessary for that step is an important cornerstone, particularly regarding the so-called hidden part [Atk+19] of the plant.

With these challenges in mind, this thesis aims to offer an interactive and robust solution for the data extraction pipeline and to enable the generation of synthetic data on dedicated visualization nodes of HPC systems. One concrete motivator to provide an interactive pipeline, i.e., as opposed to a data generator, is the fact that steering is useful in situations in where the data production is computationally intensive as in Roberts et al. [Rob+21] due to the increased effort in making the scenes photorealistic, or if the goal of the training cannot be defined in terms of the data produced, as has been shown by Li et al. [Li+24] who rely on performance indicators outside of the virtual world.

1.2 Thesis Aims

Concisely, this thesis aims to conceive and implement a pipeline that can output synthetic image data that can be annotated based on a semantic virtual world model. This virtual world model should be scalable, customizable, and efficient. To this end, this thesis will directly contribute to the field by extending how we embed and interact with Functional-Structural Plant Models (FSPMs) [Sie+14; Sou+21] in 3D environments. The representations will aim at producing visual models for both human and computer vision, allowing researchers to maximize their experimental data yield even in the face of difficult data acquisition. For the scalability of our visual embeddings, this thesis showcases the virtual world model pipeline on HPC systems, such as JURECA-DC [Thö21].

Overall, this thesis aims at investigating the following research questions that make up its objectives (TO):

(TO**1**) How should an FSPM be embedded into a 3D virtual environment to enable and support the generation of synthetic data?

(TO**2**) Does the direct and immersive visualization of model data in a virtual environment assist in the data extraction process, compared to traditional methods?

(TO**3**) What are the primary use cases of visualizing plant and field models in virtual

environments, and how can these be distributed on nodes of HPC systems?

(TO**4**) Is the virtual world embedding sufficiently faithful to the simulation model to provide input data for a photosynthesis evaluation?

The objectives and their components, as well as their relation to one another, are displayed in Fig. 1.1. There is a strong interconnection, primarily driven by the virtual world embedding. TO**1**, shown in red, includes not only the Synavis framework but also a prescribed method to embed the FSPM CPlantBox [Zho+20] in a 3D virtual environment. This core feature is an example of a domain interpretation that has multiple uses, extending into both TO**2** and TO**4**. TO**2**, shown at the bottom, is an analysis of the robustness of this embedding and if it can be used by researchers to not only annotate data, but to better visualize the data to assist their workflows in basic and noisy examples. Its primary output is a quantitative analysis of the immersive technology, a Head-Mounted Display (HMD), in the context of Root System Architecture (RSA) extraction workflow. Interactive tools are required to have more robustness in their visual representation - which means that lessons learned from the virtual embedding in TO**2** can be used to inform the embedding in TO**1**. However, TO**2** also influences all virtual world models by providing a baseline accuracy for our model embedding, as we are never exclusively generating synthetic data, but instead use models calibrated on real data in our scenes. This calibration is important and the quantification of its accuracy critical, particularly with human-in-the-loop techniques. Both TO**3** and TO**4** are specifically designed towards HPC workflows and include key aspects that are of vital importance regarding the employment of synthetic data pipelines. As previously reported, e.g. by Li et al. [Li+24], many data pipelines used for data production are non-reactive. They reason that the data generation and the training process are usually separate, and the data production is entirely unidirectional, while they include a model of a robot arm that can interact with the environment. TO**3** introduces the domain-based parallelization on HPC nodes that are aimed at distinct use cases. TO**4** employs an embedding of a simulation model that not only uses the virtual scene as base measurement but also enhances it by employing a coupling of models and an embedding of more complex information onto synthetic data.

1.3 Thesis Outline

This thesis is written in a cumulative format, meaning that the completion of the thesis' objectives is based on the completion of several publications listed below. These publications are detailed in Chapter 5, and the full text of the papers is attached to this thesis. Furthermore, this thesis has the following structure:

Chapter 1 introduces the scope of this thesis, its objectives, and the relationship between them, as well as a fundamental introduction to the problem statement.

Chapter 2 describes the central background technologies and approaches that are required to understand this work, including the important information on machine learning, plant modeling, and visualization.

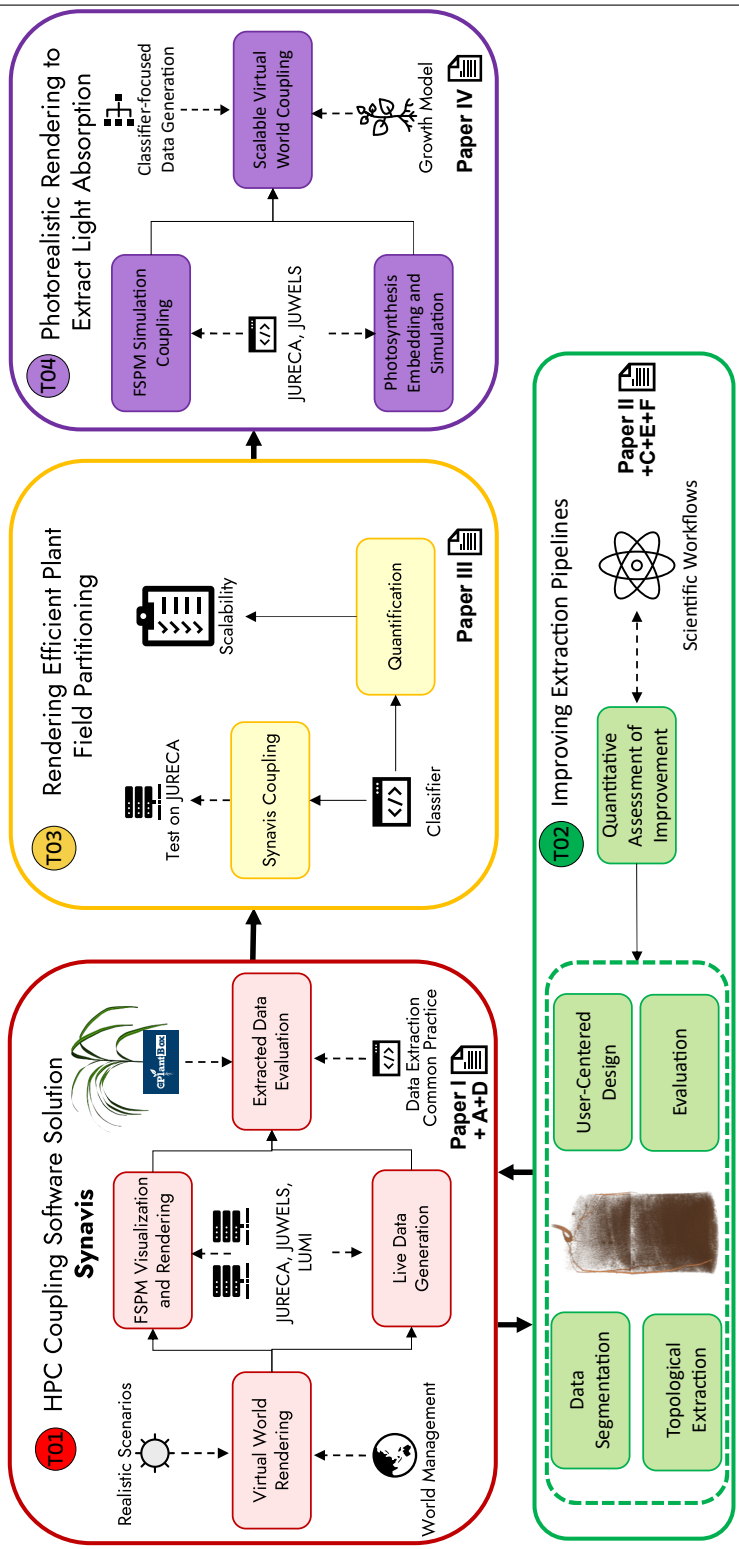

Figure 1.1. Overview of the thesis objectives and the specific components developed to address them in this thesis.

Chapter 3 surveys related works that are connected or have had an impact on the fields and challenges of importance to this research.

Chapter 4 details the implementation of the Synavis framework and the visualization of the FSPM CPlantBox in the Unreal Engine (UE).

Chapter 5 summarizes and reiterates the key points of the main contributions to this cumulative thesis.

Chapter 6 is a closing summary and a review of the current state of the topic and potential future directions that need further exploration.

The research of this thesis, in addition to being published as software products or open research data, has been published in the following papers:

Paper I D. N. **Baker**, F. M. Bauer, M. Giraud, A. Schnepf, J. H. Göbbert, H. Scharr, E. Þ. Hvannberg, and M. Riedel. "A scalable pipeline to create synthetic datasets from functional–structural plant models for deep learning." In: *in silico Plants* 6.1 (Dec. 2023), diad022. ISSN: 2517-5025 DOI: 10.1093/insilicoplants/diad022.

The PhD candidate has authored the majority of the text of the paper. The main contribution towards the publication of this article is the development and open-access publication of the Synavis[2] framework, along with the establishment of the embedding of the FSPM CPlantBox[3] in the UE. Furthermore, the evaluation and analysis of the framework was conducted by the PhD candidate.

Paper II D. N. **Baker**, T. Selzner, J. H. Göbbert, H. Scharr, M. Riedel, E. þ. Hvannberg, A. Schnepf, and D. Zielasko. "VRoot: A VR-Based Application for Manual Root System Architecture Reconstruction." In: *Plant Phenomics* (2025), p. 100013. ISSN: 2643-6515 DOI: 10.1016/j.plaphe.2025.100013.

The PhD candidate has authored the majority of the text of the paper. The main contribution towards the publication of this article is the development and open-access publication of the VRoot[4] tool. The laboratory user study was conceived and designed by the PhD candidate, and it was executed, processed, and analyzed by the PhD candidate.

Paper III D. N. **Baker**, F. Bauer, A. Schnepf, H. Scharr, M. Riedel, J. H. Göbbert, and E. Hvannberg. "Adapting Agricultural Virtual Environments in Game Engines to Improve HPC Accessibility." In: *Nordic e-Infrastructure Tomorrow*. Ed. by A. Azab and T. Malkiewicz. Cham: Springer Nature Switzerland, 2025, pp. 152–167. ISBN: 978-3-031-86240-3 DOI: 10.1007/978-3-031-86240-3_11.

[2] https://github.com/dhelmrich/Synavis

[3] https://github.com/Plant-Root-Soil-Interactions-Modelling/CPlantBox/pull/101

[4] https://github.com/dhelmrich/VRoot

The PhD candidate, along with the extension of the Synavis framework to support scene distribution on modular supercomputers. The candidate designed the scene steering, inspired by plant modeling use-cases, and conducted the performance analysis. The PhD candidate has authored the majority of the text.

Paper IV D. N. **Baker**, M. Giraud, J. H. Goebbert, H. Scharr, M. Riedel, E. T. Hvannberg, and A. Schnepf. "Virtual World Coupling with Photosynthesis Evaluation for Synthetic Data Production." 2025 DOI: 10.1101/2025.02.06.633870.

The PhD candidate extended the Synavis framework to support the coupling of a simulation model to the virtual environment. The candidate co-designed the simulation workflow that describes the virtual field, conducted the comparison and data analysis to the Selhause experimental data, and conducted the scalability experiments. The PhD candidate has authored the majority of the text.

This thesis is furthermore based on the following code repositories:

Synavis Framework, a C++ WebRTC Bridge for Unreal Engine, compatible with large-scale distributed systems, orchestrating data streams across nodes. This is the code of **Paper I**, extended in **III+IV**, available at github.com/dhelmrich/Synavis

VRoot Immersive data extraction tool for root-soil scans, allowing for diverse posture annotation and is directly outputting plant model data from the annotations, allowing for a seamless interplay between data and model. This is the code of **Paper II**, available at github.com/dhelmrich/VRoot

1.4 Contributions

This thesis is motivated by an increasing need for data sets in diverse domains but with a focus on plant science. The infrastructural question to be posed is how HPC centers can meet this data-driven demand and what use cases present themselves. Synthetic data, particularly with the growing diversity of artificial data sets, is a specific focus of the efforts to scale the training of modern DNNs. Table 1.1 shows the relative assignment of thesis objectives to the papers referenced in this thesis. Although Fig. 1.1 shows the workflow of the tasks contributing to the thesis, the elements have more interconnection than the figure suggests. Particularly, certain elements of the pipelines generalize to other thesis objectives, and all tasks individually contribute to the software stack that is one of the tangible research outcomes of this thesis, released as open source to the larger research community.

Paper I addresses key components of establishing a synthetic data pipeline environment for TO**1**. TO**1** is the basis of most works, primarily regarding the Synavis framework [Bak+23]. The coupling between the model and the UE is described by the communication framework, which is an important aspect primarily of TO**1** and TO**3–4**. However, the geometric embedding plays a key role in the implementation of all thesis objectives. Rules that define how the graph structure is mapped into the virtual environment need to be consistent for annotation workflows to succeed, making this a vital aspect of TO**2** as well. Likewise, the data extraction pipeline established in TO**2**

provides a vital baseline accuracy that is needed to produce faithful models of plants in use in other objectives.

To uncover the primary challenges and aspects of synthetic data generation that have the highest need for improvement, TO**1** includes extensive literature research as well. Baker et al. [Bak+23] provides an overview of the state-of-the-art of synthetic data generation. The gap identified in this literature research was the fact that synthetic data pipelines, while enabling essentially costless domain adaptation [Zha+20], struggle to accommodate greater workloads and HPC pipelines. With the Synavis framework, researchers can now generate synthetic data on HPC systems on-the-fly through dynamic scene creation.

Both TO**1** and TO**3** detail specific aspects of the HPC capability. With more technical limitations, i.e., security aspects, of HPC systems in mind, a central cornerstone of TO**1** is the bridging capability of the Synavis framework, allowing WebRTC [JBB21] workflows to be set up even in the face of indirect network access. However, scalability analysis focused primarily on latency or data generation, as common definitions of parallel efficiency require domain partition and a problem whose solution has a well-defined last step [EZL89]. For TO**1**, it was further explored what software is needed to support HPC workflows. In TO**3**, specific methods were investigated regarding the distribution of virtual environments for two specific domain use cases. In TO**4**, one such use case was further developed by coupling the domain decomposition to a simulation model that would yield the light influx of an FSPM, referencing a specific experiment and day. For TO**4**, further scalability measurements were undertaken, yielding a complete view of how the parallelization of light influx measurements performs. With the added functionality in Synavis developed for TO**3**, researchers can now distribute virtual scenes on clusters in a more controlled manner. The result of the coupling in TO**4** is that agricultural scientists can now develop pipelines to train their models on functional aspects, such as light absorption efficiency, a key aspect of whether to add more nitrogen to the fields [Bak+25c].

TO**2** took the evaluation of the virtual embedding of the underlying structures regarding plant models to a more concrete level. Based on previous reports of automatic data extraction failures, VRoot was developed as a high-precision extraction tool. It allows the embedding of 3D scans in a virtual immersive environment, where users annotate a RSA directly on top of the scans. In a controlled user study, the immersive and desktop approaches were compared, yielding a quantification of the impact of

	Paper I	Paper II	Paper III	Paper IV
TO**1**	x		x	x
TO**2**	x	x		
TO**3**			x	x
TO**4**			x	x
Transfer	x	x	x	x

Table 1.1. Contribution of the individual publications to the primary thesis objectives. Transfer knowledge and implementation is generally retained through all aspects of the thesis.

immersion on RSA extraction and a quantification of user behavior and errors. With VRoot as a tool, researchers can now fine-tune their data extractions for more robustness and precision, particularly with downstream use in FSPM model calibration, which is necessary for the synthetic data employed in this thesis.

In summation, this thesis increases to the state-of-the-art of plant data extraction through the embedding of FSPM model data into a virtual environment. This embedding also extends to a scalable HPC environment, where this embedding was used to generate synthetic data, distribute the virtual scene across nodes, and even couple functional information to the virtual scene. Particularly, the evaluation and analysis of the coupling between CPlantBox and the UE is a novel contribution to the field, as the embedding of plant models into virtual environments is a novel approach to synthetic data generation. Additionally, the evaluation of HMD use in RSA data extraction is novel, particularly considering the rigorous quantitative analysis of the extraction quality across human participants.

2 Background

This thesis involves the generation of model-based synthetic data on modular HPC systems. The central aspect of this thesis is a data generation pipeline that utilizes rendered models to produce data to train DNN models. The individual aspects of this pipeline stem from different disciplines, and their interplay is largely based on the consolidation of technologies and requirements, such as the prominence of DNNs and the subsequent need for scalable computing infrastructures. HPC systems are thus a primary focus of this work, both in recognizing their key role in DNN training and the production of synthetic data. A particular focus is on the use of synthetic data and what specifically a model-based approach can yield in terms of data quality. On the other hand, human-in-the-loop techniques have been employed to improve the initial accuracy of the model calibration through better data extraction, requiring some background in the human-centered side of visualization in addition to synthetic data generation. Therefore, rendering engines, the use of virtual environments and immersive technologies, and the embedding and background of FSPMs are also important aspects of this thesis.

2.1 Machine Learning

Machine learning, and subsequently deep learning, is an ensemble of methods that address uncovering mechanisms, patterns, or features in data. This can be either done from examples (supervised) or from rewards (unsupervised). This thesis focuses on virtual world models to assist deep learning pipelines. Supervised machine learning algorithms are statistical learners that generate a hypothesis on the relationship between data and label sets. Some mechanisms and properties accompany statistical learners and how the data set relates to the label set, particularly regarding the structure of the hypothesis space. For a full description of these topics beyond the scope of this chapter, refer to Shalev-Shwartz and Ben-David [SB14].

A subset of machine learning is deep learning, which takes the central idea of a perceptron learner and connects its instances to a computational directional graph. Perceptron learners are attributed to McCulloch and Pitts [MP43]. DNNs are universal approximators [Rag+17], and the complexity of the space the DNN can approximate drastically increases with the number of layers. The computational graph structure is implicitly inscribed into connected layers that encode weights of the linear sum of the input. This sum is manipulated by an activation function. More background on DNNs can be found in LeCun et al. [LBH15]. A subclass of DNNs are Convolutional Neural Networks (CNNs), which operate using kernels that have a similar structure to fully connected layers that transform a section of the input to a single output node. CNNs

have been used in a variety of applications, including instance segmentation [WMH18], depth estimation [SCN06], and plant phenotyping [Wey+22].

2.1.1 DNN Training

DNNs consist of layers of nodes connected in a loop-free network, where the output of a neuron is a scalar activation function value based on the sum of all connected inputs. Gradient optimization is employed stochastically by optimizers such as the ADAM optimizer by Kingma and Ba [KB17]. Modern DNN training frameworks recognize that optimization on noisy target functions might not be able to reach a minimum loss unless the gradient step is being controlled. Gradients might decay as they are subject to backpropagation, as the weight update is dependent on the derivative of the activation function, and certain derivatives, such as the sigmoid function, will dampen the gradient. To avoid this, linear activation functions have been re-introduced when DNN depth increased, particularly ReLu (rectified linear unit, i.e. linear but nonnegative). ReLu has been introduced to avoid gradient decay in DNNs by Nair and Hinton [NH10]. To facilitate a more direct human expert involvement, feedback loops between the algorithmic annotation and expert annotation can be implemented [Smi+20]. This approach might become necessary in cases of noisy data, as feature extraction methods might be sensitive to artifacts introduced by noise [Hor+21].

A form of training paradigm is also called *active learning*. This paradigm focuses on extracting metrics on the training performance based on the data [Agh+19] and the subsequent choice of data sets and aspects of the data based on these metrics [Cao+20]. Likewise, the notion of *transfer learning* is important, as specialized data sets might be scarser than common ones, making the pre-training on larger but non-specific or unrelated data sets valueable [Che+20], particularly also using synthetic data as in [ACT21].

2.1.2 Computational Demand

DNN training is sensitive to hardware constraints. For a single Graphical Processing Unit (GPU) to handle the backpropagation through a DNN, it must be able to contain the parameters of the network (with the structure implicitly defined by the rule of backpropagation) and the output and minibatch for the training. Modern GPUs, such as those implemented in the LUMI supercomputer, have specialized hardware that can be accessed through generalized programming paradigms [Mar+22]. This can increase the number of parameters that can be stored by reducing the resolution of certain values. However, the training of neural networks remains an optimization task in a very complex space that might have multiple locally optimal points that need to be sampled to find the global optimum. In combination with the increased complexity of problems and the memory-related hardware constraints, most large-scale DNNs, including large language models, are trained using HPC systems.

Large DNN training on HPC systems has been analyzed by Sedona et al. [Sed+19] in the context of remote sensing. Other approaches that use HPC systems for the training of DNNs scale up the potential to utilize these systems by introducing physics-based principles, as implemented by Rodekamp et al. [Rod+22]. A more practical approach

to overcome the problem of parameter tuning, particularly indirect parameters, such as layers, dropout rate, and similar, has been implemented by Aach et al. [Aac+22]. Using reasoning constraints or policy rewards to allow to either overcome sparse data or to steer DNN training has also been used in large language models, particularly those required to operate within strict constraints, such as in Hou et al. [Hou+25].

2.2 High Performance Computing

Computational demand is increasing at a higher speed than the capabilities of individual computing devices. This leads to a greater need for distributed and HPC systems to calculate problems at a larger scale, including forecast models, large-scale simulations, quantum simulators, or distributed deep learning. HPC is the discipline of using large-scale computing machines to solve complex problems. It has not only developed into a distinct discipline in the past decades but has also enabled other disciplines to increase the scale of problems that can be solved and the complexity of the models that can be used.

2.2.1 Parallelization

Parallelization is a key technique in HPC systems. It is defined as the distribution of a problem in the form of sub-problems to individual concurrent computations. Common definitions include task-based parallelism and data parallelism.

Task-based parallelism is the distribution of dynamically generated tasks based on problem size and steering. It is a common approach for visualization tasks, as they are often not only dependent on the data on a global scale but also require the data to be processed in a specific order. An example of this is distributed raytracing, a rendering technique that is not only task-based but can also be implemented to leverage non-uniform memory [Bie+17].

Data-parallel approaches distribute the data to HPC compute nodes. These approaches generally have a method of aggregation, such as the collection of the subsolutions to recombine the global solution or to reduce the subsolutions by operations such as averaging. An example of the reduction of distributed solutions is gradient averaging in distributed deep learning [Sed+19], while visualization paradigms typically depend on a combination of subsolutions, such as raytracing [Lar+15]. This memory distribution might be explicit through communication libraries such as the Message Passing Interface (MPI) or implicit through non-uniform memory access controllers.

Model-parallelism is an approach where the DNN model is distributed across nodes, while the data itself is run concurrently through parts of the model on different nodes [Mwa+22]. This technique is particularly suited for models with extremely high parameter counts.

The network topology also has a great impact on the performance of the distributed system. For modern systems, it would be unfeasible to connect each node to every other node or to connect all nodes to a central one, thus overburdening the controller. One network topology that allows for higher broadcasting efficiency (in scenarios where

the whole structure is used concurrently) is tree-based architectures [Jyo+16], which uses high-throughput connections in levels of connectivity, allowing for logarithmic worst-case communication times.

2.2.2 Domain Decomposition

How a problem is distributed onto the nodes of an HPC system is a matter of domain decomposition. Domain decomposition is historically a term from numerical mathematics, where a mesh or grid is defined over a space, yielding well-defined subproblems that share boundaries. An example of this is numerical solvers for parabolic equations, such as by Acebrón et al. [ARS10]. This is not the same technique as data parallelism, as the domain decomposition for numerical problems is a technique to compute a discretized version of a problem to approximate its continuous solution.

However, visualization and data analysis also have practical and immediate application cases for domain decomposition. Identifying subproblems that can be decoupled is a cornerstone of distributed visualization [Lar+15], and the inclusion of visualization paradigms in large-scale HPC-enabled simulations is one of the key aspects in how large data sets can be analyzed [Bau+16]. Indeed, in-situ visualization is a fully separate field of research, and its primary concepts and terminology have been detailed by Childs et al. [Chi+20].

2.2.3 Modular Supercomputing

While the unique selling point of HPC has historically been uniformity and parallel performance, newer contributions to the field have also highlighted the concept of *modular supercomputing* [SEL19; Lip21]. A modular supercomputer is not only the reaction of the infrastructure to a diversifying demand for computational power, specialized across methods and domains, but also the recognition that specialized hardware will ultimately outperform classical computing approaches when the scale is increased. Many examples of this can be found, most notably the computation of DNNs, which scales well across GPUs [Sed+19] due to their highly parallelized nature. The integration of modules into an existing HPC system also bears potential for new technologies to have a quicker adoption among scientists, such as quantum computing resource [Bec+24].

Common modular aspects of a supercomputer are a general purpose cluster for Central Processing Unit (CPU)-focused workflows, a GPU cluster for highly parallel tasks, and visualization or head nodes that are interactive or used for the orchestration of the system. Modular supercomputers might have dedicated nodes for large-memory computations, such as implemented in JURECA [Thö21]. Within this thesis, we make heavy use of GPU nodes, but one of the base requirements is graphics support. GPUs with graphics support can handle workloads that utilize graphics libraries such as Vulkan[5]. A recent survey of autonomous driving simulators also showcases that graphics capabilities on dedicated and exclusive nodes are an important requirement of certain CNN training pipelines [Sil+24]. However, certain implementations such as LUMI [Mar+22] have specialized GPU nodes that are aimed at highly parallelized

[5] https://www.vulkan.org/

workloads and are not compatible with graphics libraries.

2.3 Visualization

A core aspect of this thesis is the use of visualization in both data analysis and data augmentation. Visualization techniques generally rely on rendering techniques for display and data production. Rendering is the technique that generates an image from a virtual scene. Basic aspects of the rendering include a scene camera with a view frustum and a specific set of parameters that define how the pixels map to the scene and how points in 3D space are projected onto the 2D image. Historically [Fol95; Den90], rasterization techniques have been used, as they only rely on the geometry shaders in the GPU, converting a 3D scene into an image. The image is then post-processed by pixel shaders that can be used to apply screen-space effects such as reflections. Rasterization techniques are still used today, particularly in Virtual Reality (VR) rendering techniques, as latencies need to be minimized [FRS19]. A cornerstone of modern rendering is physics-based rendering, which is a technique that models surfaces in a virtual scene so that their reaction to light influx is realistic compared to real-world surfaces [PH10].

Game engines are software products that contain implementations of common rendering techniques and are particularly suited for creating virtual worlds. Andrade [And15] studied the use of game engines, particularly focusing on serious games. An example of a game engine, particularly relevant for this thesis, is the Unreal Engine (UE). UE includes different rendering techniques, a physics engine, hardware abstraction, interaction device support, and a range of plugins for more specific use cases, including the PixelStreaming plugin described in Sec. 4. The UE, like other modern game engines, employs raytracing techniques.

Raytracing is a rendering technique that actively samples the pixel value for each pixel individually by tracing rays from the view frustum into the virtual scene, thus yielding the color of the pixel depending on geometries, light, and surface properties. Raytracing has been considered until recently to be a too high compute workload to achieve in real-time, i.e., within milliseconds of a change occurring in the virtual scene [Den+17]. However, within the context of synthetic data generation, as employed in this thesis, both the use of HPC systems and the decoupling of the rendering from the training process enables the use of raytracing techniques without heuristics such as Lumen[6].

2.3.1 Human Vision and Visualization

Visualization will have two concurrent meanings in this thesis: Visualization for humans and for synthetic data. Historically, visualization methods encompassed methods to analyze and present data in a way that supports individual understanding of the data or decision-making. The potential complexity of certain data sets also means that data analysis, dimensionality reduction, or even computational methods for information

[6] Lumen Documentation dev.epicgames.com/documentation/en-us/unreal-engine/lumen-technical-details-in-unreal-engine

extraction, are employed to assist with this [Ise+17].

Within the context of this thesis, one system that combines visualization with modeling and data analysis is a VR system. VR is used for many different data analysis tasks in different domains, as researched by Fonnet and Prié [FP21]. Notably, many of the use cases they found were targeted at exploration rather than manipulation or extraction. VR headsets, so-called Head-Mounted Displays (HMDs), are low-persistence global displays. Examples for VR headsets today are the HTC Vive Pro[7] and the Meta Quest[8]. These HMDs are consumer-grade hardware units that allow broader access to immersive technologies. Many highly-detailed VR approaches historically have been room-scale installations, such as the aixCAVE at RWTH Aachen University [KM22]. There are advantages and disadvantages to these systems, though HMDs generally have a lower barrier of entrance, as there is no need for any special installation.

Visualization, particularly data visualization, is a discipline that combines human perception, computer graphics, and human-computer interaction. While there is much literature on the topic, cf. Bowman et al. [Bow+17], our focus is on important concepts to understand the VR technology utilized in this thesis. 3D visualizations of virtual environments are created to support use cases in which 3D rendering is the most natural data embedding. The discipline of using 3D visualizations to represent data for analysis is called Immersive Analytics (IA), and a study on this topic and its use in science has been published by Fonnet and Prié [FP21]. The software designed for IA typically involves different metaphors for interacting with data [Bow+17]. These metaphors are interaction tools designed to make the data exploration or manipulation more intuitive, particularly if the standard motions, such as swiping for panning or pinching for zooming, that are a direct translation of hand movements are not feasible.

Generally, software designed for immersive visualization should avoid and actively mitigate the risk of cybersickness [LaV00], which is a physical sensation of unwellness after using VR devices. Numerous factors contribute to cybersickness, and the pathology is different across devices, such as investigated by Stanney et al. [SKD97], who highlight that simulator sickness and what is commonly referred to as cybersickness are two distinct effects. Current best practices will usually refer to a low render latency and high reactivity while also excluding certain navigation techniques that could contribute to cybersickness.

2.3.2 Machine Vision and Visualization

With the development of CNNs, the prevalence of *computers that have vision* increased dramatically. As such, vision is no longer a uniquely human capability, which yielded a lot of promising technical advances and a need for labeled images that computers can train on. The complexity of measurement setups and other aspects of the labeling make this process the most labor-intensive. Even public data sets, such as ImageNet [Den+09], suffer from labeling errors and are prone to mistakes that might introduce unforeseen consequences for models [Nor+21].

Synthetic data is a model of the input space, which is generated from individual models or an approximation of the input distribution. The application cases for synthetic

[7] HTC Vive Pro Product Specifications: www.vive.com/us/product/vive-pro2/specs/

[8] Meta Quest 3 Product Specifications: www.meta.com/is/quest/quest-3/

data are those in which data pipelines are difficult to set up, either due to labor intensity, as for creating data sets of leaf segmentation [WMH18], or due to the required detail [McC+17]. Synthetic data requires the deterministic production of a label, meaning that a data space model should represent the distribution of the input space along with a generator that represents the desired labels. Synthetic data not only mimics the input space, such as plant images, for the benefit of the learning algorithm. Instead, synthetic data must also be diverse (and thus large) enough to cover as much of the possible label space as possible, resulting in robust data production for DNN training. How well a synthetic data set represents the true distribution depends on its variety and the correctness of the emergent property model, but not necessarily on realism, even though that may be a factor.

The *realism* of a data pipeline is very challenging to quantify. One approach, also used by Zhang et al. [Zha+20], is to use the trained model on a real-world labeled data set and subsequently analyze the successful training of the model on synthetic data. While this approach has merit, it does not make any statement on the realism of data. There are approaches, particularly in algorithm verification and evaluation, that use synthetic data to quantify error rates but purposefully sacrifice realism because it does not increase similarity to the base data in relevant statistics, as employed by Schnorr et al. [Sch+20] for time-variant analysis problems. Another approach to quantify the similarity of the training data compared to experiments is to extract derived measures or features that can be compared between the two data sets. This is an essentially model-free approach, as it does not depend on the actual performance outcome of the model but on the similarity of the data sets. To generate synthetic data, however, realism is secondary to coverage of the input space, which is a more crucial aspect, as it provides more full information on the algorithm performance [McC+17; Zha+20; Sch+20].

Dataset diversity is generally increased through augmentation methods. Augmentation methods typically transform the data point synchronously with its label or without changing the label. Examples of this are mirroring, turning, and even elastic deformations [CCP18]. Using synthetic data, domain augmentation can be performed, which means that the scene itself and the conditions in which the images are captured are changed, such as implemented by Qiu et al. [Qiu+17], Zhang et al. [Zha+20], or Zhang et al. [Zha+18]. In certain cases, domain augmentation simply means to arrange, deform, or otherwise change the scene in a way that influences geometry positions, geometry shapes, or lighting conditions. More domain-specific cases could generate data, such as seen in Ward et al. [WMH18] using fewer or more leaves, hidden leaves, or even no leaves at all.

2.4 Functional-Structural Plant Models

Understanding processes in the plant, the contribution of its organs, and the genotype of the plant is key in evolving our crop to accommodate both climate change and a growing world population. However, measurements that yield actionable information are scarce and not necessarily based on above-ground information that can be sampled easily.

A plant model is a phenotypic description of a plant, meaning it encompasses simulated information about plant traits and plant function [Vos+09]. Classically, structural plant models are Lindenmayer systems [CK09], describing the growth and spatial distribution of plant organs. An example of a functional model is a rhizosphere uptake model: Mai et al. [Mai+19] describe a multi-scale model to solve the water uptake equations, making explicit use of the FSPM structure. This is informed by a key aspect of structural models, which will generate a graph structure G_1, which is a set of vertices and edges $G_1 := (V, E)$ together with properties assigned to either.

The baseline for FSPMs is the calibration of the model to experimental data. This becomes self-evident once the number of individual parameters (> 50) in models such as CPlantBox [Sch+18a; Zho+20; Gir+23] is considered. The stochastic expressiveness of the model [Mor+21b] has been evaluated, but many experimental data acquisitions cannot be immediately mapped onto parameters, requiring stochastic optimization [Mil+19]. Modeling platforms such as Helios [Bai19] have even more burden of calibration, as the baseline model is a framework rather than explicitly encoding plant structures or experimental conditions. This is a one-domain approach that is also used in other models, such as by Paulus [Pau19]. For individual experimental conditions, there might be multiple distinct solutions to the reverse problem. This issue can be mitigated by introducing more auxiliary measurements that might lead to the correct result.

Another example of models that combine functional and structural information are photosynthesis models. These models simulate the biochemical reaction pathways in the canopy of the plants as a result of the plant's interaction with the surrounding air and the light influx. These models are not only used to model the reaction pathways in the plant [Leu95] but also assist in uncovering mechanisms across varieties of plants [RM20] and to extract additional information from measurements [Jun+21].

FSPMs are uniquely able to describe experiments [Bau+23] and further enhance the experimental data by allowing full-scale access to the structural properties that have most likely developed within the field or laboratory experiment. The most important added value for these models is to more fully inform the possible variance that leads to similar experimental measurements.

3 Related Work

This thesis is situated at the intersection of several fields, particularly visualization, plant modeling, and high-performance computing. Consequently, previous publications and related work often touch upon certain aspects of this work, but a larger survey is necessary to shed light on all aspects. This chapter goes into the details of each topic, particularly their relevance to this thesis.

3.1 Computer Vision

Minervini et al. [MST15] showed that image analysis, mainly through computer vision, is the primary bottleneck of plant data analysis. This is confirmed by the persistence of image-based methods in plant phenotyping. Computer vision problems are diverse, as are the tools researchers use to meet them, and this section focuses on the use of DNNs to extract important markers from images that can be used in plant modeling. LeCun et al. [LeC+89] introduce the concept of CNNs to extract text from handwriting samples. Since then, the employment of CNNs has had a great impact on plant data analysis. Pound et al. [Pou+17a] and Gao et al. [Gao+20] report a significant increase in the performance of plant data extraction when CNN techniques are employed. However, CNNs require large training data sets and computational resources [SS17; KP18]. Efforts to address this are three-fold: Firstly, the public availability of data sets increases, with research practices adapting to accommodate a better common knowledge and data basis, with high-quality data sets such as by Pound et al. [Pou+17b]. Secondly, Ronneberger et al. [RFB15] altered the CNN architecture by introducing layers that learn to apply filters, reducing the amount of data required for a satisfactory training result. Thirdly as Sedona et al. [Sed+19] presented, HPC systems are a key technology to train DNNs to the level of robustness and generalizability required for remote sensing problems. These approaches particularly make use of modular supercomputing architectures, as they accommodate special use cases through dedicated hardware, as implemented in JURECA-DC [Thö21], JUWELS [Kes+21], LUMI [Mar+22], and in the future in JUPITER [Dum23].

3.2 Synthetic Data and Distributed Visualization

Synthetic data, especially in digital twin modeling, has seen recent advances as the data requirements for DNN training increased. One of the prominent building blocks of synthetic data is *domain-based augmentation*, which is the augmentation of the

virtual scene through privileged information. This thesis focuses on synthetic data as produced by modeling, as opposed to a direct approximation of the data space through generative networks, such as presented in Yates et al. [Yat+22]. One of the more influential works in the realm of synthetic data generation using virtual scenes was written by Qiu et al. [Qiu+17], who developed a Python-based coupling to the UE that works similarly to OpenCV [Pul+12]. Works such as by Mania and Beetz [MB19] build on this by expanding the framework for more specific purposes, such as robotic agent environments. From the perspective of visualization that needs to achieve a high level of realism, the use of game engines such as the UE is common. Korkut and Surer [KS23] researched the use of game engines in science, particularly for visualization and data analysis tasks performed by human users, showing a broad adoption of game engines in the scientific community. However, they also point out that while very convenient, game engines might perform optimizations that make them unreliable visualizers. This concern is mostly aimed at the visual-focused nature of game engines. However, the choice of using a game engine rather than a conventional visualization tool is mostly a matter of both convenience and efficiency. Huo et al. [Huo+21] as well as UnrealCV [Qiu+17] show that the employment of a game engine-based rendering pipeline also provides auxilary functions that directly allow label data extraction. Fast prototyping is another matter, with the engine allowing the user to arrive at an acceptable image quality rather quickly. Casao et al. [Cas+23] highlight such use-cases, and though they do not provide quantification on how fast the generation was achieved, they do provide established support for this point. A data-centric approach to close the gap between synthetic and real-world data has been used by Zhang et al. [Zha+20], who implemented a pipeline that uses downstream transfer learning [ACT21]. Indeed, synthetic data as a precursor for real-world data is commonly employed if there is a gap regarding structure or features within the data, as described by Achicanoy et al. [ACT21] and Rajpura et al. [Raj+18].

A more in-depth survey of the model visualization for geospatial data has been published by Mat et al. [Mat+14]. The use of game engines, particularly in the simulation of virtual worlds, has been explored by Chance et al. [Cha+22], who studied frameworks that can assist in the coupling of autonomous agents to virtual worlds in the context of self-driving cars. Bondi et al. [Bon+18] employed a UE pipeline in the generation of a virtual world assisting unmanned aerial vehicle agents to assess wildlife presence. The strengths of using UE specifically are highlighted by Zhang et al. [Zha+20], who employ domain-based augmentation, i.e., changing of clothes on human avatars, to enhance data production capabilities of the pipeline. Furthermore, UE provides auxiliary data usable in synthetic data pipelines, such as depth information, a central component of the geometry buffer that is used for screen-space effects in post-processing shaders [Zha+18; McC+17].

However, there is still a gap between what synthetic data can achieve and real-world data sets, even with large-scale application cases, such as by Mu et al. [Mu+20] or Roberts et al. [Rob+17]. The primary challenge is that plant models carry biological meaning, and most phenotyping algorithms that need to infer that biological meaning [Bai19; Lei+24] still depend on extensive measurement setups, even modern methods as developed by Berrigan et al. [Ber+24]. Within plant science, synthetic data can be a powerful tool to overcome the burden of annotation. Implementations such as by

Ward et al. [WMH18] already incorporate some level of domain-based augmentation, namely using a labeled sub-picture to generate a composition image that will yield different properties and can extend the data set size of the training data. The strength of simulating virtual worlds for plant science has also been highlighted by Lei et al. [Lei+24], who implemented a framework for the physics-based simulation of light propagation in the context of plant science. Model-based data generation, which is particularly relevant for plant phenotyping, was implemented by Lobet et al. [Lob+17], who uses CPlantBox, an FSPM that is well known in its stochastic properties [Sch+18a; Mor+21b]. This thesis aims to establish that the use of the UE is a scalable (TO**3+4**) method of generating synthetic data with domain-based augmentation (TO**1**) and that the model realism and biological information can be retained through the use of FSPM models (TO**4**).

Distributed visualization paradigms are common in cases of large data sets that do not fit into GPU memory, such as implemented in Moreland et al. [Mor+16], Aumüller [Aum15], and Larsen et al. [Lar+15]. Setups such as the aixCAVE in Aachen use distributed rendering to allow for each node to render a part of the scene, as detailed by Kuhlen and Marthys [KM22]. Biedert et al. [Bie+17] have implemented a scalable approach for task-based parallel visualization, agnostic towards the memory model that is implemented on the hardware. An example of a distributed rendering model that directly aims at distributing virtual scene rendering is implemented by Liu et al. [Liu+22]. A contribution from this thesis also delves into the use of load balancing techniques specifically for cases in which the task, but not the data, is being transferred between nodes (**C-B**), as implemented in Demiralp et al. [Dem+22]. To our knowledge, distributed rendering techniques for our specific use case - cases in which actor movement on a single node is limited due to the overhead required in unloading and loading scene geometry - have not been established yet before this research. As such, an important aspect of this thesis is the implementation of distributed paradigms for virtual world simulators (TO**3**).

3.3 Functional-Structural Plant Models

FSPMs are models that generate a digitized plant. An early employment of this was through using Lindenmayer [Cie+21] systems, developed by Ciosek and Kotowski [CK09]. This is also true for CPlantBox, the FSPM utilized in this thesis, as its inaugural implementation, CRootBox, developed by Schnepf et al. [Sch+18b], was primarily such a system. Comprehensive surveys on the state and history of FSPMs have been performed multiple times, e.g. by Vos et al. [Vos+09], Soualiou et al. [Sou+21], and Louarn and Song [LS20]. The significance of FSPMs in uncovering new mechanisms was highlighted in a study by De Bauw et al. [De +20] for rice crops. Recent results confirm this, including a contribution from this thesis, showcasing the use of modeling in uncovering otherwise hidden mechanisms and functional properties [Bau+24] (**C-D**).

To extract the plant-light interaction, full geometric modeling needs to be employed to accurately quantify surface radiation exposure. Models for this simulation are commonly called radiative transfer models, as the properties of the light are being simulated

physically, including the reaction with surfaces. DART, developed by J. P. Gastellu-Etchegorry and Gascon [JG04], is an example of such a model based on successive simulation of photon paths through raytracing. Models that compute radiative transfer have a direct use in estimating functional properties from plants, such as chlorophyll content, as implemented by Zhao et al. [Zha+24]. Of particular note are computer graphics models that use basic radiative transfer concepts, which have been standardized by Martonchik et al. [MBS00]. The physics-based rendering and surface scattering is an important aspect of the realism achieved by UE, which depends on a model by Jimenez et al. [Jim+15] for the computation of a surface scattering component that is compatible with the deferred rendering and material shading pipeline. An analysis of the similarities between radiative transfer models and physics-based rendering models has been performed by Salesin et al. [Sal+24]. The need for a full geometric model of the plants has been highlighted by Xiao et al. [Xia+23], and while it has been implemented in models such as Lecarpentier et al. [Lec+19], the question on the 3D-functional embedding of specific models such as CPlantBox [Gir+23] remains an open question (TO**4**).

3.4 Data Analytics

Data processing is ideally fully automated, as it requires the least workload. However, there are always cases in which automated tools do not perform, and the reasons for this are diverse. Horn et al. [Hor+21] uncovered cases in which expert annotation differed from the automated approach, but reviewing cases of discrepancies, the individual decision made by an expert in this context would also be difficult to replicate for an untrained human annotator. Many tools to handle noise or low contrast issues in plant imaging data are either semi-automated or fully manual. Regarding images, Smith et al. [Smi+20] developed a tool to assist a learning framework by providing annotations that correct edge cases, and Bauer et al. [Bau+22] show that this annotation can perform for larger experimental setups. Respectively, tools such as NMRooting, a 3D extraction tool by Pflugfelder et al. [Pfl+17], try to accommodate a mixture of human and automatic annotation.

Extraction tasks are often the extraction of sparse information from dense data, as implemented by Usher et al. [Ush+18]. In this paper, the authors developed an expert annotation tool that they could extend using data analysis functionality [McD+20]. One finding is that experts are better able to annotate data sets for further analysis on immersive instead of conventional software. This comparison between desktop and immersive software is uncommon, yet the prevalence of these studies increases the impact of new technologies on domains that underutilize them. In plant data analysis, 2D methods for feature extraction are well explored, but particularly technologies such as HMDs require formal verification to assess whether they outperform similar annotation methods on desktop systems (TO**2**).

4 Synavis: Virtual Environments for Plant Models

The scientific workflows of the publications following this chapter depend on a technical infrastructure and software stack that requires a dedicated chapter to be described fully. UE is a graphics engine that has historically been used for game development but has increasingly been applied to scientific and technical applications. The visualization of plant data in this thesis permeates all publications, though it is of particular interest within the Synavis framework.

From a technical perspective, the choice of visualization back-end is centered around three main concepts: First, the visualization should be adaptable to a broader range of hardware and also make infrastructural support feasible to allow for a wider range of use-cases. Second, while realism is a secondary goal of data production, with the primary goal being coverage of the input space, it is a valuable starting position to have industry support that is primarily concerned with photorealism [Sil+24]. Third, a user-facing perspective regarding the data generation aspect needs to be considered: There should be an efficient pathway towards common label data sets, including the potential for fast prototyping. Preliminary analysis of rendering tools that was conducted for this thesis yielded two main factors that informed the choice of using a game engine such as UE over other tools: UE is open-source[9] and adaptable, which was used in the Synavis framework to some degree, as it is implemented such that it can be used with the standard UE implementation, but change it to such an extend that it is possible to run the engine in a data-generation mode. UE has furthermore been shown to be useful for synthetic data, as its render pipeline is permissible and offers its transient information up for further processing, a fact that was used by Mousavi et al. [MKE20] to establish a training framework.

The implementation of the Synavis framework is three-fold. A WebRTC [JBB21] bridge and communication framework, called Synavis, manages the communication between the endpoint application and the visualization backend. This backend is the SynavisUE plugin running in UE. The third component is the plant visualizer implemented in CPlantBox, which carries the embedding and is designed to work with Synavis to facilitate UE visualization of plant data. Synavis and SynavisUE are available as open-source software[10], and the plant visualizer has been developed for this thesis as part of the CPlantBox software package[11].

[9] Registration required

[10] Synavis: github.com/dhelmrich/Synavis

[11] CPlantBox: github.com/Plant-Root-Soil-Interactions-Modelling/CPlantBox

4.1 Model Coupling to Unreal Engine

CPlantBox produces a graph structure of a plant, as described in Schnepf et al. [Sch+18b] and recently in Giraud et al. [Gir+23]. This structure can be encoded in a structural language, described in Lobet et al. [Lob+15]. The structure, which we call G_1, consists of a graph (V,E,P) that encodes vertices, edges, and properties, respectively. In the CPlantBox visualizer and VRoot, a high-precision data extraction tool, the geometrization scheme yields an outer geometry G_3 that uses 2D shape templates and connects them using triangles. This method is a simplistic method of describing the outer geometry, but it retains access to the model by allowing relative texture coordinates to be referenced, scaling with the organ size.

This structure, G_3, is used in SynavisUE to visualize plant data. It is important to state that while the one-way visualization is an important factor, the actual coupling is synchronous, meaning that the FSPM simulation is run concurrently with the engine. This is done partially to allow for a dynamic scene generation, as highlighted in **Baker** et al. [Bak+23] and later in **Baker** et al. [Bak+25c] (Sec.5), but also to be able to change the plant model concurrently with the virtual scene. However, it should be noted that the inclusion of FSPMs depends on their calibration, and while tools such as VRoot [Bak+25b] alleviate some concern regarding the accuracy of model calibration, it is still a task that ultimately requires domain knowledge.

Within UE, the visualization utilizes procedural geometries. A procedural geometry in UE refers to a set of points, edges, and triangles that are inserted into the virtual environment at runtime. This reduces the complexity of the application within UE and allows for the runtime creation of the virtual environment. The geometry is rendered in UE by raytracing algorithms. Rasterization algorithms are available for geometric visualization, which can be used to visualize data on devices such as Oculus Quest 3 as implemented in Baker et al. [Bak+24] as part of this research (**C-E**). However, for a complete and faithful view of the scene, raytracing, primarily path tracing, is employed. Raytracing is a method of producing an image from a virtual scene by using measurement rays, yielding the final color of a pixel. The algorithm uses "reverse" light exposure, meaning that the camera shoots measurement rays into the scene that get reflected or diffused on surfaces. Fig. 4.1 illustrates the individual paths the rays take - in this instance, it is also of particular note that the diffuse measurement can be replaced by a heuristic or a surface cache. In our use case, as the data is procedurally introduced into the scene, a fully faithful visualization is only possible through path tracing. Path tracing is a UE-term referring to this measurement without scene simplification or heuristics. Path tracing will be randomly reflected to sample the hemispheric diffuse light influx onto a surface, adding a measurement with subsequent path traces, yielding a final image after a certain number of samples.

UE uses physics-based rendering. Surfaces are encoded as *materials*, which means that there is a graph-based shader description, which can be translated to high-level shader language code. This graph description encodes all physical properties needed to establish the bi-directional radiative transfer, which includes texture information and surface properties. These properties are illustrated by individual modification in Fig. 4.1 at the bottom. For the full description of how this is implemented, we refer to the official

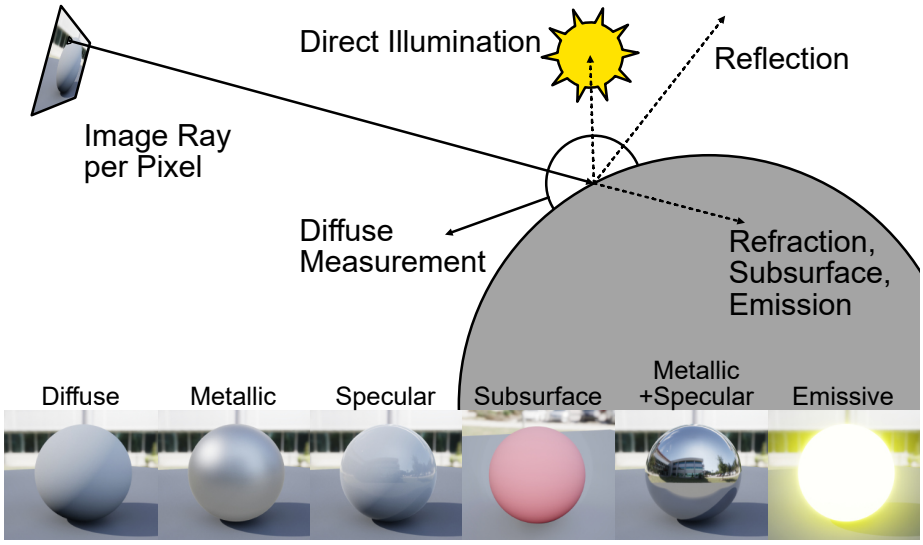

Figure 4.1. Visual description of the raytracing algorithm. Below are samples of materials that exhibit different properties as listed above.

documentation of UE[12]. A full clarification of the properties shown: Diffuse refers to a high roughness value, with the base color being grey. Metallic refers to how much the original color of the light is retained when reflected off the surface. Specular refers to the level of direct reflection. Subsurface scattering is implemented like in Jimenez et al. [Jim+15]. Lastly, emissive is an added contribution of the surface brightness value to the scene.

4.2 Synavis: WebRTC Communication

WebRTC [JBB21] is a communication protocol that aims to provide a standardized way of communicating using media streams. These media streams, such as video or audio packaged via Real-Time Protocol (RTP) [Sch+03], are accompanied by a data channel, which hosts messages and other information on a User Datagram Protocol (UDP) connection. There are numerous examples of WebRTC in industry, but for the setup used in this work, we focus on the implementation within UE and subsequently within Synavis. All control messages, either through the data channel or through the Hypertext Transfer Protocol (HTTP) signaling server, are written in JavaScript Object Notation (JSON) format, further described in Bourhis et al. [Bou+17]. JSON format is a structural format that uses key-value pairs: `json{"key":"value", "key2": {"nested-key": "nested-value"}}`.

PixelStreaming is a UE plugin, turning the visualization into a WebRTC endpoint

[12]Unreal Engine Documentation: dev.epicgames.com/documentation/en-us/unreal-engine/

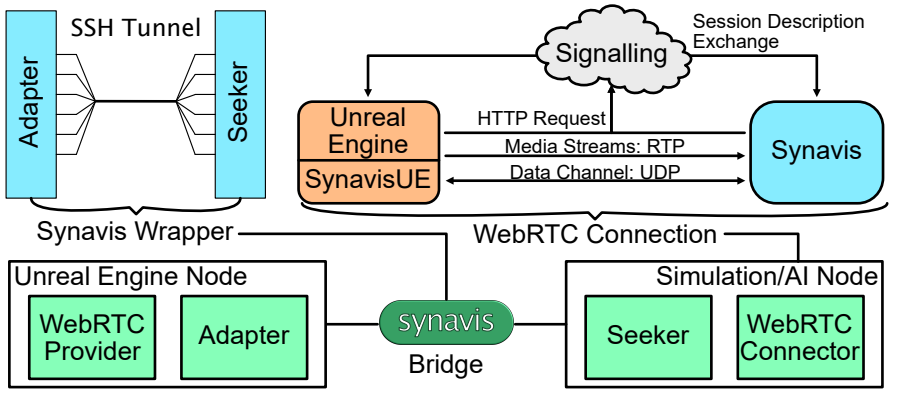

Figure 4.2. Schematic of communication components when using Synavis via WebRTC. The Synavis concept is centered around mimicking a local WebRTC connection and using the wrapper within Synavis to tunnel the communication via SSH connection. The signaling server is an HTTP-Server that is used to establish the connection between the two peers, allowing them to identify each other via public identity. Media streams are transmitted via RTP and data channel communication is UDP-based.

by encoding the final image, otherwise being displayed on a screen, into a video stream. This stream can be output to any WebRTC connection, and the primary use case for this is remote visualization use cases. The central idea of Synavis is that the plugin and the WebRTC implementation do not require an active user, nor do they require manual interaction. Synavis defines two automated WebRTC endpoints: a streamer and a receiver. The receiver is the application that uses the media streams and trains the DNN, while the streamer is the virtual environment producing the media streams. Another type of receiver is a simulator, which is a WebRTC endpoint that serves to provide geometries and scene information, such as weather conditions, to the streamer. These types of receivers will use the WebRTC standard but only to establish the setup of the data channel.

WebRTC is a framework that establishes communication by hole-punching, which means that the public identity of two peers is communicated through a signaling server, a separate entity that transmits information about potential direct connections. This direct communication, as it depends on a dynamic port range, is not always possible when spanning modules or different HPC systems. Fig. 4.2 illustrates this setup. The central communication host is the Synavis framework at the bottom, acting as a bridge by collapsing individual ports to a Secure Shell (SSH) tunnel. This is indicated with the bridge and the Synavis wrapper in Fig. 4.2. Synavis also handles the direct WebRTC connection, making it a standalone feature of the framework. If the bridge functionality is used, there are two separate WebRTC connections managed by Synavis, which are synchronized through the bridge. For the connector side, i.e., the simulator or DNN training, this is showcased in Fig. 4.2 by illustrating a WebRTC setup. The startup of the streaming connection is done in three steps: The signaling server is an HTTP server that will communicate and relay communication candidates via the session

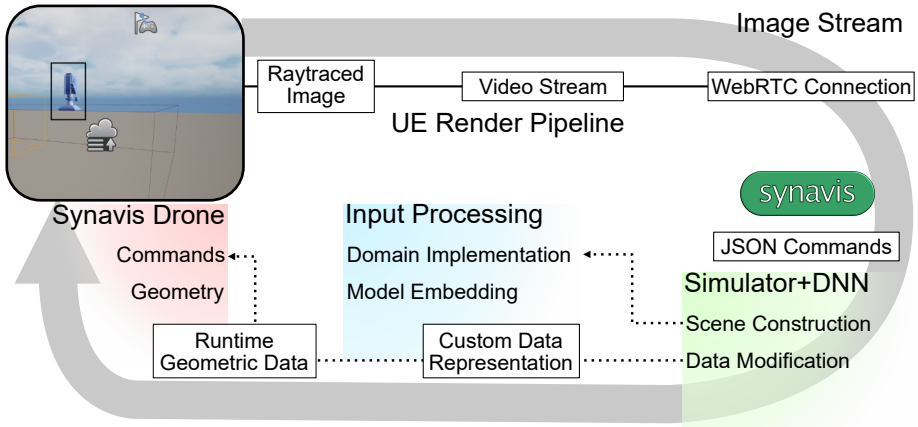

Figure 4.3. Data flow when using the SynavisUE plugin, in direct interaction with UE. Clockwise, the components are the Synavis drone generating the data, the UE render pipeline, the PixelStreaming plugin that extends the pipeline by an encoder and hosts the WebRTC connection, the Synavis framework that bridges and defines JSON commands, the connector running the simulator or DNN and controls the scene, the input processing containing user implementations and custom geometrization schemes, and the SynavisUE plugin taking commands for scene control.

description protocol. When this communication has been finalized, the media streams are established as RTP [Sch+03], and the data channel is opened. Our implementation primarily orients itself around the standard WebRTC description, seen in Jennings et al. [JBB21], while it uses a third-party implementation of the protocol, which is libdatachannel[13].

4.3 SynavisUE: Dynamic Virtual Environments

The SynavisUE plugin is a WebRTC virtual world steering plugin for UE. Its primary feature is the full control it allows over the virtual world. This is done in three ways:

1. SynavisUE exposes the entity-component system of UE in a JSON format.
2. SynavisUE contains methods to steer the virtual scene and embed geometries, which can be extended using the InputProcessing class to implement methods for a specific use case.
3. Within SynavisUE, the generation of labels and the streaming of videos can be controlled to allow for an expedient setup of the training process.

SynavisUE generates uniquely named entities, which it stores in a runtime database that can be accessed and polled. The methods to manipulate the scene are either based

[13] libdatachannel: github.com/paullouisageneau/libdatachannel

on entities or functions that directly manipulate these entities, such as changing the transformation of an object. Fig. 4.3 shows the Synavis drone, which is a central class that not only parses commands but also uses virtual cameras to generate data streams. Sparse data can also be accessed through commands, resulting in complete control over the data generation aspect of the framework. Once the Synavis drone has been placed within the editor of UE, it will automatically take control of the virtual environment and is steerable through WebRTC.

Fig. 4.3 illustrates the data flow when using the SynavisUE plugin. The Synavis drone orchestrates what data is rendered into an image to generate synthetic data. From the UE scene, as seen in Fig. 4.3 on the top, the rendered image is further processed into a video stream using encoding. This is output to PixelStreaming, which hosts the WebRTC implementation in UE. On top of this implementation, the SynavisUE plugin also uses a predefined set of JSON commands to communicate with a Synavis endpoint. On this endpoint, a simulator or DNN training framework is running, which will receive the streams and can communicate using the data channel. An example of such a communication is the polling of objects in the scene, `{"type":"query"}`, which will return a list of objects in the scene, `{"type":"query","data": ["SynavisDrone_C_0", "SkyLight_0", "Plane_0", ...]}`.

Extending the list of standard commands can be done using custom input processing methods, which can contain domain information, such as the ability to create virtual fields using simplified commands. This thesis showcases an example of this, as the implementation of the virtual crop field in Baker et al. [Bak+25c] is implemented on top of SynavisUE. For the data's transmission, the connector should break simulation data down to geometry and texture levels that the Synavis drone can directly process and place into the scene. The code from above directly showcases that UE internally uses an entity-component system, which SynavisUE exploits to make UE remotely steerable.

Downstream tasks that are of interest to DNN training are varied. With Synavis, both image-based and measurement-based tasks can be used. Particularly for image-based tasks, the transient properties of the rendering engine are particularly useful [MKE20; Tre+18]. However, Synavis also enables the generation of aggregate data for the training of DNNs. For example, counting and tracking tasks can be used to extract exact in-silico measurements on the amounts of entities (such as plant organs) or the position of objects (such as pedestrians). Line traces can be created, such as used in virtual distance sensors, to simulate light detection and ranging systems. Additionally, as highlighted in **Baker** et al. [Bak+23], and also by other works such as Zhang et al. [Zha+20], the implementation of scene variation changes the data in such a way that it influences central metrics, such as thresholds, and DNN approach performance.

5 Summary of Publications

5.1 Article I: Synavis

D. N. **Baker**, F. M. Bauer, M. Giraud, A. Schnepf, J. H. Göbbert, H. Scharr, E. Þ. Hvannberg, and M. Riedel. "A scalable pipeline to create synthetic datasets from functional–structural plant models for deep learning." In: *in silico Plants* 6.1 (Dec. 2023), diad022. ISSN: 2517-5025 DOI: 10.1093/insilicoplants/diad022.

This paper addresses TO**1** of the thesis objectives, focusing on the use of model-based world generation to support and augment DNN training. The primary technical focus of this paper is the enabling and design of a pipeline that connects plant modeling and deep learning and respective complex technical components, such as FSPMs and CNNs. It also describes how we embed CPlantBox into UE, with relevance to all thesis objectives. The video description of this paper can be found at:

D. N. Baker. *Supplemental Video for our Manuscript ISPLANTS-2023-026.R1*. FigShare. 2023 DOI: 10.6084/m9.figshare.24179136.

5.1.1 Summary

The article describes in detail the virtual world embedding of a plant model to enable the generation of image data that CNNs can use. Conceptually, CNN training uses emergent properties of image data to connect the annotated data labels to features in the image. These features can be used to infer labels from images, ideally including unknown or atypical images. The key challenge for synthetic data is to generate features in images that are emerging from plants in the scene, without prescribing model on what these features look like.

Synthetic data extends the data augmentation domain to use case-specific domain augmentation, a promising way of combating data scarcity for CNN image analysis. We employ UE to achieve a base level of realism for our model data. For this purpose, we have developed Synavis, a coupling mechanism that uses standardized protocols and allows the coupling of different software to UE (or any WebRTC [JBB21] compatible endpoint) within one ecosystem-like coupling. The coupling is furthermore explicitly implemented for HPC use cases, allowing for the tunneling of communication and the localized coupling of streaming peers.

The article highlights the scalable coupling method (TO**1+3**) and aims at establishing a baseline and framework for the setup of synthetic data pipelines for (TO**1+4**). Consequently, this article provides the primary description of how the plants are implemented in terms of their virtual embedding. The geometrization is a critical factor, but it also introduces a lot of additional parameters that either rely on standard assumptions,

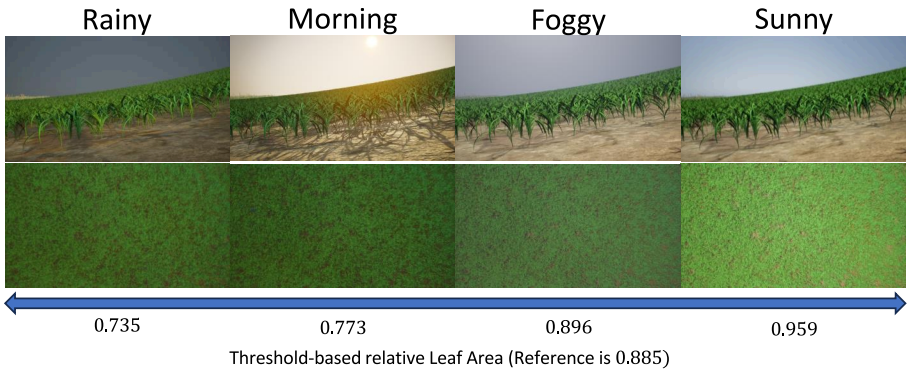

Figure 5.1. Model-free comparison of the same field image, showcasing a measure for data diversity that can be utilized to control the generation. Adapted from Baker et al. [Bak+23].

i.e., the leaf blade being perpendicular to the branching vector, or are directly derived from the experimental data. Our approach uses an extrusion-based geometrization, shaping a 3D realization G_3 from the node structure G_1, which assigns each organ of the plant a base outline connected by triangles. Examples of additional parameters introduced are stem segment orientation, radius, leaf blade orientation, and thus potential sun exposure, and more. This iteration of the pipeline mostly focuses on scalable data generation, highlighting aspects of the virtual environment that can be randomized and how data set diversity can be increased. An example of this is weather, shown in Fig. 5.1, which can be dynamically changed within UE, yielding different image properties. To highlight the change, the histogram threshold-based leaf area index is shown below each configuration, highlighting how domain-based adaptation can introduce dataset diversity. This is a model-free comparison, which is a method to measure features in the data that can be directly derived and are not dependent on an approximation. This comparison also yields another argument for the use of a game engine, particularly UE, in these scenarios, as the rapid change and the inclusion of weather effects rest on previous work already implemented in this software.

To accommodate modern and modular HPC systems, the pipeline working with the WebRTC [JBB21] implementation had to be adapted. Synavis was developed with the bridge functionality to allow this type of connection to be hosted on HPC systems. Video transmission is optional to accommodate the CPlantBox simulator endpoint. The JSON communication between the peers is used to control the virtual environment, and within SynavisUE, runtime information controls both the surfaces and the rendered image as runtime material. These runtime materials and the insertion of geometries into the virtual environment use the data connection, which has restricted message lengths. To avoid the overhead of an additional communication protocol when one is already established, the geometry information is split into packets of Base64-encoded data. The data is then reassembled on the receiving end, and the geometries are inserted into the scene as procedural content. Our article also showcases the use of templated base materials that runtime materials derive from, allowing a quick prototyping of virtual

scenes, even from a purely code-based approach. In these experiments, the FSPM has been grown to the stage measured in the experiment before the data is extracted from the virtual scene.

The article describes our implementation and evaluation of synthetic data. It is imperative that underlying hypotheses about real-world data or the assumptions made by the implementation of FSPMs are not violated by the transition and embedding in the virtual world. To this end, we used calibrated FSPMs of maize plants to recreate an experimental setup that is a virtual copy of a real experiment conducted at Forschungszentrum Jülich GmbH at the Institute of Bio- and Geosciences 3, also detailed in Bauer et al. [Bau+24] (**C-D**). When using the feature extraction pipeline that was used in Bauer et al. [Bau+24] on Synavis data, it should yield similar size distributions for each leaf, depending on its age and, thus, its size. We compared the *n*th longest leaf to the experimental distribution and showed that the synthetic data is slightly below but very similar to the experimental distribution.

This work has been presented at multiple conferences and workshops, including the Symposium on Functional-Structural Plant Models 2023 in Berlin, Germany. The software is open-source, and our investigations into data analysis pipelines using synthetic data on modular supercomputers provide key insight into scaling opportunities for a very data-scarce domain.

5.2 Article II: Immersive Root Annotation

D. N. **Baker**, T. Selzner, J. H. Göbbert, H. Scharr, M. Riedel, E. þ. Hvannberg, A. Schnepf, and D. Zielasko. "VRoot: A VR-Based Application for Manual Root System Architecture Reconstruction." In: *Plant Phenomics* (2025), p. 100013. ISSN: 2643-6515 DOI: 10.1016/j.plaphe.2025.100013.

This paper addresses TO**2** of the thesis objectives, focusing on the data extraction that is assisted by the direct visualization and interactive implementation of a root system architecture that allows for the tracing/editing of RSAs and the manipulation/screening of automated tracings. It directly uses the geometric embedding also developed for TO**1**, though its original root structure visualization predates that of Baker et al. (2023) [Bak+23]. The video description of this paper can be found at:

D. N. Baker. *Video on VRoot: An XR-Based Application for Manual Root System Architecture Reconstruction*. FigShare. 2024 DOI: 10.6084/m9.figshare.26003494.

5.2.1 Summary

The article describes our immersive RSA extraction application that uses HMDs to enable an intuitive and direct interaction with the data. In VRoot, as also visualized in Fig. 5.2, the 3D soil column scan is superimposed with an annotation of the root structure. The immersive implementation of the visualization and the interaction methods allows for a precise annotation because it removes positional or perspective-based ambiguity that is present in desktop implementations.

VRoot is implemented on top of UE, which serves as a platform for cross-device compatibility. It is an immersive application that allows the direct extraction of RSAs from 3D scans of soil columns. All visualization in the application is geometry-based, and appropriate interaction metaphors have been implemented that assist in the extraction of root system architectures in VR. This workflow is particularly useful in cases where automatic tracing fails to incorporate expert or situational knowledge, as previously reported by Horn et al. [Hor+21].

The implementation includes a data analysis server running on top of a Python[14] /VTK[15] software stack. Implemented methods are called, and data is communicated via ZeroMQ[16]. Most more complex tasks that require only basic interaction, such as topology changes to the RSA, are executed on the server. Both the 3D soil column scans and annotations are stored remotely, and the VR client receives geometric information as well as Root System Markup Language (RSML) data to construct an interactive model visualization. The server additionally keeps track of versioning and enables the undo action for the client side. A user-centered view of the application can be seen in Fig. 5.2.

Within UE, the embedding of the data is done with the geometrization scheme developed for TO**1** in Baker et al. [Bak+23]. For the visualization, different methods of translucency are used to enable the perception of depth differences. Users can choose their posture, and the application uses fully dynamic elements. 3D scans must adhere to common formats, and the resulting RSA is output and read in RSML format ideally, though polyline-based formats are also supported. The choice of interaction metaphors was based on feedback from a core user group, together with the observation that a high-precision extraction tool requires fine-tuning on different scales of the data. For example, the data set size can be changed, and users have been observed to refine the tracing at different scales in the course of the annotation.

Overall, the application enables direct interaction with the data for annotation and correction of tracings that were produced by automated tools. The user study performed to analyze the application showed that untrained users perform better in the median when using VRoot compared to NMRooting, while the root length extraction yields a lower deviation from the correct length, as shown in Fig. 5.2. The study addresses whether the immediate interaction with the embedding in a virtual space through more intuitive metaphors can improve the data extraction process, as posed in TO**2**. Specifically, there are tool sets around topological editing, also developed for the article by Selzner et al. [Sel+23] in the contribution (**C-C**). A more full-fledged application, with a focus on root system annotation and the tools needed to quickly fulfill the tasks as an untrained user, has been developed for the article in Baker et al. (2024) [Bak+25b], and further refined as presented in the demo [Bak+24] (**C-E**). Our contribution is both methodological and multidisciplinary: We developed an immersive method to extract RSAs, which we evaluated to yield significant improvements both with and without noise effects. The development of VRoot is part of a larger workflow to couple data to models, which is further detailed in Selzner et al. [Sel+25] (**C-F**). The steps between data extraction and FSPM simulation are crucial, and a joint interplay between DNN

[14]Python: python.org
[15]VTK: vtk.org
[16]ZeroMQ: zeromq.org

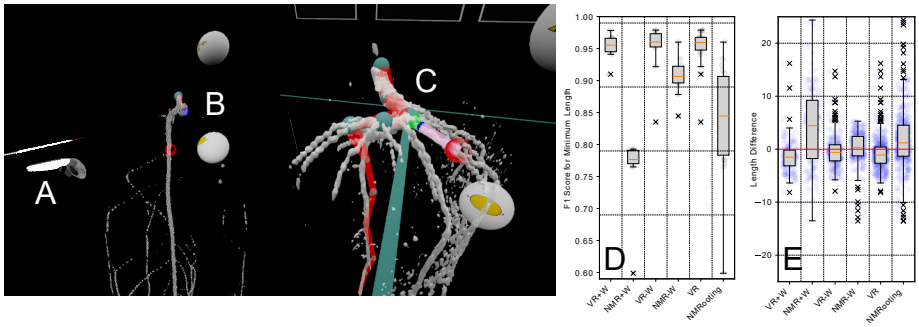

Figure 5.2. Overview of key points of VRoot in practice. A: The user uses an HMD to interact with the data. B: More complex interaction tools are available through widgets or menu buttons. C: The FSPM structure is directly visualized and interactive. D: Untrained users perform better in the median when using the immersive application compared to a standard workflow in a laboratory user study. E: The high-precision metric of deviating from the correct root length is also lower on average when using immersive techniques. Adapted from Baker et al. [Bak+25b].

methods [Sel+23] and manual correction [Bak+25b] is necessary to ensure the quality of the data [Sel+25]. The application is open-source and has been presented both internally and at multiple conferences and workshops, including the ACM Symposium on Virtual Reality, Software and Technology, seen in Baker et al. (2024) [Bak+24] (**C-E**).

5.3 Article III: Synavis Parallelization of Unreal Engine

D. N. **Baker**, F. Bauer, A. Schnepf, H. Scharr, M. Riedel, J. H. Göbbert, and E. Hvannberg. "Adapting Agricultural Virtual Environments in Game Engines to Improve HPC Accessibility." In: *Nordic e-Infrastructure Tomorrow*. Ed. by A. Azab and T. Malkiewicz. Cham: Springer Nature Switzerland, 2025, pp. 152–167. ISBN: 978-3-031-86240-3 DOI: 10.1007/978-3-031-86240-3_11.

This paper, submitted to and presented at the Nordic e-infrastructure collaboration conference in Tallinn, Estonia, has been written to fulfill TO**3** of the thesis objectives. The paper describes the parallelization of UE on HPC systems, supported by the Synavis framework and in preparation for its use in TO**4**.

The video description of the paper has been uploaded to:

D. N. Baker. *Scene Distribution using Synavis for Game Engine HPC Use-Cases*. FigShare. 2024 DOI: 10.6084/m9.figshare.25723773.

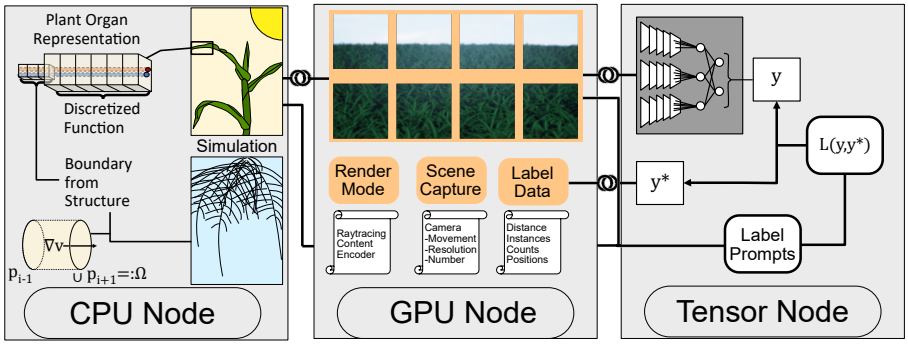

Figure 5.3. Overview of how the HPC embedding uses a distributed but concurrent workflow of simulation, rendering, and DNN training. The FSPM uses a discretization of its model but is ultimately solved as one task per plant. The rendering is distributed according to either domain decomposition based on field segments or based on interactive data streams. The role of the DNN training is to provide active learning feedback. Baker et al. [Bak+25a].

5.3.1 Summary

UE and rendering tasks that rely on all rendered data being locally available are challenging to parallelize. Image stitching is a method where rendered sections of the image are distributed to GPUs, allowing for the task-based parallelization of the rendering workload. However, data-parallel approaches are generally more beneficial since they distribute the workload across GPUs, but there might not be a image-based domain partition available for certain render tasks. Our analysis has shown that the number of concurrent FSPM simulations actively updating the UE scene will decrease the rendering speed to below 25 frames per second at ≥ 2000 FSPM instances. This is a challenge when using large-scale scenes, as the render performance required to produce a drone footage-like scene is memory-bound. However, within the use case of crop production, typically, the focus is on a smaller area per image, meaning that the actual field size is beyond the scope of any current training algorithm being able to deduce field properties. This means that to distribute the scene, remote parts of the field can be omitted from the rendering instead of resorting to level-of-detail approaches.

Virtual environments are hard to parallelize, as the render focus and the point of greatest detail might need to shift to allow the rendering of a broader area. Particularly in aerial vehicle-based remote sensing simulations, these points are widely distributed. Autonomous systems for measurement and intervention are important tools for more accurate, high-yield crop production as discussed in Taiz [Tai13]. One essential factor is that in multi-agent systems, such as implemented by Skobelev et al. [Sko+18], the individual agents might have different reference frames and could be either in line of sight or not in line of sight with one another. Visualizing cooperative agents that can view each other is harder to parallelize if the crops need to be visualized at high detail or with physical embedding as well. Figure 5.3 shows the infrastructural markup that was

derived for this work. The actual computation of the FSPM simulation is typically done on a CPU (or a CPU compute node), as the diversity of the individual models means a GPU computation is bound by the transfer of information. In the experiments described in this paper, the FSPMs are simulated successively, and the snapshot geometries are visualized in UE while the FSPM continues simulating.

Our pipelines offer unique possibilities when coupling DNN training to UE. The primary distinction from other frameworks is that we utilize a loose coupling to enable scalability and usability across cluster modules. The pipelines are especially useful for the LUMI supercomputer, which uses dedicated nodes for vectorized calculation but does not allow graphics to run on its primary GPU nodes [Mar+22]. This challenge is at the core of what the effort regarding TO**3** was about - the implementation of a coupling and data generation system that is compatible with other modern efforts to scale DNN training on HPC systems. The research question TO**3** also provides a valueable contribution to later work in TO**4**, which depends on the scalability of the data generation pipeline.

Not only does our approach facilitate HPC use for model-based training, but it also encourages the mostly physics-driven community of high-performance computing to consider alternative use cases of infrastructures. HPC funder and centers often acknowledge that visualization is a useful and often necessary tool to inspect data and scale simulations in a matter that is most useful to scientists, but oftentimes, funds are not allocated for these modules. An example of this is the JUPITER system, whose early benchmark process is focused on the parallel implementation of simulation architectures to gear them towards exascale computing [Her+24]. The architecture of JUPITER [Dum23] is indeed focused on extreme-scale computing and does not currently allocate a visualization module. While insights, especially in-situ monitoring of results, has proven to be an effective and sometimes necessary [Bau+16] tool when using the large-scale resources available, funds for the specific support of this kind of use case are often underallocated. We discussed this both in our publication [Bak+25a] and on the conference.

5.4 Article IV: Synavis-Driven Photosynthesis

D. N. **Baker**, M. Giraud, J. H. Goebbert, H. Scharr, M. Riedel, E. T. Hvannberg, and A. Schnepf. "Virtual World Coupling with Photosynthesis Evaluation for Synthetic Data Production." 2025 DOI: 10.1101/2025.02.06.633870.

This article was written in partial fulfillment of TO**4** of the thesis objectives. The article has been sent for review to in silico plants. We present a functional coupling between the FSPM CPlantBox and UE.

The video description of the paper has been uploaded to:

D. N. Baker. *Video to: Virtual World Coupling with Photosynthesis Evaluation for Synthetic Data Production*. FigShare. 2025 DOI: 10.6084/m9.figshare.28280780.

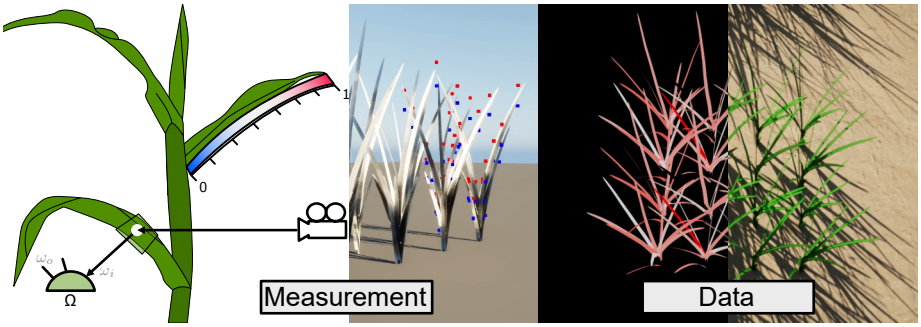

Figure 5.4. Overview of the sampling process for light evaluation. The measurement is done using virtual screen space sampling using path tracing. Surface properties are implemented using a model scene that reflects appropriate wavelengths. The data mapping is a texture mapping in a data scene with textures. Adapted from Baker et al. [Bak+25c].

5.4.1 Summary

Photosynthesis equations depend on both the solution to the equations governing the biochemical reactions and the interplay between the photosynthesis solver and the other parts of the plants that inform its boundaries. For a well-posed system, the coupling between the individual organs and plant systems needs to be densely defined. CPlantBox contains a photosynthesis module, developed by Giraud et al. [Gir+23], which solves the photosynthesis problem based on similar models developed by Farquhar et al. [FCB80] and Leuning [Leu95].

The primary purpose of geometric modeling of plant data is the computation of radiative transfer true to the actual surface of the plant [JG04]. For this purpose, techniques that evaluate photon propagation through 3D geometric data have been developed, illustrated in Fig. 5.4 left. Since this development was driven from a remote sensing perspective, such as in Malenovský et al. [Mal+21], it somewhat diverged in terms of new technical advances from the base raytracing algorithms. Very similarly, computer graphics also developed radiative transfer models for physically-based rendering. The similarity of the techniques and their overlap was highlighted in a study by Salesin et al. [Sal+24]. In UE, the rendering pipeline uses as much separation of components as possible, as game developers tend to optimize the rendering pipeline by omission of certain components. An example of this is surface scattering, which uses a model by Jimenez et al. [Jim+15], and can be turned on and off along with the remaining material shader pipeline.

This publication highlights our work to elevate the FSPM visualization to a true model embedding. Three primary contributions are highlighted, especially due to their relevance and relative novelty when combined into a single pipeline. First, we coupled our synthetic data pipeline to a specific functional model to compute the photosynthesis equations within the virtual world. The model is fully integrated into the pipeline, and the data is directly visualized. This functional view of the model is visualized mid-measurement in Fig. 5.4. It is an indirect response to a common critique of model-

driven data pipelines: Typically, these pipelines are one-directional, as reported by Li et al. [Li+24]. Though Synavis [Bak+23] does not fall into this category because of its inherent online-generation aspect, an extension of the pipeline by including functional information is an important step towards using automated systems to discover hidden functional traits in plants.

Second, the output of the photosynthesis model is directly visualized and implemented into a data generation pipeline. The primary purpose of this coupling is to allow for an efficient in-place evaluation of functional properties without needing to couple to an additional GPU-intensive task separately. The UE coupling is effective and efficient in its evaluation, as we condense spectrum-wide information into a modeling platform that evaluates quickly based on surface property calibrations. This data view that includes light influx information is visualized in Fig. 5.4.

A common critique of pipelines that use graphics engines is the validity of the data. In contrast to non-heuristical frameworks such as DART [JG04] that simulate the whole spectrum, engines such as UE are often considered unreliable visualizers [KS23]. This critique directly questions the use of game engines as a valid heuristic for data visualization and in this research also the functional embedding. However, the representativeness of a reduced-spectrum view is not a question of implementation but of data calibration, as the underlying principles are the same [Sal+24], such as implemented in UE by Jimenez et al. [Jim+15] for surface scattering. In Baker et al. (2025) [Bak+25c] we show that the replication of an experiment using our framework, is possible when effective checks are used and the scene is properly calibrated. Examples of adaptions to the render pipeline are: disabling of geometry culling, increased lightmap resolution or path tracing, conversion of experimental measurements into UE units and subsequent check using clearsky measurements for sun radiation.

Third, the HPC embedding of the pipeline has been evaluated on the supercomputer JURECA-DC [Thö21]. This evaluation is important as it provides a baseline measurement for this pipeline and for the implementation of the task distribution inside the UE-Synavis pipeline and framework. The implementation is sufficienctly scalable to match the distributed training scalability of DNN pipelines [Sed+19], which is a core aspect of this thesis, namely TO**1+3+4**.

6 Conclusion

The aspects of a full data generation and training pipeline are diverse and for plant science also challenging to combine. This thesis not only provides a framework to couple the simulations that contribute to plant function to a rendered virtual world, but it also provides insights on how to embed, scale, and adapt virtual world data to accommodate machine learning pipelines.

6.1 Summary

The work presented in this thesis yielded a framework and new method for generating and steering virtual environments based on loose coupling between a simulation or training framework and the virtual environment (TO**1**). The primary contribution of this framework is the establishment of a coupling concept that does not only support high-fidelity rendering on dedicated hardware but is also compatible with diversifying hardware and modular HPC system constraints.

The use of virtual worlds to assist synthetic data as well as data extraction through the use of immersive techniques rests on developing visualization techniques for plant models that not only accurately reflect their simulation but also enable direct interaction with the underlying database (TO**2**). The use of FSPM models for the generation of synthetic data is a novel approach as it provides a basis for a very complex model coupling. Models such as CPlantBox are designed to be extendable and the boundary communication between plant functions allows for dedicated computing on separate hardware. Individual coupling mechanisms vary, using a streaming protocol for Synavis, while VRoot uses a message-passing interface, but the general concept of the geometric embedding into the virtual world remains the same.

The main goal of this thesis, increasing the utility of partially limited and precious experimental data, has been tackled in three ways. Firstly, the use of FSPMs to generate synthetic data has been established, which is a novel approach to addressing data scarcity in plant data analysis (TO**1**). Secondly, both the parallelization and the combined embedding of experimental data and FSPMs into the virtual environment advance this approach to not only be scalable but also more biologically relevant while retaining the advantages the use of a graphics engine provides (TO**3**). Thirdly, the FSPM structural data embedding and the involvement of interaction techniques have shown, provably, that the use of immersive techniques enhances an individual's ability to extract data from noisy or complex experimental data sets (TO**2**). This extends the potential to use more realistic experimental conditions without needing to resort to either expensive equipment or methods that induce plant stress. The accommodation of more realistic

measurement scenarios is particularly highlighted when using FSPMs to facilitate a domain-specific data production pipeline, which allows for the generation of applicable synthetic data with real-world use cases (TO**4**).

6.2 Future Directions

A host of use cases would need more support to enable this pipeline for more scalable applications. One such use case would be the automotive industry and autonomous driving, which has not only seen a lot of development but is also in dire need of robustness training. Generally speaking, in many instances, the challenge is less setting up the data pipeline and more faithfully representing a real-world object, such as a robotic device, in a virtual world. This challenge is being tackled in numerous ways, but the overall pipeline of using real-world models that depend on sometimes fine-grained physics to inform its interaction with the world is still a challenge. For certain applications, it also remains to be determined whether this level of physical accuracy is needed. However, autonomous driving is a problem for synthetic data production which depends on the algorithm knowing the car's capabilities and physical properties. The Synavis framework is suited for these use-cases for the orchestration of the data streams, but a down-side of the Synavis framework is the fact that for physics-based learning models, the UE scene needs to be adapted, as this type of information cannot be established at runtime due to how UE is implemented.

Large-scale DNNs need to be trained to facilitate a greater expansion of present best practices and work dedicated to training DNNs is needed to uncover new paradigms in training using synthetic data, especially from the standpoint of online randomization of training data. Domain-based randomization is a powerful tool, and its expansion into a more sophisticated physics-based space would greatly benefit many use cases, such as drone footage. One use case that could benefit from a scalable embedding is to not only train a large model that detects draught stress, but also to proof it against false-positives to preserve resources.

Lastly, synthetic data production and particularly a more fine-granular geometrization of plant organs, or geometric data from other domains, could greatly benefit from generative methods that produce geometries. Approaches such as Shu et al. [Shu+19] have the potential to increase the realism and augmentation performance of synthetic data frameworks. The inclusion of DNNs into the visualization pipeline could benefit the data production as well as the data annotation. Assisted annotation could save a lot of time for manual expert annotation in applications such as VRoot, which has been shown by McDonald et al. [McD+20] for a neuron tracing application. However, a high level of precision and robustness is needed, otherwise users would need to fallback on manual methods again with the risk of increasing workload beyond a purely manual annotation.

Concisely, virtual worlds are an important tool that can benefit the robustness of DNN approaches and the rigurous validation of them in synthetic, even unlikely, scenarios might also improve public trust.

Article I

A scalable pipeline to create synthetic datasets from functional–structural plant models for deep learning

D. Baker, F. Bauer, M. Giraud, A. Schnepf, J.H. Göbbert, H. Scharr, E.Þ. Hvannberg, and M. Riedel

https://doi.org/10.1093/insilicoplants/diad022 2023

Submitted to in silico Plants in August 2023, Accepted December 2023

in silico plants, 2023, **6**, 1–18
https://doi.org/10.1093/insilicoplants/diad022
Advance Access Publication 08 December 2023
Technical Advance

Technical Advance

A scalable pipeline to create synthetic datasets from functional–structural plant models for deep learning

Dirk Norbert Baker[1,2], Felix Maximilian Bauer[3], Mona Giraud[3], Andrea Schnepf[3,*], Jens Henrik Göbbert[2], Hanno Scharr[4], Ebba Þora Hvannberg[1] and Morris Riedel[1,2]

[1]School of Engineering and Natural Sciences, Sæmundargata 2, 102 Reykjavík, Iceland
[2]Jülich Supercomputing Centre, Forschungszentrum Jülich GmbH, Wilhelm-Johnen-Straße, 52428 Jülich, Germany
[3]Institute of Bio- and Geosciences 3 (Agrosphere), Forschungszentrum Jülich GmbH, Wilhelm-Johnen-Straße, 52428 Jülich, Germany
[4]Institute for Advanced Simulation 8 (Data Analytics and Machine Learning) Forschungszentrum Jülich GmbH, Wilhelm-Johnen-Straße, 52428 Jülich, Germany

*Corresponding author's e-mail address: a.schnepf@fz-juelich.de
Citation: Dirk Norbert Baker, Felix Maximilian Bauer, Mona Giraud, Andrea Schnepf, Jens Henrik Göbbert, Hanno Scharr, Ebba Þora Hvannberg, Morris Riedel. 2023. A Scalable Pipeline to Create Synthetic Datasets from Functional-Structural Plant Models for Deep Learning
In Silico Plants **2023**: diad022; doi: 10.1093/insilicoplants/diad022

Handling editor: Tsu-Wei Chen

ABSTRACT

In plant science, it is an established method to obtain structural parameters of crops using image analysis. In recent years, deep learning techniques have improved the underlying processes significantly. However, since data acquisition is time and resource consuming, reliable training data are currently limited. To overcome this bottleneck, synthetic data are a promising option for not only enabling a higher order of correctness by offering more training data but also for validation of results. However, the creation of synthetic data is complex and requires extensive knowledge in Computer Graphics, Visualization and High-Performance Computing. We address this by introducing *Synavis*, a framework that allows users to train networks on real-time generated data. We created a pipeline that integrates realistic plant structures, simulated by the functional–structural plant model framework CPlantBox, into the game engine Unreal Engine. For this purpose, we needed to extend CPlantBox by introducing a new leaf geometrization that results in realistic leafs. All parameterized geometries of the plant are directly provided by the plant model. In the Unreal Engine, it is possible to alter the environment. WebRTC enables the streaming of the final image composition, which, in turn, can then be directly used to train deep neural networks to increase parameter robustness, for further plant trait detection and validation of original parameters. We enable user-friendly ready-to-use pipelines, providing virtual plant experiment and field visualizations, a python-binding library to access synthetic data and a ready-to-run example to train models.

KEYWORDS: Computer vision; deep learning; FSPM; HPC; synthetic data; visualization.

1. INTRODUCTION

Machine learning (ML) algorithms usually perform well when trained on large quantities of data well covering the input space. Deep learning (DL) techniques are a subset of ML and utilize the training of many-layered compute graphs. Pound *et al.* (2017) show that DL techniques have the highest performance on plant image analysis, which, in turn, has been established to be a significant bottleneck for plant phenotyping (Tsaftaris *et al.*, 2016; Kamilaris and Prenafeta-Boldú, 2018; Yang *et al.*, 2020; Scharr and Tsaftaris, 2022). Many image-based applications for plant phenotyping involve semantic or instance segmentation, that is, associating each pixel with a class label such as organ type and/or organ numbering (Scharr *et al.*, 2016; Tsaftaris *et al.*, 2016; Yang *et al.*, 2020). In terms of DL in biological image analysis, data are often rare and hard to extract from real-world measurements (Pound *et al.*, 2017). This is due to the high variability of environmental conditions, including light conditions, rain, soil types, stresses influencing plant appearance even for the same genotypeand the high intrinsic variability of plants and their changing appearance over time due to growth or senescence. Even lab condition data cannot always be acquired in optimal circumstances and cannot easily be reproduced as plants would need to be regrown. Consequently, data sets acquiredover years of intense

measurement campaigns are often heterogeneous and can only cover small parts of a large data space. Therefore, developed plant image analysis solutions are typically highly specialized, for example, for specific organs and/or plant types, and do not generalize broadly (Lobet *et al.*, 2013). In most cases, the full potential of the data remains unused. Analysis methods that exploit large quantities of heterogeneous data sets, covering a more significant part of the data space, would be highly beneficial for robustness and generalization.

DL frameworks and algorithms can address data analysis challenges but need a lot of data to perform well. Synthetic data generation can provide arbitrary amounts of well-annotated data, thus enabling the scaling of DL methods to a more powerful capacity for generalization and accuracy. Previous approaches showed that introducing synthetic data makes high-quality training examples available at low cost (Pollok *et al.*, 2019; Zhang *et al.*, 2020). The generation of versatile synthetic data, however, requires a lot of inter-domain expertise on plant biology as well as computer graphics and High-Performance Computing (HPC). Solutions to generate more data for specific tasks often end up highly specialized, such as for 2D image generation approaches by Ubbens *et al.* (2018), who generate leaf images for counting tasks or 2D root rasterization with noise components as implemented by Lobet *et al.* (2017). Similar approaches for root systems have also been implemented by Benoit *et al.* (2014) or the 3D voxelization approach by Masson *et al.* (2021). Functional–Structural Plant Models (FSPMs) produce, using cultivar-specific input parameters, plant morphological and topological data that can then be used to produce synthetic images. To assist with making FSPM rendering more realistic, some approaches also include physics-based surface simulation and reaction to light spectra (Bailey, 2019). In contrast, Hartley and French (2021) address the potential synthetic-to-real data gap by applying domain adaptation using image generation, such as Gao *et al.* (2023). Other approaches additionally use few-shot learning, that is, a limited number of observations/samples, and transfer learning to bridge the gap between synthetic and authentic images (Zhang *et al.*, 2020). An example application where synthetic data can be very impactful are rhizotron experiments, which is a method for whole-plant measurement that requires plants to be grown in specialized containers, such as produced by Nagel *et al.* (2012). The measurements of this type of special setup are resource consuming and produces accurate but little data. The significant overhead for a single plant measurement causes challenges for the DL models that need to work on limited data as well as a need to process the present data with as much accuracy as possible.

To use the full potential of synthetic data in the training of DL approaches, we developed Synavis (Synavis Framework: https://github.com/dhelmrich/synavis), a coupling framework that is versatile enough to allow different approaches and makes use of HPC to allow an interplay of FSPMs, visualization and DL model training. We developed a pipeline using visualizations of the FSPM CPlantBox (Zhou *et al.*, 2020) and the Synavis software, enabling the coupling of the FSPM with Unreal Engine (UE) (Unreal Engine, https://unrealengine.com/) and the DL framework. Synavis handles dynamic workflows with UE and is scalable for HPC systems. The combination of CPlantBox with Synavis in UE also allows for the easy addition of wind, leaf diseases imposed onto the leaf texture, and other effects such as rain or degraded image quality. The primary goal of this work is to improve image analysis processes for plant phenotyping and model parameter extraction. To support the data generation for these difficult tasks, there are bottom-up approaches to directly simulate surface radiation (Bailey, 2019) and top-down approaches aimed at replicating measures (Bouvry and Lebeau, 2023). Synthetic data can also be generatively produced to assist with specific tasks, such as leaf segmentation (Ward *et al.*, 2018).

This article highlights the central points of our contribution, which are centred around three main axes. We implemented a geometrization and texturization for FSPM Visualization with CPlantBox. Furthermore, we introduce our Synavis framework that enables the integration of CPlantBox models into UE in an HPC compatible method for the purpose of flexible synthetic data generation. We test our approach against experimental data by implementing the specific use case of rhizotron experiments to demonstrate feasibility and performance. Furthermore, we detail the field scale rendering in our virtual environments and what aspects of the pipeline give this method advantages over other approaches.

1.1. Description of the CPlantBox FSPM

FSPMs describe digitized versions of a phenotype (Soualiou *et al.*, 2021) of a plant, providing means of assessing interventions, crop combinations, photosynthesis assessments and even nutrient and soil interaction. They can provide insight into *in vivo* counterparts by providing access to more measures and modeled information, making them ideal digital twin candidates. The interaction between the *in silico* and *in vivo* versions of plants can provide valuable insight, especially concerning possible success from interventions, as shown by Perez *et al.* (2022).

CPlantBox (Zhou *et al.*, 2020; Giraud *et al.*, 2023) is a modeling framework for FSPMs based on the graph formalism as seen in Fig. 1. Plants are described as a series of vertices linked by edges describing the abstract morphology of the plant. The plant object stores, moreover, a series of arrays that can contain any kind of necessary information for each point or edge, such as radius or age. CPlantBox is a stochastic model, where all parameters are defined via their truncated normal distribution. CPlantBox can be used to generate raw plant structural data mimicking various plant development dynamics under specific environmental conditions. Parameterization of CPlantBox is illustrated in Fig. 1 and can be done in two ways: Calibration of the model are sometimes done directly from experimental data, such as in Bauer *et al.* (2023), who investigate the effect of phosphorus deficiency in *Zea mays* plants. Another approach is measuring a target distribution from experimental data, and running an estimator for the posterior distribution of the parameter space w.r.t. the target distribution, as done by Morandage *et al.* (2021) in their work on analysing how well model parameterizations can fit onto synthetic field data. Additionally, CPlantBox can be coupled with other models that simulate dynamic soil or atmospheric conditions. Such models can exhibit growth properties that are grounded in, for example, nutrient availability, such as developed by Giraud *et al.* (2023).

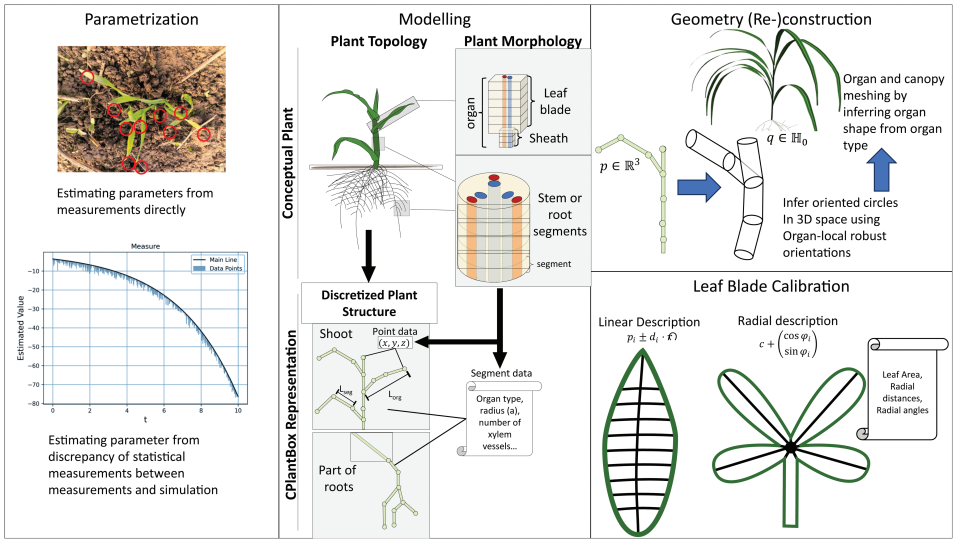

Figure 1. Overview of CPlantBox parametrization, adapted from Giraud *et al.* (2023), simulation of structural and functional properties, and visualization. Parametrization exemplifies different measuring techniques—direct and through stochastic optimization. Modelling shows our approximation model for the true plant shape, which is the discretized point/edge model. We further introduce two options for leaf-blade calibration. The visualization approach introduced with this work infers a 3D plant shape from the graph-like structure by assuming that by default, there are no drastic local changes to the rotation.

2. METHODS

2.1. Modelling CPlantBox in Unreal Engine

As one of the cornerstones of our synthetic plant data pipeline, this section will detail our 3D meshing and the modeling framework for texturing plants in SynavisUE.

For the generation of the CPlantBox geometry, as seen in Fig. 1 on the right, we implemented a visualization module that produces triangulated meshes from the CPlantBox data. For this, we needed to write a geometrization scheme that could convert the point/edge graph structure into a mesh. The mesh is then rendered in UE as object that can be placed in the scene using Synavis.

Generally, models of the leaf surface should incorporate its morphological properties. It greatly enhances the realism and visual quality of the rendering, as was shown by Wang *et al.* (2006). While there are different established ways of producing and texturing leaves in graphics engines, one of the more common methods is masking. This technique displays a rectangle that contains a texture with an binary mask that describes the leaf area. As exemplified in Bailey (2019), masking the leaf surface as a texture is a good tool of fast creation of varying plant shapes as the masking is very cheap in both ray-tracing and rasterization renderers. It has drawbacks, however, as modern graphics engines such as UE might introduce additional geometric features, for example, Nanite Hierarchical Culling (Karis, 2021), which might change masked geometries more than their unmasked counterparts, or their rendering does not accurately reflect the geometry of the plant as optimizations are made. This leads to a discrepancy between the masked leaf shape and what is visible in the scene. Another approach is to declare a full triangulated geometry for the shape of the plant, such as performed by Yun *et al.* (2019) for simulated Light Detection and Ranging (LiDAR) data (Behroozpour *et al.*, 2017). In this work, we use the latter approach for producing leaf shapes. The advantage of this is that there exists a pre-rendering correct geometry that DL approaches can train against. Our approach infers an orientation from the point position by either assuming that the local forward vector (i.e. in growth direction) is upright or by branching off from the parent organ. For leaf meshing, this orientation must be robust, as the inferred orientation at the origin point of the leaf blade has more extreme effects at the sides of the leaf. The leaf blade is scaled using the CPlantBox parameter sets that also contain its description. We detailed the CPlantBox geometrization approach in Appendix E. Fig. 1 also shows how the leaf area templates are saved in the parameter file, either as linear description as distances from the midline or as radial distance-angle pairs. Leaf blade distances are generally shape-only parameters and are pending the scaling using the leaf area parameters or growth stage.

For the visualization of CPlantBox plants, textures are warped onto the geometry. This means that there is an internal relative coordinate system spanning the leaf surface. This projection of the surface onto the texture is similar to some world map

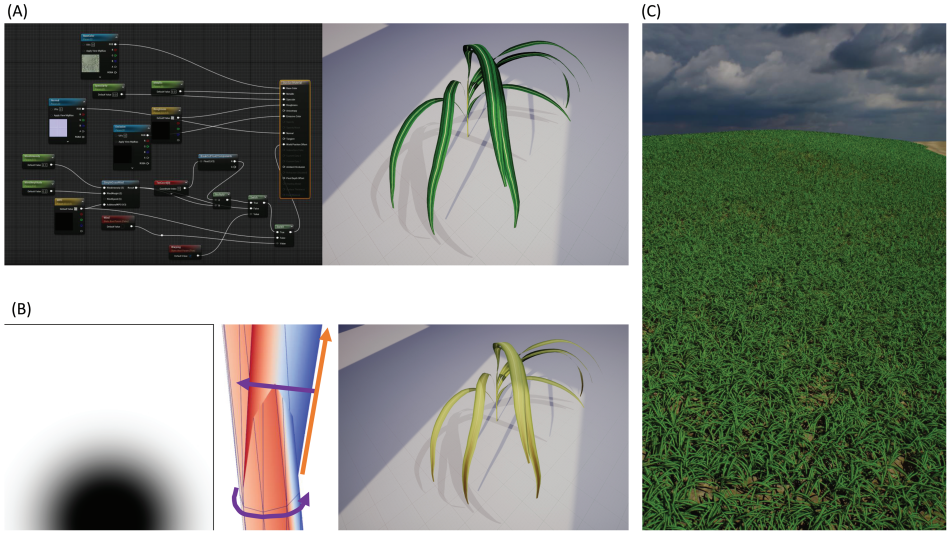

Figure 2. (A) We model the material, that is, the shader language graph representation, in UE for sample textures to make sure both dynamically set parameters and operations on textures are feasible. (B) Starting from this graph, we dynamically instantiate not only surface colour textures but also additional effects, such as normal maps or opacity masks. Example shows use of composition, with discolouration masks and parameters set to 0 by default (such as in A), which can by used during runtime to generate additional effects, such as discolouration. (C) UE is capable of handling a lot of geometries. While there are diverse geometry pipelines supported, Synavis uses procedural meshes as they directly accept geometry buffers. To differentiate between organs, their individual geometries have to be transmitted separately into mesh sections.

projections. This allows for more pixels at the leaf tip and the start of the leaf, where often more complex surface properties are exhibited as structures merge and leaf veins flow together. Leaf textures either stretch the whole leaf length or are automatically repeated by the UE. However, graphics engines also support the creation of rectangular textures that again provide more resolution along the leaf length. Masking the leaf texture comes at the cost of potentially not allowing for enough pixel space at potentially crucial parts of the leaf. With the warping of the leaf texture, we also make maximum use of the texture resolution and the texture buffers on the GPU. Moreover, there is an obvious midline on the leaf, which is exactly the $m : \mathbb{R} \ni x \mapsto (0.5, x) \in \mathbb{R}^2$ coordinate line in the texture. For masked textures in contrast, there always needs to be a fixed starting point and the midline of the leaf might not be obvious. Positioning leaf sickness effects or structural defects such as holes can be done by generating the appropriate discoloration and transmitting the texture as buffer to SynavisUE.

Stem textures are similarly warped, but the texture coordinates for cylindrical plant organs are wrapped around the axis counter-clockwise. Generally, this behaviour of the system can be adapted by changing the geometrization, but mapping discolourations onto parts of the plant is easier when using a normalized coordinate space.

Fig. 2 gives an overview over the individual steps involved. Users can achieve clearer data mapping and visualization when using warped data-containing textures as opposed to vertex colours. The surface modelling approach for the Synavis framework mainly rests on the material shaders in UE as shown in Fig. 2A. The structure of the surface descriptions allows for dynamic parameters that can be set during runtime, such as in Fig. 2B. Parameter ranges can be investigated within UE, such as how discolourations are projected onto the surface, while actual data augmentation is done through Synavis. Parameters introduced in shaders will be available for runtime adaptation. Visual editing modes in UE are helpful for validating value ranges and assessing how the material reacts to changes in values. This templated material increases the possible training data space significantly. Fig. 2B shows this with the example of a simplistically generated texture mask that is superimposed onto the leaf, generating a discolouration of the leaf surface. Field settings, such as Fig. 2C rely heavily on stochastic augmentations of either geometry or material, which are alterations of the basic parameters with random chance. Plant textures used in this example were generated using simple generative color filters and calibrated based on image data from experiments shown in Fig. 5. Soil and environment texture choice depends on what is being trained. If it serves only secondary purpose, we refer to either free-to-use

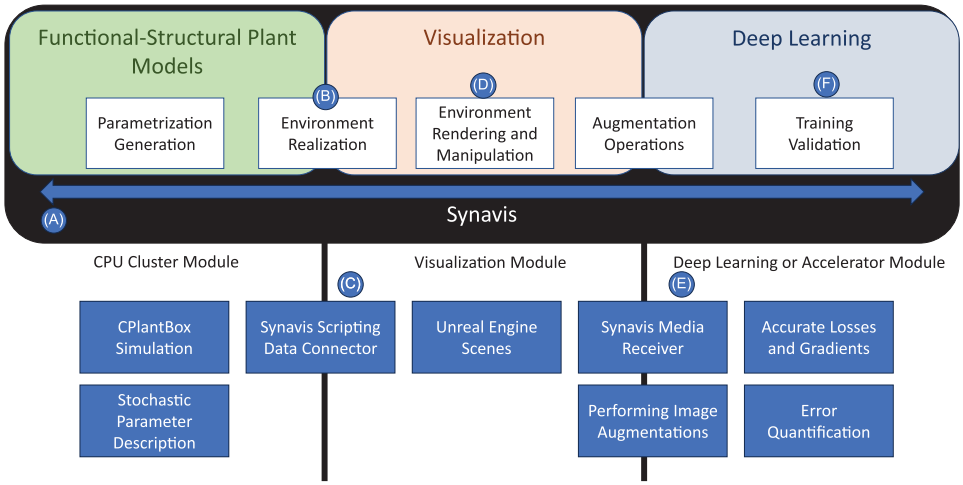

Figure 3. Overview of the total workflow setup. Synavis is a coupling framework that utilizes standardized communication across a supercomputing network. Oftentimes, different applications must run on specific architectures. This include highly parallelizable algorithms such as backpropagation for DL model training. In contrast do visualization, these can also run on accelerator compute nodes that mainly provide tensor cores, which, in turn, cannot provide rendering support.

scanned textures (examples include Quixel Megascans), or paid custom asset collections (Unreal Engine Marketplace).

The main difference of this work, separating it from all previous approaches, are three main points. Synavis enables a *versatile* workflow that works well on plant data generation, but is not restricted to it, as it can be added to any existing project. Furthermore, our approach is provable to be *scalable*, with its primary design being catered towards its use on HPC systems. Finally, we aimed to make it more *accessible* by using the default protocols and behaviours of UE, such as WebRTC (Jennings *et al.*, 2021). An overview of the framework components can be seen in Fig. 3.

As shown in Fig. 3A, Synavis handles the connection and communication between UE and the DL training frameworks. Furthermore, Synavis was designed as bridge service similar to the one described in Reddy *et al.* (2020), connecting compatible endpoints in a modular supercomputing architecture where software might run most optimally on specialized hardware (Suarez *et al.*, 2019). An important feature is the steering of the virtual scene using JavaScript Object Notation (JSON) commands, as in Fig. 3B. Synavis also integrates well with UE without requiring a direct coupling to it. The network interfacing used by this method follows a standard and provides simple tools. Moreover, this framework allows the coupling of UE to many different endpoints—providing simulation data, geometry, textures, commands, information for training or communication with the simulation.

During our investigation of current practice, we uncovered some key workflows that we wanted to support, with tangible applications either in demand, or already in use: Firstly, we provide access to otherwise inaccessible data for training workflows using synthetic data, such as depth estimation (McCormac *et al.*, 2017) with UE, as was already shown to integrate well with the underlying rendering pipeline (Jansen *et al.*, 2022). Secondly, we use UE to digitally mimic the environment for synthetic data such as light interception (Kim *et al.*, 2020), or scalable workflows to meet the rising demand for high-quality data (Scharr *et al.*, 2016; Pound *et al.*, 2017; Qiu *et al.*, 2017; Zhang *et al.*, 2020).

The central theme in these use cases is the combination of complex frameworks and very domain-specific workflows. Especially in plant sciences, in combination with visualization and DL, it can be a challenge to overcome the technical requirements of different systems and users. Synavis is a pathway for collaborative use of these techniques. Incorporating a specific visualization for an FSPM, plant scientists can dictate the look and structure of the virtual scene at runtime.

2.2. Central concepts of Synavis

Synavis uses the standardized WebRTC communication method as well as a command structure based on JSON. Generally, WebRTC is a framework to couple participants in real-time communication. This means that the communication is always non-blocking and reception, while possibly assured through certain protocols, does not occur in any predictable order. This setup is preferable when messages are sent by peers at the same time and an ordering might not be possible.

Fig. 3C shows that CPlantBox is connected to UE via a *Data Connector*, which is the main component that accommodates the possibility of transferring data to and from UE. The data

connector also introduces the possibility of connecting models directly to each other or to the DL framework. For communication with UE, the data connector provides functionalities consisting of a base set of commands that are pre-integrated to enable many simple workflows. As a method of communication, the data connector uses a WebRTC concept called *data channel*, which is the direct peer-to-peer message passing connection that is the minimum required setup for a connection with the framework. Enhancing the data connector with a *media track* yields a setup similar to a video conference client receiving a video stream, which in Fig. 3D serves as a UE–DL connection. This type of coupling class is called *Media Receiver* in Synavis and its added functionality is the capturing of video frame packages (Schulzrinne *et al.*, 2003). The media receiver also offers an easy coupling to decoding services that work with streams, such as OpenCV with GStreamer (Nimmi *et al.*, 2014). The setup of such an example can be found in Appendix C.

Finally, communication through the Synavis framework uses JSON. This method of communication encapsulates data well without requiring a lot of meta text and is the default for the Pixel Streaming Plugin as well. An example of a simple prompt would be `{"type":"query"}`, which prompts UE to relay the list of objects within the virtual scene. All objects are accessible that are present within the reflection system of UE. This system is a separate data structure generation system that contains meta information on classes, properties and objects of code definitions in UE.

2.3. SynavisUE visualization

UE is a graphics engine that has been in development since 1995 (Sanders, 2016). Apart from game development, it has uses in scientific and industrial disciplines (Qiu *et al.*, 2017; Bondi *et al.*, 2018; Zhang *et al.*, 2020). We are using UE in combination with Synavis to generate virtual scene data. Using a graphics engine such as UE has a number of advantages. Focusing on the ones that are directly relevant to our workflow, we want to highlight that (i) UE is capable of handling scripted runtime manipulation while offering a system for fast prototyping through editors. Whereas digital twin models require transference of measurements, the engine itself further enables the design and use of virtual environments that are designed rather than programmed—the user interface will make it much easier for most users to arrive at a visually realistic environment. Large-scale environments produced by the engine have already seen some synthetic data uses, such as the Unmanned Aerial Vehicle (UAV) approach by Bondi *et al.* (2018). (ii) UE, furthermore, uses a reflection system, which this framework makes heavy use of, as highlighted in Fig. 3E. Reflection systems are generally helpful in counting or selection tasks, retrieving properties of objectsand tracking objects. UE also grants access to the GPU buffers that contribute to the rendering of the final image, such as distance buffers. These buffers provide an expedient way of accessing distance and segmentation information, which has already been taken advantage of by approaches such as by Jansen *et al.* (2022).

PixelStreaming is a plugin that also changes the render pipeline of UE to include an encoding step, providing a video stream of the virtual scene. Typically, the *final image* buffer is the last step of the rendering scene, preceeded by pixel shaders. This image is usually transmitted to the display and discarded. The plugin handles the final image in an another step, encoding a video stream on the graphics card using the H264 (ISO/IEC 14496-10:2022, 2022) or H265 (ITU-T H.265, 2023) encoding standards. There are alternate CPU-based encoders that have performance drawbacks but increased compatibility, such as an arbitrary number of encoders. The encoded image is transported using Real-Time Protocol (Schulzrinne *et al.*, 2003) packages. The plugin enables easy and web-based access to a stream from a data centre's backend. We are using this pipeline to enable our coupling between the visualization with UE and the DL pipeline. We developed a managed way of coupling video streams with DL frameworks through a framework akin to a kind of relay server (Reddy *et al.*, 2020). This standard also provides us with means of inserting data into UE.

2.4. Training with Synavis

The central techniques we will illustrate in this work, which enable workflows driven by Synavis, are *tracking*, *segmentation* and *mapping*. This relates to Fig. 3F.

Tracking is a method of obtaining frame-wise information about elements of the virtual world at each point in time. The temporal resolution of this information is ultimately dependent on the frame time of the engine. However, the temporal resolution of the tracked information always matches the temporal resolution of the world. Prompting tracked information is done using a JSON prompt that triggers the transmission of the tracked information through the data channel. The information is sent every frame, and to compensate unordered arrival of the data channel messages relative to the video track frames, a timestamp is added to the messages to provide ordering information. As this timestamp is also present in the video track messages (Schulzrinne *et al.*, 2003), the messages can be ordered. In our use-cases, we train through the use of an intermediate image buffer. This allows the training to progress through the data set at any speed, decoupled from the rate of image production. While this is not practically an issue, it does alleviate some scheduling and fetching delay concerns for supervised learning.

Segmentation and mapping are both techniques that require pixel-level information. UE can deliver semantic segmentation maps via multiple techniques, specifically a separate depth rendering pass for segmentation maps. This way, objects can be assigned an ID that will be rendered to ground truth by SynavisUE. Mapping on the other hand, is largely related to mapping properties onto the scene, on the geometry level or on population level. Plant surfaces can contain values mapped onto colours much like in standard visualization pipelines. In Synavis, these additional information carriers are then marked as visible only to the *information camera*. Alternatively, the information can also be mapped on patches of the image rather than the actual plant geometry. This way of generating ground truth is also possible remotely within Python.

Overall, Synavis offers direct callbacks for frame reception or alternatively, rerouting into decoding frameworks, such as GStreamer, which is commonly used for remote video platforms (Nimmi *et al.*, 2014; Ahmadi *et al.*, 2016). The base implementation to couple Synavis to a DL framework is implemented as a

buffering of images, such that the reception and decoding does not limit throughput in the training framework. This handling, however, is implemented within the DL framework and while Synavis offers a template on how to handle this, there are many hyperparameters that limit generalization capacity of any single implementation, such as output resolution, or whether images or a video stream is needed. Image augmentations that are being considered best practice (Kuznichov *et al.*, 2019) should still be used, albeit as a precursor to fetching the next image as opposed to an operation on the data set. In this way, the actual image is drawn from a selection of equally likely possibilities $I \in \{I, I^T, (-x, y)_{x,y\in I}, (x, -y)_{x,y\in I}, ...\}$. This augmentation should be random depending on rank, such that the individual instance will not usually receive the same image during training.

3. RESULTS

The creation of synthetic data for model training is useful because it is scalable (Scharr *et al.*, 2016; Qiu *et al.*, 2017). The strongest advantage, even surpassing any potential synthetic-to-real data gap, as exemplified by Zhang *et al.* (2020), is the ability to generate varying data of different visual properties, while retaining exact information on training labels. A non-exhaustive list of randomization options can be found in the Appendix A, where remarks are included on how and when to introduce stochasticity to virtual scenes.

We generally recommend making as much use of stochasticity as possible. A strong reason for this is the fact that previous investigations into learning behaviour of DL models found that more general pre-training greatly improves later out-of-distribution performance (Jitsev, 2021).

We note that for entirely custom scenes, UE provides a full-featured 3D editor as well as integration of common architectural data types, such as computer-aided design (CAD) drawings. Our framework, furthermore, targets runtime-creation of scenes. Initializing and coupling Synavis to UE, or its respective SynavisUE plugin, the data channel connection provided is capable of handling operations such as spawning and scene manipulation. SynavisUE allows a combination of pre-made assets and scenes with runtime generation of objects, and anything on the scale of purely one or the other. Steering of the scene is done using the Python bindings of Synavis, which allows for the direct interaction of the DL framework with the virtual scene, including the training of agents using reinforcement learning.

3.1. Setting up data generation for scalable use cases

Important aspects of the rendering should be changeable. It is important to note that in a lot of cases, such as UnrealPerson (Zhang *et al.*, 2020), synthetic-to-real transfer learning yields better results than simple application of a synthetically trained model onto real-world data. As such, we setup HPC-targeted workflows and highlight how the different aspects of the scene can be changed to target scalability rather than exact replication. UE uses a game loop that constantly produces new images as part of the rendering.

Through the reflection system, any registered parameter and function can be accessed. Parameters can be changed, and Synavis can spawn geometries as well as textures. Synavis was further designed to make educated guesses as to the properties of objects, so that scene setup is fast while retaining versatility.

Users should change positions, materials, geometries if applicable as well as the camera properties, as this further introduces data diversity. More possibilities and descriptions can be seen in Appendix A.

3.2. Pipeline performance on the JUWELS Booster

In our measurements, we focused on the performance of the individual pipeline components that contribute to the total performance. As such, we measured the *field-filling* capacity of CPlantBox, meaning the timing of the generation of enough plants to realistically fill a plant field in Table 1.

Furthermore, we measured the performance of Synavis for different message passes. We measured the performance of Unreal Engine in dealing with the rendering of a large-scale plant field, as is also shown below in Section 3.3. The results of the frame time measurements can be seen in Fig. 4.

Queries for objects or properties are delegated to either the reflection system or the object management system. Many modern game engines run such a system under-the-hood, and specifically in the case of UE, this allows for a functionality extension that is otherwise not possible.

An important factor of scalability is making use of the network topology, namely using the fastest network available on the supercomputer. This means that all services have registered their expected communication via the InfiniBand network interface (Pentakalos, 2002). We refer to literature (Krause, 2019; Alvarez, 2021; Kesselheim *et al.*, 2021) for an in-depth overview of the supercomputing system that we are using. One the Jülich Wizard for European Leadership Science (JUWELS), one cluster compute node has two Intel Xeon Platinum 8168 CPUs. Booster compute nodes additionally have four NVIDIA V100 GPUs. Furthermore, the JUWELS supercomputer has, in addition to its Ethernet network, an InfiniBand high-performing high-throughput network that is entirely unrestricted. The signalling server included in the Synavis framework also contains methods of automatically detecting and using the InfiniBand network. Overhead is introduced by the layers of application management. These layers are the PixelStreaming plugin itself, the transport layer using WebRTC, and also the message passing management by Synavis and its handling on the Python side. This amounts to an average of 0.1 s per message ($\sigma = 7.79\text{E}{-}05, n = 1000$). The low standard deviation shows that the main delays are in fact the traversal of the communication layers. This average is also a total of diverse messages, but the individual categories are not very different, such as for a simple query in $\sim N(0.1003, 8.771\text{E}{-}05, n = 250)$ seconds in contrast to transmitting geometry in $\sim N(0.1004, 8.01\text{E}{-}05, n = 250)$ s.

For complex scenes and geometries, it is also important to understand the underlying processes that happen within UE during scene generation. To showcase UE performance on JUWELS, we ran a measurement of the complete engine tracing (Unreal Insights Tracing) on the JUWELS booster module. The setup for this tracing is simple and can be seen in the listing in Appendix B. The timings of certain UE processes can be seen in Table 2.

Table 1. Timing of CPlantBox generation of plant structures. For this experiment, the plant parameters from Section 3.3 were used. The visualizations were taken into account for the stem and leaf part, as for the field scene, the roots are not visible. Note that these measurement were done using a descriptive callibration of CPlantBox for experimental measurements and thus only capture the structure and morphology of a maize plant as observed in the laboratory. Coupled simulations *will* take longer. CPlantBox simulations were run on 4 CPU nodes with 192 MPI instances total (48 cores each). Timings were wallclock timings before simulation run until after geometrization was completed.

Task	Simulation of one plant	Parallel total	Num cores
1 M plants/7 days	0.015 s $\mu = 0.005$	121.40 s	192
1 M plants/14 days	0.067 s $\mu = 2.6E{-}4$	353.34 s	192
1 M plants/28 days	0.1 s $\mu = 3E{-}4$	535.44 s	192

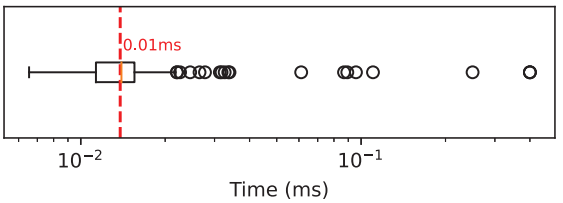

Figure 4. Performance measurement on field scene in Unreal Engine, see Fig. 6. This measurement was done on the JUWELS Booster with about 340 K plants and a total of 80 M triangles. The average frametime is measured for each call of the field drone's update method. There are spikes in the framerate for initialization and geometry loading. The maximum frame time is 0.4 s.

Table 2. Timing statistics of UE runtime on a JUWELS Booster node, total time of experiment was 6 min, and measurements were taken per frame. This table shows the most significant timings that were relevant to the game performance. Lumen Screen Probes are used for dynamic shadowing and global illumination, whereas the scene captures are separate full render passes. Timings were taken using Unreal Insights, an application shipped with Unreal Engine.

Task	Average time per call (s)	Total seconds
Time until rendered image is encoded to video	0.0366	19.3
Rendering of the synthetic data onto the image	0.006	54.3
Raytracing steps for the image composition	0.005	70
Cumulated two-sided rendering of the image	0.018	163
Full run of the application loop	0.052	360

The configuration of this test can be seen Appendix B. UE performs on the JUWELS cluster using the installed compatible graphics interfaces and NVIDIA drivers. The program is packaged in Windows and only uploaded to the cluster. Alternatively, UE packages can also be generated within the cluster environment, using the shared memory as build directory.

For larger fields, plant generation might be an issue. We measured how long the FSPM CPlantBox, callibrated on lab-experiment grown plants, needs to generate one million individual geometries. This includes the loading of the parameters and full FSPM simulation, which also creates a single realization of the stochastic parameter set, as well as our geometry generation module. The measurements can be seen in Table 1 and were performed on the JUWELS CPU module.

3.3. Synavis for augmentation of a Rhizotron experiment

To validate the Synavis framework and showcase that it correctly replicates experimental (Data Source: Repository of CPlantBox) data, we first produced a virtual replica of a rhizotron experiment seen in Fig. 5 and then scaled the CPlantBox model up to field scale in Fig. 6. There is a strong bottleneck in data acquisition for rhizotron experiments as the effort to produce data is very high: Rhizotrons need to be filled, seeded and stored during the measurements and most of the processes must still be done manually. This causes a large overhead for the data acquisition that is not comparable to field data, as field data can be seeded in bulk and the restrictions on how to grow the plants are less limiting. To repeat a rhizotron experiment many times, either a lot of resources are needed to run the experiments in parallel, or an long time is needed for the back-to-back repetitions of the individual experiments. Moreover, it is necessary to achieve the proper environmental conditions for each expermental run. Therefore, often only a very limited number of shoot and root images can be obtained from rhizotron experiments. Since it is of huge interest to have data of shoot and root from the same plant, a precise segmentation of all organs is required. Consequently, labelled image data from rhizotron experiments are both needed and scarce. The scarcity of the data makes training a segmentation network based on this kind of data very hard. As such, it is necessary to implement methods that make optimal use of rhizotron images. To approach these challenges, we design

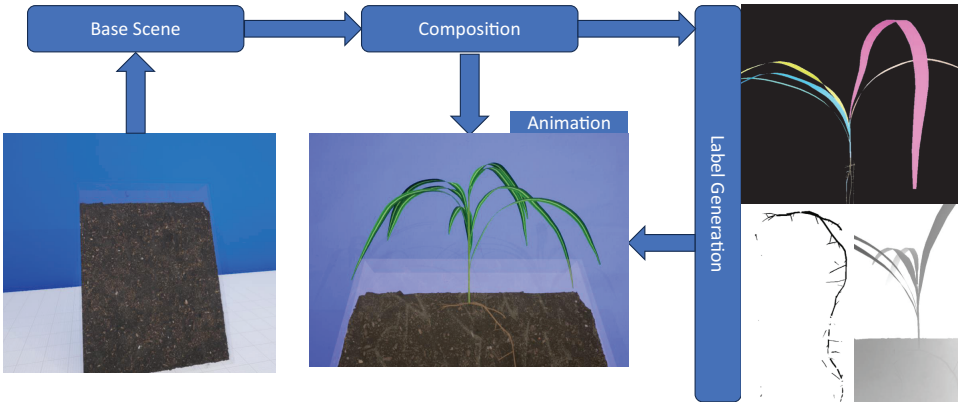

Figure 5. Overview of experiment setup. For this setup, we chose to model the soil within UE by making use of the modelling toolkit. From the material properties, we can change this dynamically similar to how the plant is positioned (middle). Synavis allows us to change properties of the plant as well as what measurements we make, whether pixel-based (right) or quantitative, as presented here by examples of leaf instances, root segmentationand depth mask.

a virtual rhizotron experiment using the FSPM CPlantBox, coupled via Synavis to the UE. In the course of this section, we will highlight how the setup works, how data are produced, how to fetch the data and furthermore, present a variety of test scenarios to use for training. The experiment setup for this example is publicly available and we encourage the tuning and customization of all content we produce.

3.4. Setting up CPlantBox for the laboratory scenario

The rhizotron used as baseline has an interior measurement of $(20 \times 60 \times 2)$ cm in $w \times h \times d$ as well as a distance of 138 cm to the camera. In the UE scene, for the sake of generating synthetic data, some of these measurements might be varied. While the orientation of the camera to the rhizotron matters, as it, of course, must be directed at the object, the measurements are not impactful and, in fact, should be changed for the training. The base setup can be seen in Fig. 5 (left).

However, root growth must happen within the constraints of the rhizotron, which can be done using signed distance functions to limit root growth (Zhou *et al.*, 2020). This restriction helps with the visualization in this instance, but does not detract from the stochasticity of the simulation. The seed position of the plant is put on top of the rhizotron box, and the simulation time is 25 days in steps of 0.5 days.

3.5. Integrating the geometry into UE

Simulation of the CPlantBox plants is done sequentially: individual plant geometries are inserted into UE uniquely and regularly to exchange the data. This is shown in Fig. 5 in the middle, where a plant is rendered within the otherwise empty rhizotron. For this implementation, we automatically add certain textures to the geometry, by using a container class in UE that can separately receive a stem, leaf, and root geometry group. This separation of the organ parts allows the use of different materials for the individual organs. This is not strictly necessary, but it is much easier to separate the individual components both on the Python endpoint as well as in UE.

Geometry generation is handled in the same script as the coupling, as the generation is done in regular intervals. This is a hyperparameter for training: more regular exchanges of the geometry might help in some circumstances. For feature extraction tasks where the orientation of the shoot to the root does not matter, we can further randomize shoot orientation for more images.

We note that the segmentation is done by using an alternate depth rendering pass within UE, which makes sure that we have a separation of the roots from the rest. However, it is also important to only show the root system that is actually visible, which is being taken into account during soil rendering. In Fig. 5, we highlight now this part of the rendering pipeline works by contrasting the root rendering. There is a possibility of estimating the whole biomass or leaf area from a drone perspective, but there is a strong distinction between the tasks *estimate total leaf area* and *segment and calculate visible leaf area*, which depends on the label sets and the measurements of the accompanying real-world data, if applicable.

3.6. Randomizing the laboratory scene

Scene randomization in laboratory settings is challenging, but not impossible. A lot of robustness also stems from changing camera properties. Lens properties, such as shutter speed, ISO factor and aperture, can be configured in UE through Synavis to allow for some more physically based randomization. UE also offers features such as film grain to allow for more augmentation. This is a filter that would introduce noise that does not correspond to what the information rendering is seeing, similar to depth of field. Distance measures through UE can be altered

but will be exact by default. Users might choose to artificially decrease data quality through selection or filtering algorithms, as LiDAR data can be imperfect as well (Zhao *et al.*, 2022). At this point, image-based augmentation usually done on image datasets should still be employed, even on streams. We calibrated leaf surface colours in the direct comparison using average colour measurements.

3.7. Training concepts

In this framework, the active camera in the scene continuously emits an encoded video stream, much like a webcam would. The other data transport method is the data channel—a direct messaging pathway with low latency and minimal overhead. The data channel is also the minimum required communication channel to setup a WebRTC connection.

Fig. 5 on the top-right shows an example of leaf instance segmentation that was handled via the data camera. Note that this is not necessary if the training algorithm also has access to the plant data, from CPlantBox, along with its orientation and position on the map. There, leaf instances on regions of the image can also be inferred directly from the plant data. However, ultimately it depends on how the real-world data are labelled, and the synthetic data should be labeled the same. In this case, pixel labels can be acquired much faster than any other type.

3.8. Upscaling the simulation to field level

Large-scale examples of the Synavis framework are possible, but require a lot of resources. In our case, the best way to utilize the scalable aspects of the framework is to make use of the modular supercomputing architecture. We present a large-scale example of synthetic field data generation using Synavis and the FSPM CPlantBox. The setup is shown in Fig. 6. For the setup of the field, usually a boundary definition is sufficient. The addition of geometries for the soil can be done premeditated in the editor or at runtime through Synavis. When generating plants using CPlantBox dynamically, we refer to the measurements of the associated processes in Section 3.2.

Randomization between compute nodes is done using JSON descriptions that are generated based on local rank, as exemplified in Fig. 6. This can be included in the runscript, or explicitly in the python environment. Generally, it is possible to have both varying (SynavisUE Lighting Management or Dynamic Volumentric Sky Lighting Management) as well as static-property scenes for comparison between compute nodes of model robustness. This is especially shown in Fig. 6.

The setup is aimed at producing high stochastic variability while allowing for cross-comparison between the individual data extractions. As such, we utilize a large field that is centrally populated by the FSPM with plant geometries. UE uses primarily GPU resources, but will also have a lot of CPU work to do, as the virtual world has to be simulated and UE needs to respond to inputs through the Synavis framework. As such, UE can run either on the booster module or the visualization module of a supercomputer. This is only true for booster modules that allow rendering, such as JUWELS. Training frameworks and extraction tasks are best run on the booster module and connected to UE via a common network interface or through tunnelling. The CPlantBox generation is best run on pure CPU nodes, as the generation of plant models can be run in parallel or, in the case of coupled simulations, would use too many resources to be used together with UE on the same node. In our approach, we do not compute the CPlantBox geometries locally on the UE node. One bottleneck, however, is that the spawning of geometries occurs in batches for each frame, resulting in a warm-up time until the scene is loaded and present.

4. DISCUSSION

We developed a novel framework for a direct and scalable coupling of an FSPM visualization with UE. This framework, Synavis and its plugins SynavisUE, are primarily designed to allow for a loose coupling of data generation with synthetic data training on HPC systems. The framework is designed to allow the transport of FSPMs to UE, while also allowing easy access to objects in the scene as well as their properties. Synavis automatically conveys information from UE and is extremely suitable for people who develop their own applications in UE, especially building on top of existing projects.

4.1. Analysis of the synthetic data

Synthetic data must, most of all, convey the information and retain the measurable properties that the experimental data also exhibits. To analyse the performance of SynavisUE in the example use-case, the rhizotron experiment, we ran the data through the extraction pipeline that is also used for the model calibration on rhizotron experiments (Bauer *et al.*, 2023). Using the skeletonization and subsequent topological analysis, we extracted leaf blade length measures from UE as well as the experiment, as seen in Fig. 7.

We want to highlight that, in general, it is more beneficial to create diversity rather than exact replicas. Nonetheless, the synthetic data should be targeted at the use-case. Scalability can be targeted while training problems that are closely related to the target measurements. We see in Fig. 7 that the synthetic measurements are lower, in general, with a higher standard deviation. Lower measurements might be caused by the flatness of the leafs. The comparison of the synthetically generated data and the experimental data in Fig. 7 right shows that relevant plant measures are replicated and are not distorted by the synthetic data generation pipeline.

4.2. Comparison to other approaches

For comparison, we highlight the use of UnrealCV (Qiu *et al.*, 2017) in science. An example of this is the data set generation for UnrealStereo by Zhang *et al.* (2018). Conceptually, UnrealCV is not dissimilar to SynavisUE, with the distinction that SynavisUE is based on PixelStreaming and loose coupling rather than a direct interaction. Furthermore, we put more emphasis on real-time data generation than writing data sets. The loose coupling on our end also enables us to separate technologies and make use of a system where interaction of the DL models with the scene mimic a robotics environment with all its latencies, as the world continues to simulate and does not wait for input. Moreover, we

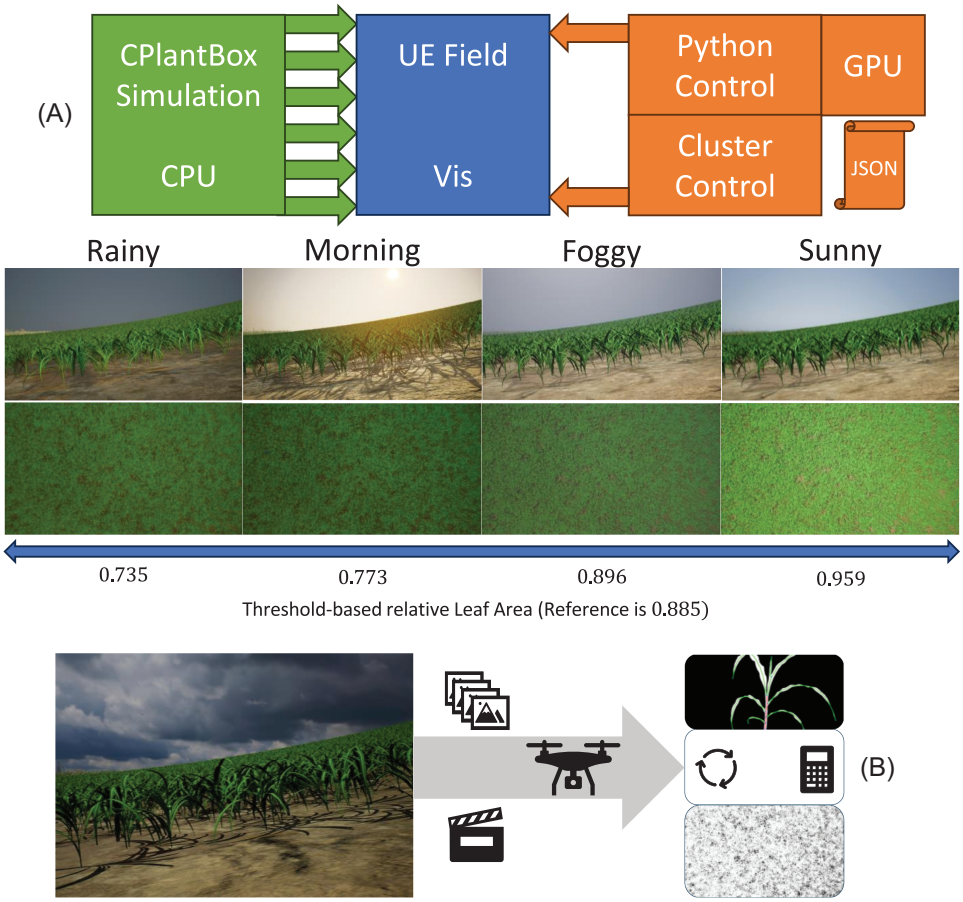

Figure 6. (A) Overview of the field data generation pipeline. We also show scene property change can lead to significant changes in acquired data. The setup for a field visualization with a cluster is used to assess robustness of models against influences of the image. In this example, the field is not randomized to allow for a comparable result in all compute nodes. Influences on the image can be numerous, and here we primarily showcase the change of weather that impacts image rendering. For more direct management, the SynavisUE plugin also contains methods of quickly altering lighting effects. Since all assets have UE accessible properties, the JSON description for the rank automatically manages the condition. This can be done preemptively using a JSON file, or through Synavis dynamically. The prime advantage of using the pipeline is versatility, as measurements as well as data labels can be generated very easily. The field scene is especially impactful for a number of aspects, such as conditional instance segmentation by using prior knowledge on which parameter set was used for the plant generation. Rule for calculating the relative leaf area is extracting by channel and then by threshold.

expand on the concept of synthetic data generation and present a whole-plant rendering approach together with data generation in UE as well as a coupling to training algorithms.

We especially highlight the embedding of Synavis into a modular supercomputing system (Suarez *et al.*, 2019). Both JUWELS (Alvarez, 2021; Kesselheim *et al.*, 2021; Krause, 2019) and JURECA supercomputers (Krause and Thörnig, 2018; Thörnig, 2021) are modular supercomputers. Newer systems, such as LUMI, further developed this concept to focus their accelerator systems on general compute-based programming paradigms (Markomanolis *et al.*, 2022). For these systems, it is more optimal to separate the visualization and training component onto different modules, which for modern tensor core GPUs is even necessary as they do not provide rendering infrastructure. We designed Synavis with this concept in mind.

Moreover, we would like to make the important distinction between the generation of a virtual environment and algorithms such as by Gao *et al.* (2023). These approaches make the images

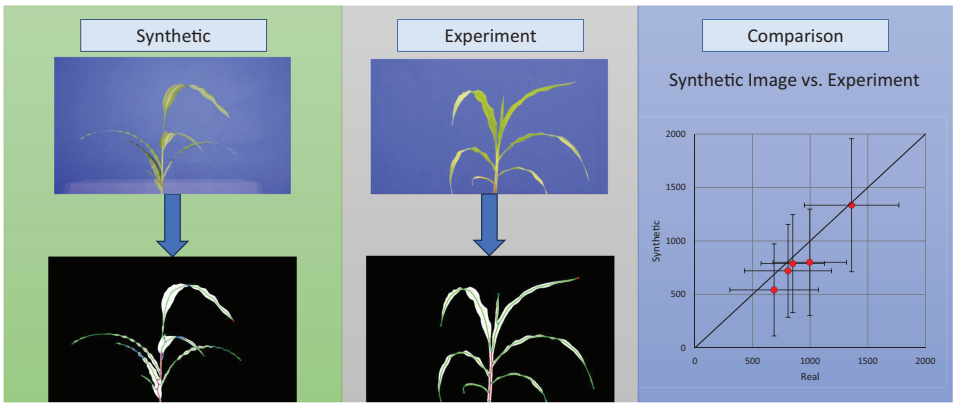

Figure 7. Comparison of parameter extraction pipeline between synthetic and real-world data. Real-world data, a total of 23 plant images at this growth stage and angle, were acquired in controlled rhizotron experiments. Bottom: Analyzed skeletons of shoot organs, starting with the pseudo-stem. Right: Comparison of blade lengths in mm, compared across samples sorted by longest first. The error bars indicate the standard deviation on each axis.

more realistic and the training more robust, but ultimately flatten the virtually generated environment to images by augmenting them unpredictably. These approaches have also been reported to be unable to retain the geometric qualities of the virtual scene as described as CycleGAN failure cases by Hartley *et al.* (2021) and Zhu *et al.* (2017). SynavisUE is a framework that lets training models interact with the virtual scene by steering and manipulation. Furthermore, our scalability setup utilizes the fact that manipulating settings in the scene creates augmented images whose main advantage is the fact that one can interpolate settings between a success and a failure case while always retaining a coherent and well-annotated image.

4.3. Synavis: possibilities and limitations

Synthetic data are not the final training step, if fine-tuning has to be performed. We believe that the main strength in synthetic data training is its scalability rather than exactness.

Transfer learning approaches such as ImageNet (Deng *et al.*, 2009) based encoders have exhibited good performance in many use-cases, as analysed by Huh *et al.* (2016) and Morid *et al.* (2021). The reason behind this is that a lot of tasks in computer vision are re-usable and generalize well. This concerns individual feature maps as well as activation weights that are transferable. One example of this is presented by Chen *et al.* (2020), who analyse transfer learning performance in image-based plant disease detection.

Synavis and SynavisUE are most comfortably usable between a pre-trained network and the final tuning. Synthetic data provide a lot of variety when given proper stochasticity commands, and its use in training DL models is potentially very impactful. Synavis is created with strong data augmentation in mind, as scene properties can be changed at runtime, changing the visual properties of the data.

WebRTC is a video real-time communication experience standard. For our user-less framework, this means that the video information might not be in lossless format. On HPC machines, we usually circumvent this by scaling up the image size and downsample on the DL side. Formats like H264 (ISO/IEC 14496-10:2022, 2022) are very light in network usage, as shown by Van der Auwera *et al.* (2008), but might impact the customizability of the framework because the numbers of encoding sessions are limited, or by the aspect ratio of supported image formats.

4.4. Outlook

In this article, we showed the promise and practical use of our coupling framework, together with a visualization of the FSPM CPlantBox. Visualization of more morphological features is needed in the future to push the limits of the visualization. We also aim for calibration measurements such as light metring in comparison to UE lights, as currently we can make only relative but not absolute statements about how plants would behave in the simulated scenes.

WebRTC is inherently interactive, which means that data are sent both ways (Jennings *et al.*, 2021). We will build on this concept further by allowing the steering of the data generation during training.

There are possibilities in advancing algorithm robustness that are both depended on scene variety as well as training strategies. We will explore both those options in the near future.

Incorporation of different data sources is also possible within Synavis, but more fine-tuning should be done to arrive at a successful scene visualization.

5. CONTRIBUTIONS BY THE AUTHORS

D.H. has implemented the FSPM visualization, the coupling framework, the Unreal Engine implementation and authored this paper primarily. F.B. has contributed to the scientific workflow, the calibration of the FSPM model to plant data, the

implementation of the analysis pipeline used in this paper and contributed input about field measurements. He has contributed to the text of this paper. A.S. supervises the CPlantBox FSPM development, has contributed to the text of this paper, and has given input on scientific workflows. M.G. has implemented large parts of features and analysis pipelines used within the FSPM used in this paper. She has implemented calibration workflows for the FSPM and contributed scientific counsel to this paper. She has contributed to the text of this paper. J.H.G. has contributed to the text of this paper, implemented functions and methodology used within the HPC environment, is responsible for the implementation of new visualization packages and technologies on JURECA and JUWELS supercomputers and has contributed scientific counsel to this paper. E.H. has contributed to the text of this paper, given scientific counsel, access to the LUMI supercomputer for testing and is supervisor to this project for the University of Iceland. H.S. has contributed to the text of this paper, given scientific counsel and is supervisor to this project for the University of Iceland. M.R. has contributed to the text of this paper, given scientific counsel, provided access to compute systems and information about methodology. He is primary supervisor of this project.

CONFLICT OF INTEREST STATEMENT

None declared.

FUNDING

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 bio-economy" (Rhizo4Bio), subproject CROP (ref. FKZ 031B0909A).

MODEL AND DATA AVAILABILITY

The code is open source and available under the Synavis and SynavisUE repositories with an example avaliable under SynavisUEexample. The CPlantBox official code can be found at on the institute's GitHub page. The branch associated with this article has been forked to this page.

We have prepared introduction videos to the subjects. An overview of the framework and pipelines that are possible can be found here:
Baker, Dirk Norbert (2023). Supplemental Video for our Manuscript ISPLANTS-2023-026. figshare. Media. https://doi.org/10.6084/m9.figshare.24179136.v2
A step-by-step guide can be found here:
Baker, Dirk Norbert (2023). Step by Step guide to setup Synavis with CPlantBox and Unreal Engine. figshare. Media. https://doi.org/10.6084/m9.figshare.24182592.v1

LITERATURE CITED

Ahmadi M, Gross WJ, Kadoury S, 2016. A real-time remote video streaming platform for ultrasound imaging. *Proceedings of the 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. doi:10.1109/EMBC.2016.7591698.

Alvarez D. 2021. JUWELS cluster and booster: exascale pathfinder with modular supercomputing architecture at Jülich Supercomputing Centre. *Journal of Large-Scale Research Facilities*, 7:1–14. doi:10.17815/jlsrf-7-183.

Bailey BN. 2019. Helios: a scalable 3D plant and environmental biophysical modeling framework. *Frontiers in Plant Science* 10. doi:10.3389/fpls.2019.01185.

Bauer FM, Lobet G, Helmrich DN, Galinski A, Kahlilova Z, Zaner L, Kuczkowska M, Yu P, Dörmann P, Schaaf G, Schnepf A. 2023. In silico investigation on phosphorus efficiency of zea mays: an experimental whole plant model parametrization approach. *International Conference on Functional-Structural Plant Models, FSPM2023*, Berlin, Germany, March 2023, 86–87. doi:10.34734/FZJ-2023-04031.

Behroozpour B, Sandborn PAM, Wu MC, Boser BE. 2017. LiDAR system architectures and circuits. *IEEE Communications Magazine* 55:135–142. doi:10.1109/MCOM.2017.1700030.

Benoit L, Rousseau D, Belin É, Demilly D, Chapeau-Blondeau F. 2014. Simulation of image acquisition in machine vision dedicated to seedling elongation to validate image processing root segmentation algorithms. *Computers and Electronics in Agriculture* 104:84–92. doi:10.1016/j.compag.2014.04.001.

Bondi E, Dey D, Kapoor A, Piavis J, Shah S, Fang F, Dilkina B, Hannaford R, Iyer A, Joppa L, Tambe M. 2018. AirSim-W: a simulation environment for wildlife conservation with UAVs. In: *COMPASS '18*. Association for Computing Machinery, New York, NY. doi:10.1145/3209811.3209880.

Bouvry A, Lebeau F. 2023. Digital twin of a smart plant factory for plant phenotyping: data assimilation between measured and simulated 3D point cloud data in the CPlantBox FSPM. *International Conference on Functional-Structural Plant Models, FSPM2023*, Berlin, Germany.

Chen J, Chen J, Zhang D, Sun Y, Nanehkaran Y. 2020. Using deep transfer learning for image-based plant disease identification. *Computers and Electronics in Agriculture* 173:105393. doi:10.1016/j.compag.2020.105393.

Deng J, Dong W, Socher R, Li L-J, Li K, Fei-Fei L. 2009. ImageNET: a large-scale hierarchical image database. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*, Miami, FL, June 20–25, 2009. doi:10.1109/CVPR.2009.5206848.

Gao Y, Li Y, Jiang R, Zhan X, Lu H, Guo W, Yang W, Ding Y, Liu S. 2023. Enhancing green fraction estimation in rice and wheat crops: a self-supervised deep learning semantic segmentation approach. *Plant Phenomics* 5:0064. doi:10.34133/plantphenomics.0064.

Giraud M, Gall SL, Harings M, Javaux M, Leitner D, Meunier F, Rothfuss Y, van Dusschoten D, Vanderborght J, Vereecken H, Lobet G, Schnepf A. 2023. Development and calibration of the FSPM CPlantBox to represent the interactions between water and carbon fluxes in the soil-plant-atmosphere continuum. doi:10.1093/insilicoplants/diad009.

Hartley ZKJ, French AP. 2021. Domain adaptation of synthetic images for wheat head detection. *Plants* 10. doi:10.3390/plants10122633.

Hartley ZKJ, Jackson AS, Pound M, French AP. 2021. GANana: unsupervised domain adaptation for volumetric regression of fruit. *Plant Phenomics* 2021. doi:10.34133/2021/9874597.

Hughes JF, Van Dam A, McGuire M, Sklar DF, Foley JD, Feiner SK, Akeley K. 2014. Introduction to fixed-function 3d graphics and hierarchical modeling. *Computer graphics, principles and practice*, 117–148. Boston, MA: Addison-Wesley Longman Publishing Co. ISBN: 978-0-201-84840-3.

Huh M, Agrawal P, Efros AA 2016. What makes ImageNet good for transfer learning? In: *2016 IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, NV, USA, June 27–30, 2016. IEEE Computer Society 2016, ISBN 978-1-4673-8851-1. doi:10.48550/arXiv.1608.08614.

ISO/IEC 14496-10:2022. 2022. H.264 : advanced video coding for generic audiovisual services. Standard, International Telecommunication Union, Geneva. CH. ID: T-REC-H.264-202108-I.

ITU-T H.265. 2023. H.265 : high efficiency video coding. Standard, International Telecommunication Union, Geneva. CH. ID: T-REC-H.265-202309-P.

Jansen W, Huebel N, Steckel J. 2022. Physical LiDAR simulation in real-time engine. In: *2022 IEEE Sensors*. doi:10.1109/SENSORS52175.2022.9967197.

Jennings C, Castelli F, Boström H, Bruaroey J-I. 2021. WebRTC 1.0: real-time communication between browsers. In: *Recommendation of the World Wide Web Consortium*. ID: REC-webrtc-20230306.

Jitsev J. 2021. Impact of large-scale pre-training on intra- and inter-domain transfer learning in full and few-shot regimes. In: *Workshop Machine Learning on HPC Systems, International Supercomputing Conference*, Frankfurt, Germany.

Kamilaris A., Prenafeta-Boldú F.X. 2018. Deep learning in agriculture: a survey. *Computers and Electronics in Agriculture*, 147:70–90. doi:10.1016/j.compag.2018.02.016.

Karis B, Stubbe R, Wihlidal G. 2021. A deep dive into Nanite Virtualized Geometry. In: *SIGGRAPH* 2021.

Kesselheim S, Herten A, Krajsek K, Ebert J, Jitsev J, Cherti M, Langguth M, Gong B, Stadtler S, Mozaffari A, Cavallaro G, Sedona R, Schug A, Strube A, Kamath R, Schultz MG, Riedel M, Lippert T. 2021. JUWELS booster—a supercomputer for large-scale AI research. In: *High Performance Computing*, Springer International Publishing, Cham. doi:10.48550/arXiv.2108.11976.

Kim D, Kang WH, Hwang I, Kim J, Kim JH, Park KS, Son JE. 2020. Use of structurally-accurate 3D plant models for estimating light interception and photosynthesis of sweet pepper (*Capsicum annuum*) plants. *Computers and Electronics in Agriculture* 177:105689. doi:10.1016/j.compag.2020.105689.

Krause D. 2019. JUWELS: modular tier-0/1 supercomputer at the Jülich Supercomputing Centre. *Journal of Large-Scale Research Facilities*, 5. doi:10.17815/jlsrf-5-171.

Krause D, Thörnig P. 2018. JURECA: modular supercomputer at Jülich Supercomputing Centre. *Journal of Large-scale Research Facilities*, 4. doi:10.17815/jlsrf-4-121-1.

Kuznichov D, Zvirin A, Honen Y, Kimmel R. 2019. Data augmentation for leaf segmentation and counting tasks in rosette plants. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2580–2589.

Lobet G, Draye X, Périlleux C. 2013. An online database for plant image analysis software tools. *Plant Methods* 9:38. doi:10.1186/1746-4811-9-38.

Lobet G, Koevoets IT, Noll M, Meyer PE, Tocquin P, Pagès L, Périlleux C. 2017. Using a structural root system model to evaluate and improve the accuracy of root image analysis pipelines. *Frontiers in Plant Science* 8. doi:10.3389/fpls.2017.00447.

Markomanolis GS, Alpay A, Young J, Klemm M, Malaya N, Esposito A, Heikonen J, Bastrakov S, Debus A, Kluge T, Steiniger K, Stephan J, Widera R, Bussmann M. 2022. Evaluating GPU programming models for the LUMI supercomputer. In: *Asian Conference on Supercomputing Frontiers, Singapore, Singapore*. New York City, New York, USA: Springer International Publishing. doi:10.1007/978-3-031-10419-0_6.

Masson AL, Caraglio Y, Nicolini E, Borianne P, Barczi JF. 2021. Modelling the functional dependency between root and shoot compartments to predict the impact of the environment on the architecture of the whole plant: methodology for model fitting on simulated data using deep learning techniques. *In silico Plants* 4:diab036. doi:10.1093/insilicoplants/diab036.

McCormac J, Handa A, Leutenegger S, Davison AJ. 2017. 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)*. doi:10.1109/ICCV.2017.292.

Morandage S, Laloy E, Schnepf A, Vereecken H, Vanderborght J. 2021. Bayesian inference of root architectural model parameters from synthetic field data. *Plant and Soil* 467:67–89. doi:10.1007/s11104-021-05026-4.

Morid MA, Borjali A, Del Fiol G. 2021. A scoping review of transfer learning research on medical image analysis using ImageNet. *Computers in Biology and Medicine* 128:104115. doi:10.1016/j.compbiomed.2020.104115.

Nagel KA, Putz A, Gilmer F, Heinz K, Fischbach A, Pfeifer J, Faget M, Blossfeld S, Ernst M, Dimaki C, Kastenholz B, Kleinert A-K, Galinski A, Scharr H, Fiorani F, Schurr U. 2012. GROWSCREEN-Rhizo is a novel phenotyping robot enabling simultaneous measurements of root and shoot growth for plants grown in soil-filled rhizotrons. *Functional Plant Biology* 39:891–904. doi:10.1071/fp12023.

Nimmi S, Saranya V, Theerthadas, Gandhiraj, R. 2014. Real-time video streaming using GStreamer in GNU radio platform. In: *2014 International Conference on Green Computing Communication and Electrical Engineering (ICGCCEE)*. doi:10.1109/ICGCCEE.2014.6922233.

Pentakalos OI. 2002. An introduction to the infiniband architecture. In: *High performance mass storage and parallel I/O: technologies and applications*. doi:10.1109/9780470544839.ch42.

Perez RPA, Vezy R, Brancheriau L, Boudon F, Grand F, Ramel, M, Artanto Raharjo D, Caliman JP, Dauzat J. 2022. When architectural plasticity fails to counter the light competition imposed by planting design: an in silico approach using a functional–structural model of oil palm. *In silico Plants*, 4. doi:10.1093/insilicoplants/diac009.

Pollok T, Junglas L, Ruf B, Schumann A. 2019. UnrealGT: using unreal engine to generate ground truth datasets. *Advances in Visual Computing*. doi:10.1007/978-3-030-33720-9_52.

Pound MP, Atkinson JA, Townsend AJ, Wilson MH, Griffiths M, Jackson AS, Bulat A, Tzimiropoulos G, Wells DM, Murchie EH, Pridmore TP, French AP. 2017. Deep machine learning provides state-of-the-art performance in image-based plant phenotyping. *GigaScience*, 6. doi:10.1093/gigascience/gix083.

Qiu W, Zhong F, Zhang Y, Qiao S, Xiao Z, Kim TS, Wang Y. 2017. UnrealCV: virtual worlds for computer vision. In: *Proceedings of the 25th ACM International Conference on Multimedia*, 1221–1224. doi:10.1145/3123266.3129396.

Reddy T, Johnston A, Matthews P, Rosenberg J. 2020. Traversal using relays around NAT (TURN): relay extensions to session traversal utilities for NAT (STUN). *Proposed Standard of the Internet Engineering Task Force (IETF)*. doi:10.17487/RFC8656.

Sanders A. 2016. *An introduction to Unreal engine 4*. London: CRC Press. ISBN 978-1138427440.

Scharr H, Minervini M, French AP, Klukas C, Kramer DM, Liu X, Luengo I, Pape JM, Polder G, Vukadinovic D, et al. 2016. Leaf segmentation in plant phenotyping: a collation study. *Machine vision and applications* 27:585–606. doi:10.1007/s00138-015-0737-3.

Scharr H, Tsaftaris SA. 2022. Meeting computer vision and machine learning challenges in crop phenotyping. *Advances in plant phenotyping for more sustainable crop production*. Burleigh Dodds Science Publishing. doi:10.19103/AS.2022.0102.11.

Schulzrinne H, Casner S, Frederick R, Jacobson V. 2003. RTP: a transport protocol for real-time applications. *Standard of the Internet Engineering Task Force*. doi:10.17487/RFC3550.

Soualiou S, Wang Z, Sun W, de Reffye P, Collins B, Louarn G, Song Y. 2021. Functional–structural plant models mission in advancing crop science: opportunities and prospects. *Frontiers in Plant Science* 12. doi:10.3389/fpls.2021.747142.

Suarez E, Eicker N, Lippert T. 2019. Modular supercomputing architecture: from idea to production. In: Jeffrey S Vetter (ed), *Contemporary high performance computing*, 223–255. CRC Press. London, UK: Chapmann & Hall. ID: FZJ-2019-03055

Thörnig P. 2021. JURECA: data centric and booster modules implementing the modular supercomputing architecture at Jülich Supercomputing Centre. *Journal of Large-scale Research Facilities*. doi:10.17815/jlsrf-7-182.

Tsaftaris SA, Minervini M, Scharr H. 2016. Machine learning for plant phenotyping needs image processing. *Trends in Plant Science* 21: 989–991. doi:10.1016/j.tplants.2016.10.002.

Ubbens J, Cieslak M, Prusinkiewicz P, Stavness I. 2018. The use of plant models in deep learning: an application to leaf counting in rosette plants. *Plant Methods* 14. doi:10.1186/s13007-018-0273-z.

Van der Auwera G, David PT, Reisslein M. 2008. Traffic characteristics of H264/AVC variable bit rate video. *IEEE Communications Magazine* 46:164–174. doi:10.1109/MCOM.2008.4689260.

Wang L, Wang W, Dorsey J, Yang X, Guo B, Shum HY. 2006. Real-time rendering of plant leaves. In: *ACM SIGGRAPH 2006 Courses*, Boston, Massachusetts, United States of America. ACM (Association for Computing Machinery). doi:10.1145/1185657.1185725.

Ward D, Moghadam P, Hudson N. 2018. Deep leaf segmentation using synthetic data. In: *Proceedings of the British Machine Vision Conference (BMVC)*, British Machine Vision Association Newcastle, UK, Workshop on Computer Vision Problems in Plant Phenotyping (CVPPP). doi:10.48550/arXiv.1807.10931.

Yang W, Feng H, Zhang X, Zhang J, Doonan JH, Batchelor WD, Xiong L, Yan J. 2020. Crop phenomics and high-throughput phenotyping: past decades, current challenges, and future perspectives. *Molecular Plant* 13:187–214. doi:10.1016/j.molp.2020.01.008.

Yun T, Cao L, An F, Chen B, Xue L, Li W, Pincebourde S, Smith MJ, Eichhorn MP. 2019. Simulation of multi-platform lidar for assessing total leaf area in tree crowns. *Agricultural and Forest Meteorology* 276–277:107610. doi:10.1016/j.agrformet.2019.06.009.

Zhang T, Xie L, Wei L, Zhuang Z, Zhang Y, Li B, Tian Q. 2020. UnrealPerson: an adaptive pipeline towards costless person re-identification. In: *2020 IEEE Conference on Computer Vision and Pattern Recognition*. doi:10.1109/CVPR46437.2021.01134.

Zhang Y, Qiu W, Chen Q, Hu X, Yuille A. 2018. UnrealStereo: controlling hazardous factors to analyze stereo vision. In: *International Conference on 3D Vision (3DV)*, Verona, Italy, hosted by University of Verone. doi:10.48550/arXiv.1612.04647.

Zhao X, Su Y, Hu T, Cao M, Liu X, Yang Q, Guan H, Liu L, Guo, Q. 2022. Analysis of UAV lidar information loss and its influence on the estimation accuracy of structural and functional traits in a meadow steppe. *Ecological Indicators* 135:108515. doi:10.1016/j.ecolind.2021.108515.

Zhou X-R, Schnepf A, Vanderborght J, Leitner D, Lacointe A, Vereecken H, Lobet G. 2020. CPlantBox, a whole-plant modelling framework for the simulation of water- and carbon-related processes. *In silico Plants* 2. doi:10.1093/insilicoplants/diaa001.

Zhu J-Y, Park T, Isola P, Efros AA. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. In: *2017 IEEE International Conference on Computer Vision (ICCV)*, Venice, Italy, IEEE. doi:10.1109/ICCV.2017.244.

APPENDICES

A. VARIATION OF VIRTUAL SCENES

Table A1. Selection of scene variation possibilities to improve larger-scale data augmentation using Unreal Engine and Synavis. The list is not exhaustive, but should give a good impression that stochastic augmentation should be applied wherever possible.

Object	Property	Description
Sun	Position	2 DoF $\phi, \eta \in [0, \pi]$
Sun	Strength	[Lumen]
Sun	Ambient light	Intensity [Lumen]
Field	Plant position	Varying position 2 DoF, dependent on scene and placing, interior experiment scenes have less potential variability here
Field	Plant density	Handled via position, this would be a good way of targeting different DL models that also react sensitively towards changes in complexity
Plant	Plant age	determined by parametrization, usually within $T \in [0, 30]$
Plant	Plant leaf bending	Partly visual feature—leaf bending is influenced by environmental conditions. When rendering a wet scene, leaf bending should be increased. For accuracy, it should be determined how large the impact of additional weight is on the leaf structure. If it can be assured that the leaf bending is not taken into account to solve the target problem, this can be estimated.
Room	Room lights and diffusion	For digitized experiments, lights should be calibrated to actual brightness values as stated by the bulb manufacturer.
Room	Room wall material	RGB texture $X \times Y \times R \times B \times G$, Specularity, Roughness
Camera	Camera lens properties	A collection of scalars describing the cameras lense and depth-of-field effects in UE
Camera	Camera position	Camera paths throughout the scene might further reveal critical insight into algorithm performance in specific and extraordinary circumstances.
Post processing	Colour and screen effects	Film grain, movement effects, and more effects can be introduced using Post Process Materials. Not that encoding effects would be captures using the style of encoding and if finetuning towards those effects should be done, the reference should produce effects that might be affected by the encoding the strongest.

B. UNREAL ENGINE CLUSTER CONFIGURATION

```
m = syn.MediaReceiver()
...
tracefile = "trace_" + str(int(time.time())) + ".utrace"
m.SendJSON({"type":"console", "command":"trace.File "+ tracefile})
start_time = time.time()
runtime = 360
while time.time() < start_time + runtime :
  time.sleep(0.05)
  pos = np.random.rand(3) * 100 + np.array([0, 0, 0])
  size = np.random.rand() * 0.9 + 0.1
  v, i, n = cube_geometry(size, pos)
  v = v.flatten(); i = i.flatten(); n = n.flatten()
  m.SendGeometry(v, i, "cube", n, None, None, True)
m.SendJSON({"type":"console", "command":"trace.stop"})
m.SendJSON({"type":"console", "command":"quit"})
```

Listing B.1. Setup for the scalability test performed with an increasing number of cube geometries within the scene. These were randomly spawned through Synavis.

Table B1. Configuration for unreal engine runtime on JUWELS Booster.

Property	Value	Info
Encoder	H264	GPU-Encoding
Resolution	3860 × 2160	4k UHD
Keyframe interval	1	Triggers transmission of the full image every second
Scene capture virtual shadow map size	4096	This allows the scene capture components, the InfoCam and SceneCam of the Synavis Drone, to have larger shadow maps than in a classical setting (default 512)
Render offscreen	true	Needed for cluster headless mode
Max GPU count	4	Allows unreal to make use of NVLink
PixelStreaming degradation preference	Maintain quality	Prevents UE from subsampling the encoding

C. MEDIA RECEIVER IN SYNAVIS

```
import Synavis as sv

media = sv.MediaReceiver()
media.Initialize()
media.SetConfig({"SignallingIP":"625.174.856.321", \
    "SignallingPort": 8080})
media.SetTakeFirstStep(False)
media.StartSignalling()
media.SetMessageCallback( \
    lambda message : message_buffer.append(message) \
)

while media.GetState() != sv.EConnectionState.CONNECTED :
    time.sleep(0.1)
```

Listing C.1. Simple coupling with media receiver.

D. OVERVIEW OF PIPELINE COMPONENTS

Table D1. Short overview of all pipeline component mentioned in this article, their function, and where to find out more and/or download them.

Pipeline component	Function	Futher information
CPlantBox	• Generation of plant model • Provision of geometry	see Giraud *et al.* (2023), https://github.com/Plant-Root-Soil-Interactions-Modelling/CPlantBox
Unreal Engine	• Visualization of the plant in a virtual scene • Scene variation • Streaming of images	www.unrealengine.com
Synavis UE	• Reception and integration with PixelStreaming • Dynamic command handling • Integration with UE system	gh/dhelmrich/synavisue
Synavis	• Coupling between frameworks/software • Connection handling and signalling server • JSON format parsing and python binding	gh/dhelmrich/synavis
OpenCV/Pytorch	• Reception of images • Random image augmentation • Data analysis	pytorch.org, opencv.org

E. GEOMETRIZATION

Generally, for a complete geometry as seen in Fig. E1, we compute vertices $V \subset \mathbb{R}^3$, vertex-assigned normal vectors $N : V \ni x \mapsto n(x) \in \mathbb{R}^3$, triangles (with an ordered set V): $T \subset V \times V \times V$, as well as texture coordinates $A : V \ni x \mapsto a(x) \in \mathbb{R}^2$ and tangents. For a full introduction to geometry modelling, we refer to Hughes *et al.* (2014).

We extended CPlantBox with a geometrization pipeline based on its graph formalism and common assumptions. In this section, we describe how the plant data are used to create the plant morphology. Moreover, some characteristics of the plant morphology are not simulated by CPlantBox. Default geometrization schemes have, therefore, been setup while keeping the resulting geometry mainly sensitive to the CPlantBox outputs, as seen in Fig. 1 and further in Fig. E1.

Stem geometrization is based on generating a continuous space of ordered circles in a 3D-space that is connected by triangles. Their orientation has to be inferred from the graph structure, which means that their rotation Quaternion $q^{o,i,i-1} \in \mathbb{H}$ is defined by the angle between two consecutive segments $i-1, i$ on the organ o. The rotation operator is defined as $\mathrm{Rot}(q, x) := q \cdot (0, x) \cdot q^{-1}$ where $q \in \mathbb{H}$ and $x \in \mathbb{R}^3$. To prevent ambiguity of the local geometry, we generate the circle in *local space* $P_L^{o,i}$ and transfer it to *world space* by using the new direction and the axis of the previous segment. The transformation from local to world space can be described as

$$P_W^{(o,i)} = d^{(o,i)} \cdot \mathrm{Rot}\left(P_L^{(o,i)}, q^{(o,i,i-1)}\right) + C^{(o,i)} \tag{E1}$$

where P_W and P_L are the positions in world and local space, respectively, d is the local diameter, q is the world orientation quaternion and C is the position of the segment i on organ o. We compute a quaternion by the vector of two subsequent points i and $i+1$ as well as the last coordinate system vectors u (forward), v (right) and w (up). When using splines, that is, smooth curves in 3D space, to interpolate points, we can also compute rotation using the curve derivative. Within our rotation space $\mathbb{H}$, we can use spherical interpolation by applying $q_\alpha(q_0, q_1, \alpha) := q_1 \cdot (q_1^{-1} \cdot q_2)^\alpha$ with $\alpha \in [0, 1]$ being the interpolation factor, which we use for angular smoothing. Triangles are computed by connecting subsequent pairs of vertices with their predecessors. To update the coordinate system with a new forward vector $u^{(i)}$, we compute the new up vector $w^{(i)} = v^{(i-1)} \times u^{(i)}$ and the new right vector $v^{(i)} = w^{(i)} \times u^{(i)}$.

For leaf geometrization, we interpolate the graph structure of the underlying FSPM by using a series of splines. For the leaf structure specifically, we employ two distinct techniques to describe its surface: Linear and radial description. A linear leaf description is a geometrization that assigned each point p_i on the midline of the leaf a distance to the edge of the leaf blade in each direction. In contrast, radial geometrization is a kind of template that describes a shape by the distances d_i at specific angles φ_i from a centre point of the leaf, such as the end of the petiole. For the leaf structure, we are rendering the leaf as two-sided flat surface. This means that all leaf variants are rendered into a series of triangles spanning the top and bottom part of the plant. Within UE, we can also simulate this by not culling the backface triangles of the geometry, and instead rendering the backfaces like frontfaces.

Additional structural features of the plant geometry can be inferred from texture as well: UE can render plants with additional world position offset, which changes where the vertices are rendered in the scene, regarding a displacement vector given in the material shader. This is especially useful when simulating wind, especially in conjunction with our texture warping technique as we can prescribe forces at specific positions relative to the leaf surface. Masked leafs might behave somewhat unpredictably with this addition, as it is not always clear where the first leaf point connects to the stem.

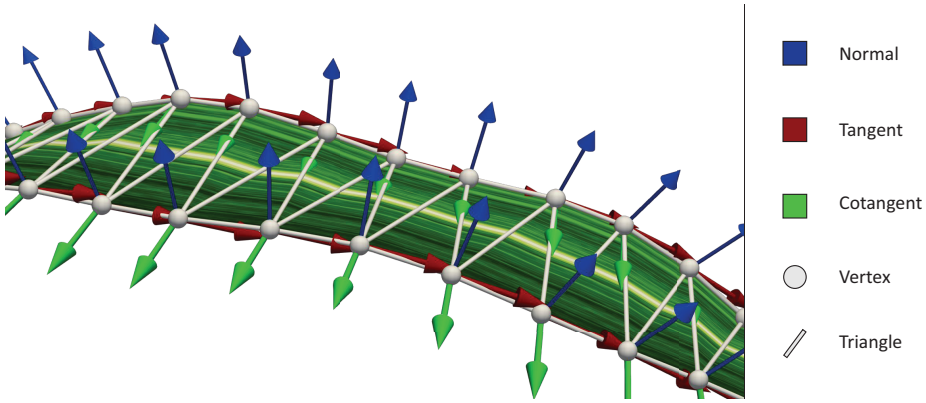

Figure E1. Leaf-level geometry, applied texture onto texture coordinate map, as well as local coordinate spaces. Texture image axes correspond to tangent directions. A local coordinate system is essential for a robust geometrization of the leaf; otherwise instabilities cause significant visual artifacts. Local coordinates are not always guaranteed to be right-handed but uniform across the leaf surface. The surface normal is also the local z-axis, while the surface tangent in the direction of the leaf is the local x axis while the cotangent (to the right) is the local y axis. By default, we heavily punish the z component of the cotangent vector while updating with each new CPlantBox point information.

Article II

VRoot: An XR-Based Application for Manual Root System Architecture Reconstruction

Dirk Norbert Baker, Tobias Selzner, Jens Henrik Gobbert, Hanno Scharr, Morris Riedel, Ebba Bora Hvannberg, Andrea Schnepf, Daniel Zielasko

DOI: 10.1101/2024.06.13.598253 2024

Submitted to Plant Phenomics in June 2024, Accepted January 2025

Plant Phenomics 7 (2025) 100013

Contents lists available at ScienceDirect

Plant Phenomics

journal homepage: www.sciencedirect.com/journal/plant-phenomics

Research Article

VRoot: A VR-Based application for manual root system architecture reconstruction

Dirk N. Baker [a,b,*], Tobias Selzner [c], Jens Henrik Göbbert [a], Hanno Scharr [d], Morris Riedel [b,a], Ebba þóra Hvannberg [b], Andrea Schnepf [c], Daniel Zielasko [e]

[a] Jülich Supercomputing Centre, Forschungszentrum Jülich GmbH, Jülich, Germany
[b] School of Engineering and Natural Sciences, University of Iceland, Reykjavík, Iceland
[c] Institute of Bio- and Geosciences 3: Agrosphere, Forschungszentrum Jülich GmbH, Jülich, Germany
[d] Institute of Advanced Simulation 8: Data Analytics and Machine Learning, Forschungszentrum Jülich GmbH, Jülich, Germany
[e] Human-Computer Interaction, Trier University, Trier, Germany

ARTICLE INFO

Keywords:
Virtual reality
Root phenotyping
Root system architecture
3D image analysis
Immersive analytics

ABSTRACT

This article describes an immersive virtual reality reconstruction tool for root system architectures from 3D scans of soil columns. In practical scenarios, experimental conditions will be adapted to fit the need of the data analysis pipeline, including sieving and drying the soil before scanning. Based on previous reports of automatic systems that do not represent what experts would annotate, we developed a virtual reality system to assist with the extraction of root systems in cases in which automated approaches fall short of expert knowledge. The aim of the present study is to evaluate whether our immersive method is superior to classical annotation approaches when tested on synthetic data sets using untrained participants. Our laboratory user study consists of evaluating the root extractions of participants, along with their rating on central user experience and usability measures. We show significant improvement in *F*1 score across conditions (noisy or clear data) as well as an improved usability. Our study highlights that using virtual reality in root extraction improves accuracy, and we perform an in-depth evaluation of biases that occur when users trace roots in soil volumes.

1. Introduction

Understanding root systems is an underappreciated part of the process of sustainable agriculture [1,2]. Historically, it was not possible to access root systems except by using difficult excavation processes. However, more recent research highlights that an analysis of the spatial configuration of roots, i.e., the root system architecture (RSA), is a critical aspect of understanding plant response [3]. Notably, there is a large variance in root traits [4], which impacts functional aspects of plant behavior. The root traits of a single plant can be measured from its RSA. A digitized version of the RSA yields a full description of the functional and structural traits of the root system. Functional-Structural Plant Models (FSPMs) are coupled simulations of plant structure and functional traits. RSAs can be used as boundaries in these functional simulations to provide insight into plant performance that cannot be measured directly. FSPMs bridge gaps between measurable indicators that might not directly correlate unless the plant is viewed as a continuum model [5].

Ideally, non-destructive observations of RSAs are used in experiments to allow repeated measurements, such as is possible using rhizotrons for statistical descriptions of roots [6]. A full RSA reconstruction in the face of partial root obstruction or destructive measurements is challenging. Non-invasive 3D imaging methods, like Magnetic Resonance Imaging (MRI), can assist by giving a more complete insight into the root system architecture [7]. 3D imaging techniques do not require direct intervention into the plant growth, and they do not require the introduction of transparent obstructions. While the plant growth in a soil column is more restrictive than in a field, key insights can be gained from an in-depth analysis of 3D imaging data, such as the progression of diseases in the plant [8]. Another key aspect in the dissemination of information from plant image data is the potential to gain insight through root modeling, as both in-silico experiments [9] as well as quantification of the continuous processes between soil, plant and atmosphere can provide valuable insights [5].

The extraction of RSAa is more challenging and depends on the quality of the image data. Most approaches to extract RSAs from 3D image data result in tree-like or centerline structures that describe the

* Corresponding author. Jülich Supercomputing Centre, Forschungszentrum Jülich GmbH, Jülich, Germany.
E-mail address: d.baker@fz-juelich.de (D.N. Baker).

https://doi.org/10.1016/j.plaphe.2025.100013
Received 18 June 2024; Received in revised form 8 December 2024; Accepted 17 January 2025
Available online 26 March 2025

morphology of the root systems. Fully automatic approaches fall into the categories of topological analysis [10] or optimization-based approaches [7]. There are also semi-automatic approaches, that require user interaction for key aspects, or to correct the automatic propositions. For a fully automatic approach to function, the globally optimal solution to the extraction problem must reflect the correct RSA. This is not necessarily the case, as in some measurements, artifacts due to soil composition and soil water content can lead to a difference in measured and actual morphology [11].

Manual approaches are a way of dealing with challenging image data properties, as expert knowledge can be required to completely extract the RSA to a degree that it is useable in further analysis. Selzner et al. [12] show this in an analysis of MRI image segmentation and how it impacts automated tracings, an analysis which uses a previous version of VRoot as expert baseline. Past approaches typically aimed at solving this challenge through guided optimization [11] or semi-automatic correction [7]. To assist with manual RSA reconstruction, Virtual Reality (VR) software has been developed, which was used to model RSA functional properties [13]. However, usability of these systems was limited as they were less portable than current VR hardware, which have improved in accessibility and display properties. The use of modern Extended Reality (XR) or VR hardware and software has the potential to increase the space of experimental conditions that are useable in combination with 3D RSA extraction techniques, thus closing the gap between data analysis requirements and realistic experimental conditions. Within VR, RSAs can be visualized directly in a more intuitive embedding, as the 3D displays will be able to accommodate and visualize depth as well as spatial configuration. VR is a promising tool, as it has been shown to improve extraction quality for similar tasks in other disciplines, such as neural imaging [14]. To improve our workflows of manual RSA reconstruction, and to investigate the applicability of VR in these workflows, we have developed VRoot, a VR application that assists in the reconstruction of RSAs by visualizing the 3D volume and providing intuitive toolsets for the reconstruction and adaption of RSAs.

We have built VRoot with the toolsets needed for exact and expedient RSA reconstruction. There are many tasks for which VR improves the quality of task completion in comparison to desktop applications. In this work, we investigate two research questions: Does VRoot improve the data extraction workflow for users annotating 3D MRIs? Furthermore, is the RSA reconstruction using VRoot more exact than using classical methods?

To answer these questions, we have conducted a laboratory user study with participants on-site. With an in-silico 3D root image, we have tasked participants with extracting the root system using VRoot and NMRooting, a state-of-the-art desktop RSA extraction and analysis application [7]. With the resulting RSA reconstructions, we have quantified key traits of the root system, and more importantly, the accuracy of the extraction in comparison to the original RSA.

This work makes several key contributions to 3D plant phenomics. First, we present a new VR-based method to interactively reconstruct root systems from 3D imaging techniques. We quantify the user-based errors and reconstruction artifacts in a laboratory user study. Lastly, we evaluate the use of VRoot based on user feedback obtained through controlled questionnaires.

In the remainder of this section, we briefly describe basic terminology for VR and provide a background for our reference application, NMRooting. In Sec. 2, we describe VRoot in the context of our day-to-day extraction workflow as well as the setup of our laboratory user study. Results of the study and our data analysis results are shown in Sec. 3, while we discuss the findings and implications as well as future directions in Sec. 4.

1.1. Immersive Analytics

Visual Analytics is a discipline of supporting data analysis and reasoning through visualization and graphical interfaces [15]. A subset of this field, and the collection of techniques it encompasses, is Immersive Analytics (IA), which involves the use of immersive interfaces, such as VR, which itself is a subcategory of XR. There have been a large variety of use cases for IA in science [16], and the VR application described in this work is another example. We focus on use-cases that are comparable to the use of VRoot, to provide context and an overview of instances in which IA has provided increased insight for data analysis pipelines. The neuron tracing application developed by Usher et al. [14] use Head-Mounted Displays (HMDs) for the sparse annotation of 3D image data. Usher et al. [14] developed an application to trace neuron connections in 3D space with handheld controllers and consumer-grade VR hardware. Usher et al. show that there is a significant speedup for experts to annotate neuron traces in VR in comparison to a classical desktop application. This result has been further improved by the introduction of topological features assisting with the extraction of neuron traces as shown by McDonald et al. [17].

Immersive displays are varied, ranging from room-scale installations to small portable devices. In comparison to the ImFlip150 system used in Stingaciu et al. [13] for root annotation, HMDs require less space, are mobile/movable, and can be comfortably used at any office workplace. A disadvantage of HMDs in this comparison can be the loss of reference in the real world [18]. The HTC Vive Pro used in this work is a wearable low-persistence display [19], similar to other HMDs. It is tracked using base stations that help the HMD infer user position and orientation. VR controllers are similarly tracked, in addition to containing buttons that users can press for interactions. Interaction in VR is commonly done using interaction *metaphors*, which are user actions done using controllers or gestures that impact the virtual world, as the user cannot directly interact with it. These include grabbing to virtually pick up items, as is common in VR applications, but also pointing for movement outside of the restrictions of the installation or walkable space [16]. Our central consideration of choosing VR techniques without involving pass-through or see-through augmented reality devices involves reduced depth perception of 3D renders in see-through devices [20] and reduced text readability in pass-through devices [21]. As such, any use-case of multitasking between the virtual and real world that directly relates to the RSA data would suffer from these drawbacks. Additionally, performance on difficult tasks tends to be reduced when multitasking [22].

Evaluating the use of human-centered techniques, particularly in the case of immersive systems, is challenging [23]. However, there is a long history of formal analysis in human-computer interaction that we can make use of. For example, a common method of evaluation by users is the System Usability Scale (SUS) [24], which introduces the notions of usability regarding task completion. Questionnaires such as SUS have been predominantly designed for software system evaluation but can be used for immersive software, as the inherent effects are similar [25]. Evaluation metrics used in this work are described in Sec. 2.4.

1.2. NMRooting

NMRooting [7,26] is a framework and application for the extraction of root system architecture by extracting the minimum-weight shortest paths with additional functionalities, such as gap closing and semi-manual extraction. We chose this application as a baseline as it is a well-established application that is also a proxy for similar applications and approaches, such as those developed by Horn et al. [11] or Zeng et al. [10]. Furthermore, NMRooting has seen regular use, including recently by Le Gall et al. [27], who used the non-destructive investigation of the RSA through NMRooting to analyze whether the root water uptake profile is an indicator for plant development. NMRooting uses desktop user interaction metaphors such as clicking and dragging to fulfill 3D annotation. Clicking in NMRooting traces a selection ray from the viewing surface to the isosurface data, marking the first surface that it hits. Dragging is a metaphor that allows users to turn the camera around the isosurface data, as well as zoom and pan. Within NMRooting, users can alter the automatic tracing of the data set, which is more in-line with

the use of applications such as TopoRoot [10] or the application developed by Horn et al. [11]. However, more fine-granular alteration to the reconstructions are possible by directly interacting with the data, yielding higher extraction accuracy in cases such as larger gaps as reported by Horn et al. [11].

2. Methods

Fig. 1 shows the total setup of our workflow, from data extraction to eventual use. Plant containers are typically plastic containers with a single plant growing in sieved soil. We are aiming to measure and extract RSAs from a large variety of soils and water contents. The resulting soil volume is generally a slice based 3D volume. This volume might require stitching, depending on the explicit setup of the MRI scanner. The soil water content together with the soil type (such as either loam or sandy loam) impacts the overall signal-to-noise ratio [26,28]. Furthermore, ferromagnetic particles within the soil might impact the measurement quality and might even disrupt root signal continuity, resulting in gaps in the root system [11,12].

Segmentation is generally a voxel-based mapping that labels a part of the image as either foreground or background, thus reducing the complexity of the data. For our pipeline, we generally use 3D segmentation by deep-neural networks, as implemented by Zhao et al. [29] and Uzman et al. [30]. Previously, we have shown that this step improves both automatic as well as manual RSA reconstructions [12]. Fig. 1B contains an illustration of this step. Generally, network architectures vary, but the key challenge for segmentation networks is providing a full view of the root system in the soil, while removing noise from it.

This paper uses a semi-automated method as reference, and most automatization methods rely on sparse extraction from volumetric data. NMRooting developed in Van Dusschooten et al. [7] uses a voxel signal cutoff to determine what areas could include root voxels, followed by a signal-strength-weighted shortest-path algorithm to extract the root system. A key aspect that drove our implementation of VRoot is a challenge presented in the automatic tracing implementation by Horn et al. [11], who included a comparison to manually annotated RSAs. Their algorithm optimizes signal-based features. They highlight cases in which extractions that differ from manual annotation are computed, because the algorithm chooses smaller (more optimal) gaps based on signal strength, while the expert annotation bridges a comparably much larger gap in the segmented image data. This kind of expert knowledge is hard to incorporate into automated algorithms in cases where the information cannot be gained otherwise, for example through repeated measurements or measurements at a later point in time when the roots have developed further.

Our output structure relies on the Root System Markup Language (RSML) [31]. It is based on the XML standard and can be extended to other use-cases. We use RSML both as data output as well as the primary source of information when rendering RSA structures in 3D.

Ultimately, the VR application presented in this work provides an immersive visualization of root systems and can easily be coupled to automatic tracing algorithms through their standardized output. Our application furthermore benefits from segmentation approaches that enhance the spatial visibility of the root morphology [12]. Lastly, using our approach, one can use previously unusable image data, either because automated tracing algorithms still fail in certain cases, or because a level of precision needs to be reached that would be otherwise unobtainable with desktop software.

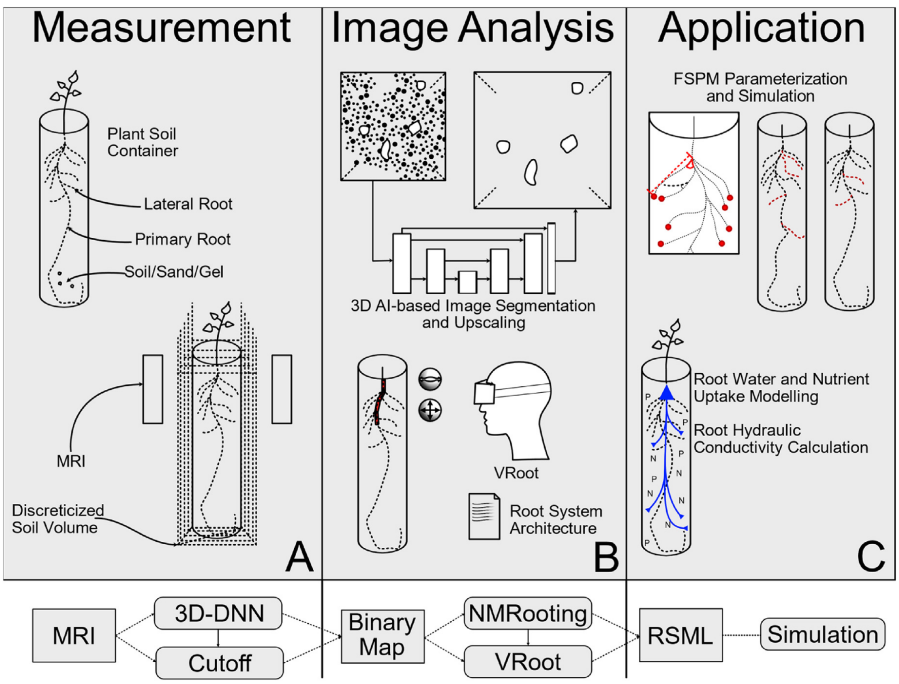

Fig. 1. Process overview of our MRI analysis pipeline. A: Plant containers are non-magnetic tubes with sieved soil. These are scanned using an MRI scanner, resulting in a voxelized soil volume. B: If the data is too noisy, 3D-U-Net segmentation can be employed in addition to the explorative functions of VRoot. C: The resulting RSAs can be used for FSPM calibration or functional simulation boundary. Below, we present a flowchart of the overall steps in our pipeline.

2.1. Virtual reality root tracing

We present *VRoot*, a novel method for manual RSA reconstruction and correction. To obtain the RSAs with the accuracy that we require, we developed a tool that allows expedient fine-tuning. Manual annotation in VRoot is loosely based on methods developed and used by Stingaciu et al. [13] and Usher et al. [14]. To support a wider usability in terms of target hardware, and customization options, we developed[2] VRoot on top of Unreal Engine[1], primarily to allow for reproduction even if hardware constraints are altered. A key aspect of the rendering of 3D imaging data in VR is the ability to look at the data from different perspectives while retaining the dimensionality of the data regarding their perception. We implemented VRoot with the key idea that different users might want to interact on different scales and in different postures, as previously highlighted by Zielasko and Riecke [32].

VRoot is an application that assists with very accurately tracing RSAs in 3D volumes, by providing immersive visualization and intuitive interaction. VRoot uses the Unreal Engine for cross-platform rendering and porting, while the interface to VR hardware uses the OpenXR abstraction standard. The data analysis system is a python server, communicating with the application via a remote connection implemented in ZeroMQ. VRoot almost exlusively uses geometric representations of data, augmented by custom interaction that allows editing of graph structures and visualization of soil scans. In the application, the soil volume is thresholded and visualized as isosurface computed around a cutoff value that can be chosen dynamically within the application. We use the Visualization ToolKit (VTK) [33] implementation for isosurfacing. VRoot is used to manually extract and edit RSAs, which are visualized on top of the 3D volume using the geometrization scheme described in Baker et al. [34]. Our implementation of VRoot consists of a full analysis pipeline using interaction metaphors, implemented[2] using the Unreal Engine. We implemted an RSML authoring system in VR, along with the possibility to extract, change, as well as fine-tune RSAs.

Fig. 2 shows sample views from the user's perspective in the application. We chose a darker environment to reduce eye strain. Users interact with two controllers, one for selecting as well as tracing, while the other controller is used for grabbing metaphors.

Fig. 2A shows the basic user interaction components. Interaction with the RSA is done via the nodes. All nodes in the RSA are selectable and while tracing depends on manual user interaction, changing of properties is selection set-based, meaning that users can mark as many nodes as desired to change their properties. The widgets (grey) change the selection set's properties, such as diameter and position. Fig. 2B shows a snapshot of a user drawing a root, indicated by the pink interaction. This is automatically available once the number of selected nodes is exactly one, or without any tracing present. The widgets to edit node properties always work on the selection set, changing the diameter or position of the selected nodes. The volume itself is displayed as an isosurface, as seen in Fig. 2C, whose signal cutoff value can be changed from within VR through the use of a slider widget. The full visualization of the RSA always uses the RSML topology and assigns colors to root order. We are using dithered translucency for the isosurface to avoid depth-perception issues with the surface. More general root functionality that is outside of node editing is accessible via a point-and-click menu, such as assistance tools for time series annotation or editing tools for RSA topology. Users can generally place nodes freely within the 3D environment, though it needs to be stressed that there is a degree of freedom in VR, the rotation, that is not captured by common data types, such as RSML or VTK format. To enable a more expedient workflow, many more complex tasks, including the visualization algorithms, are offloaded onto a server to allow the interactive application to run smoothly while still allowing the completion of complex tasks. For a selected data set, the system searches for the most recent tracing and displays it on top of the 3D image.

We designed the application with expert workflows in mind and have been continuously improving the workflow to assist with observations such as by Stingaciu et al. [13] and Selzner et al. [12]. However, the assessment of a system that depends on human interaction is not as straight forward as assessing the quality of an algorithm, even with a ground truth data set present. We are evaluating this in a more controlled fashion by performing a user study that is being guided by synthetic data and user questionnaires, under the restriction of using a pool of potential users that all have a similar knowledge level about the applications. We chose to evaluate untrained user performance for this reason, allowing us to focus on the relative performance of the applications without needing to balance the data for previous experience.

2.2. Laboratory user study

To answer our question on whether VR annotation can outperform state-of-the-art desktop annotation for RSA reconstruction, we performed a mixed design laboratory study, assessing the applications within-subjects with the between-subject condition of water noise. We compare our software against NMRooting described above, since NMRooting is not only state-of-the-art for 3D annotation, but also is similar to other applications. The evaluation of the study is aimed to answer the question on whether the VR software yields a higher reconstruction accuracy as well as a higher usability. We map performance to reconstruction accuracy in a virtual MRI scan: Our comparison between applications and conditions relies on the assumption that how closely a participant (after a short training phase) follows the ground truth with their annotation is a direct indicator of the usefulness of the application. In this instance, the term laboratory study refers to a controlled setting in which human participants with similar starting conditions could perform tasks and evaluate the applications. Through the use of a sufficient number of individual participants, effects that are individual to certain people should be eliminated, and the overall usability of the software can be evaluated. To enable this process, the set of possible options within a single application has to be restricted, so we exclusively use "tip-to-tree" and node annotation in NMRooting and basic drawing without correction in VRoot.

The user study was designed to answer our questions and assumptions on the improvement of software and measurement quality from MRI scans. We postulate the following hypotheses on the application performance on the software level as well as on the data level.

Improved Workflows: We expect that the major indicators for software quality will be im-proved when using VR software. These indicators are an improvement (**H1**) of System Usability as well as an improvement in the subjective pragmatic performance (**H2**) of the software. These hypotheses will be tested using the participants' evaluation using the questionnaire after task completion.

Extraction Accuracy: For the extracted root systems, we postulate that key relevant measures will be impacted by the use of VR. The total root length is expected to be different (**H3**), which also applies to the branching density (**H4**). We believe that the VR software will result in a higher overall accuracy (**H5**) which is further impacted by the presence of water noise (**H6**). Water noise and impact of signal-to-noise ratio have well-reported impacts on reconstruction accuracy [7,10–12], which is why we assume that it is a significant factor in the reconstruction accuracy of participants. We expect that the total root length is closer to ground truth when using VR (**H7**) and that the VR extraction of the branching density does not differ from the branching density within the virtual MRI (**H8**).

2.2.1. Tasks & measures

The main task that participants were asked to perform is extraction of the RSA from an MRI soil column scan. This includes the extraction of the pathway of individual roots as exhibited within the MRI scan, loosely

[2] VRoot implementation: https://github.com/dhelmrich/VRoot.

[1] Unreal Engine, developed by Epic Games Inc. https://www.unrealengine.com/, accessed 2025-03-27.

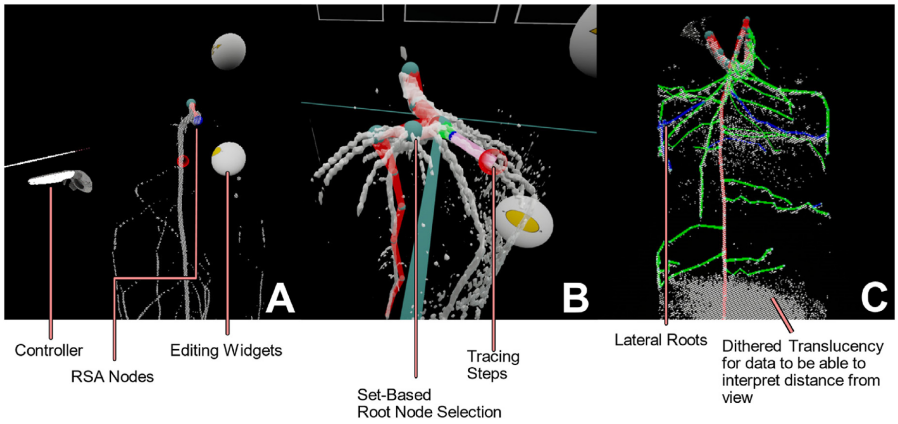

Fig. 2. Annotated screenshots of VRoot. A: Two-handed use of the drawing function. New position and connection is indicated in pink. B: Selection is set-based and changes are done on all selected nodes, drawing is only possible with one. C: Subsequent root orders have high color contrast and MRI is dithered for depth-preserving translucency.

based on signal strength. Participants were asked to mitigate noise effects if present and extract a fairly simple explanation for the signal that they were shown. Furthermore, participants created a *labeled* RSA, which includes the order of the root explicitly. Participants were asked to label both primary and lateral roots as such. The tracing of the RSA resulted in each case in a full RSA, including positioning but excluding diameter. Participants were asked to provide their demographic information as well as a subjective evaluation of the software they were tasked with using. In total, participants performed the extraction task two times, with the evaluation of a questionnaire in between and at the end.

Participants were tasked with extracting a root system from an MRI scan, once using the VR application and once using NMRooting. The water noise condition spanned both data sets, meaning that independent of the order, a participant either completed the task for each application with water noise, or without. Participants were tasked with tracing a virtual MRI scan, as described in Sec. 2.3. Extraction of the RSA was done with both applications, and the resulting structures were compared against ground truth.

Participants evaluated each application with the System Usability Scale (SUS) [24], the User Experience Questionnaire (UEQ) [35] as well as the NASA Task Load Index Short (TLXs) [36]. These are described in Sec. 2.4.

2.2.2. Procedure

Participants gave their informed consent. Participants were divided into four groups by ID. The first distinction was made on whether a participant received data with water-like noise. Furthermore, it is varied which application a user tested first, resulting in four conditions. The conditions were *order of application* and *water noise*. The study procedure consisted of five steps. In the first step, participants would quantify their own previous experience and calibration measurements were made to setup the HMD. Afterwards, participants would be introduced to the first application (Desktop or VR) and after this initial training phase, the study data set would be loaded, and the participant performed the task without help. Participants then evaluated the application using questionnaires. Lastly, these two steps would be repeated with the other application. For the full description of all steps involved in the individual phases, see App. B.

2.2.3. Apparatus

In our experiment, we ran VRoot on an HTC Vive Pro HMD. In our tests, the application framerate was typically within 80–90 frames per second. The entire study was conducted in the "Virtual Reality Laboratory" in the Institute of Bio- and Geosciences 3 of the Forschungszentrum Jülich GmbH. The study was conducted sitting at the laboratory desk, facing the monitor. The VR software was used sitting by all participants. For considerations on whether to support or design a system for standing or seated setups, we refer to current literature [37].

The questionnaires as well as the NMRooting application were used on a desktop PC with a desktop resolution of 1920 × 1080 with mouse and keyboard. Our tests were performed on a PC with an Intel i7-8700K CPU, 32 GB of RAM and an NVIDIA 2060 RTX SUPER GPU.

2.2.4. Participants

The user study data set was acquired from over 20 participants working on-site at Forschungszentrum Jülich. We have contacted potential participants, pre-emptively excluding anyone with either previous VRoot experience or knowledge of the goal of the study. Furthermore, we required normal or corrected-to-normal vision.

Total participation in the user study was $n = 20$. These include in total 15 male, 4 female, 1 non-binary and 0 other. Age distribution was almost uniform from 20 to 43 years, with a median age of 33. Self-reported experience using 3D applications was 4 participants with no experience, 8 users who reported using 3D applications at least once, 6 sporadic users and 2 experts. Self-reported experience using VR applications was 5: None, 5: Once, 7: Sporadic and 3: Expert. None of the participants had previous experience in the specific application NMRooting or the specific application VRoot.

2.3. Evaluation using FSPM simulated root data

We are evaluating user-based extractions of root systems in the context of a virtual MRI scan. These virtual MRI scans were designed specifically for this study and the RSAs were simulated using the FSPM CPlantBox [5]. We calibrated the simulation for the task and slightly increased the inter-lateral distance for the first-order lateral roots. This RSA serves as a ground truth measurement. With noise that we typically see on a larger scale, such as Fig. 3A, we modeled a smaller bean root system seen in Fig. 3B and imposed a noise model on it.

We computed the signal strength of the resulting MRI scan by using a simple heuristic based on the total volume of a root segment in a certain voxel. To avoid non-uniform task performance in the extraction task, the

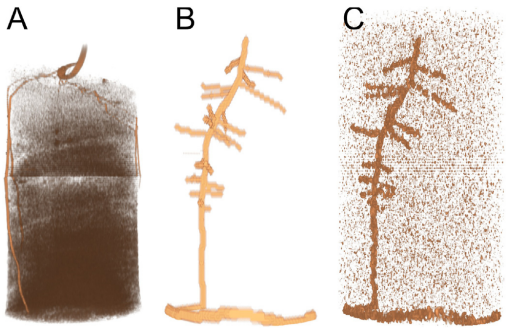

Fig. 3. Side-by-side comparison of the study image data sets, scaled to image height as opposed to real height. A: We typically observe, depending on soil type as well as soil water content, highly ununiform noise. Locally, this might be expressed as smudges around the roots. B: Our simulation of a faba bean is being rendered with respect to the relative root length/diameter through a voxel. This image data set is the use-case for the noise-free participants. C: We added and subtracted noise features using a Weierstrass transformation. This results in slightly more complex, but uniformly complex, image data.

between-subject condition of water noise was chosen such that either the whole data set had noise effects or no part of the data. Within a soil cylinder of 1.5 cm diameter, a soil volume with water noise was seeded. The noise was locally scaled with the signal-to-noise ratio of 4.3. Noise addition/substraction was followed by applying a weierstrass transformation.

Since the original root system has been created by an FSPM, we can directly compare it with the manual extractions. This allows for an exact quantification of errors, which helps in assessing the factors in the decision-making process on what application to use for the pipeline. CPlantBox has been well-researched in terms of its stochastic properties [38], and is fit to be used as baseline for a user-based evaluation of extraction software. Additionally, any effects we would see in terms of the impact of the synthetic nature of the plant we would see in both applications equally.

As described in Horn et al. [11], to be able to confidently match between user-annotated RSAs and those generated by software or simulations, the correspondence between the architectures needs to be calculated. In this evaluation, we have used the distance-matching threshold of $d = 15$ in voxel units that was used by Horn et al. [11]. Since any length of simulated root could correspond to one or more manually annotated segments, and likewise one manually annotated segment might correspond to more than one simulated root, we compute the full segment distance matrix as

$$(D)_{i,j} = \min_{x,y} \left\| P_i + x \cdot (P_i - P_{i-1}) - P_j + y \cdot (P_j - P_{j-1}) \right\|_2 \text{ where } x, y \in (0,1) \subset \mathbb{R} \tag{1}$$

where P_i is the point coordinate of the ith segment, x and y are optimization parameters and the resulting distance matrix D only captures the minimal euclidean distance between two lines. This will result in a base distance metric that we use to match segments of different lengths. The matching process first assigns 1-to-n correspondences to simulated segments before it tries to match any unmatched manual segments to those simulated segments that were previously matched to exactly one segment. Organ-level label continuity is provided through matching roots. Two roots are considered matched if their cumulative distance $D \cdot (e_n \times e_m) \mapsto d^{(m,n)} \in \mathbb{R}$ is the lowest among all other possible assignments with respect to the sets e of the segment indices.

We computed the accuracy based on root matching to ensure that the correct identification of roots is rewarded and to measure extraction differences that contribute to differences in root length. This topologically-aware accuracy is computed using a ground truth root set of I_{GT} and a set of traced roots I_T, with a set of $I_c := \{n \in I_{GT} \,|\, \text{argmin}_{m \in |R|} d^{(m,n)} \cap d^{(m,n)} \leq d\}$, signifying the correctly matched roots.

To compute the measures for the correctness of the extracted RSAs, we first match the root systems to the ground truth. This ensures that there is topological information in the resulting scores. Root Lengths L_n generally refer to the total length of all segments corresponding to the root, namely $L_n := \sum l_i^n$ of the $i \in I_n$ segments of organ index n. We use the ground truth L_{GT} as the reference for the scores, where L_{GC} is the total length of correctly matched roots computed from $I_{GC} \subseteq I_{GT}$. The root(s) that are matched to a root in the ground truth are a subset of the root set of the tracing, $I_{TC} \subseteq I_T$. It follows that the total length of false negative roots that were not traced is $L_{FN} = \sum l_i^n I_{GT} \setminus I_{GC}$. We especially highlight that the total length of correctly, which means matched, ground truth roots is not necessarily the same length as the sum of matched roots in the tracing, meaning that $L_{GC} \neq L_{TC}$ because $I_{GC} \subseteq I_{GT}$ whereas $I_{TC} \subseteq I_T$. L_{LP} is the total length of false positive roots, i.e., computed from segments that were not present in the ground truth but present in the tracing. The *recall* value R is a measure that encapsulates how much (in length) of the root system was traced, defined as

$$R = \frac{L_{GC}}{L_{GC} + L_{FN} + \min(0, L_{GC} - L_{TC})} \in [0,1] \tag{2}$$

where we further penalize the tracing in cases where the extracted root length is smaller than the length of the ground truth. On the other hand, the *precision* value P encapsulates whether the manual extraction contains only as much length as the ground truth root system:

$$P = \frac{L_{GC}}{L_{GC} + L_{FP} + \min(0, L_{TC} - L_{GC})} \in [0,1] \tag{3}$$

For the precision, we further penalize roots that were extracted correctly but are too long in comparison to the ground truth. The precision and recall values are asymmetrical, decreasing with different metrics, but can be summarized by the symmetrical F_1 score, which decreases with both false positives and false negatives:

$$F_1 = 2 \cdot \frac{P \cdot R}{P + R} \in [0,1] \tag{4}$$

The F_1 score is a comparison score that yields relatively similar values for different deviations from the ground truth. This score allows the comparison of applications that exhibit different characteristics, but due to its symmetrical nature, allows the comparison between them.

2.4. Measures for application comparison

The measures on usability of the software as well as user experience are difficult to measure objectively, and thus, questionnaires are typically utilized. These questionnaires include the System Usability Scale Questionnaire (SUS) [24], the User Experience Scale (UEQ) [35], and the NASA Task Load Index Short (TLXs) [36]. Due to the human interaction component of the system, we chose to use standard methods of evaluation with the added component of knowing the ground truth of a virtually generated MRI scan. Thus, we obtain a combination of subjective and objective measurements for assessing the software. We attached the full participant survey in the supplemental material.

The SUS is primarily aimed at quantifying the subjective user assessment of whether the given system is fit to help the user solve the problem. The resulting score is scaled within $[0,100] \subset \mathbb{N}$. Commonly, the software is considered to rate well on this scale if it is above 85 [39].

UEQ scores are a way of evaluating the quality of the subjective user experience regarding basic descriptions of the software. The UEQ is a mix of adjectives that are presented in a contrasting manner. The adjective

sometimes has overlapping meanings, and the general assessment of the software regarding these properties is very subjective. The NASA Task Load Index Short is a questionnaire to assess the subjective difficulty and strain on the user when completing the tasks. Users evaluated the applications in an online questionnaire, which we have attached to the supplemental material. We measured the extracted RSA, as well as camera data from participants, in addition to participants completing the questionnaire.

2.5. Data analysis

We aggregated the data into two groups, based on the water noise condition. We computed the subjective scores per participant according to the respective guidelines. This applied to SUS [24], UEQ [35], and the TLXs [36]. We performed the data analysis entirely in Python. In tables or figures, we will refer to VRoot simply as VR, and to NMRooting as Water/No Water Conditions are referred to as + W or -W respectively. As such, VR + W refers to all data points of the VR software that have the water noise condition. In the following, total (T) refers to all data points. For hypotheses testing, our confidence cutoff is $p = .05$.

We tested all measures for normal distribution using the Shapiro-Wilk test, to ensure subsequent tests are informative and valid. For the sake of uniformity and comparability, we are using non-parametric tests in cases where not every condition is normally distributed. We chose the Mann-Whitney (Summed Rank) test for differences in median in cases where we do not find a normal distribution.

Statistical reporting includes the test statistic $t(\mathrm{DoF})$, the critical value p, the effect size value (Cohen's d [40], defined as for differences in statistics T_i), as well as degrees of freedom (DoF). In cases where ground truth is available, we test for a specific means using t-tests. The test values for normal distribution can be found in App. A. We use t-tests for SUS, UEQ, and TLXs. Mann-Whitney tests will be used for total and average root length, F_1 score, and inter-lateral distance.

3. Results

We omitted one subject (female, +W) from the study after the subject succeeded the task *trace the taproot* within the training phase but did not succeed with that task in the study phase. This means that we have 10 data points for the condition -W and 9 for the condition + W. The tracing and details on reasons of omission can be found in App. C.

3.1. Descriptive data

We include box-plot descriptions of the relevant measures. Fig. 4 shows the SUS scores over all conditions as well as the TLXs scoring and UEQ pragmatic quality. The SUS scores are scaled within [0, 100], though must not be understood as percentages. We have computed the median in instances of data points that have no normal distribution, which is indicated by the orange line in the boxplots. Data sets that contain a ground truth (simulation) value indicate this value with a red line. Data points outside of the inter-quartile range $(1.5 \cdot (Q_3 - Q_1))$ have been included and marked as x-symbol.

The within-subject condition of the order of the application was combined it served as balancing of the applications against learning effects. The subjective scores that are relevant to the applications can be seen in Fig. 4. Herein, we present the SUS, the Task Load as well as the pragmatic quality. Fig. 5 shows the accuracy scores of the data sets. For a more in-depth understanding of the individual effects, we further present relevant RSA measures in Fig. 6. Herein, we have a ground truth measurement for all conditions. Ground truth measures were extracted from the virtual MRI algorithmically, meaning that we extracted measures with the MRI mapping and voxelization in mind. For the comparison, we show as a red line the ground truth value from the actual simulated data set as opposed to the parameterization. For the number of lateral roots, we filtered roots of a length $l_i < 3$ [cm] to allow for the evaluation of the extraction without unnecessarily including tracing artifacts in NMRooting (seen in Fig. 7). The inter-lateral distance was calculated on the taproot only.

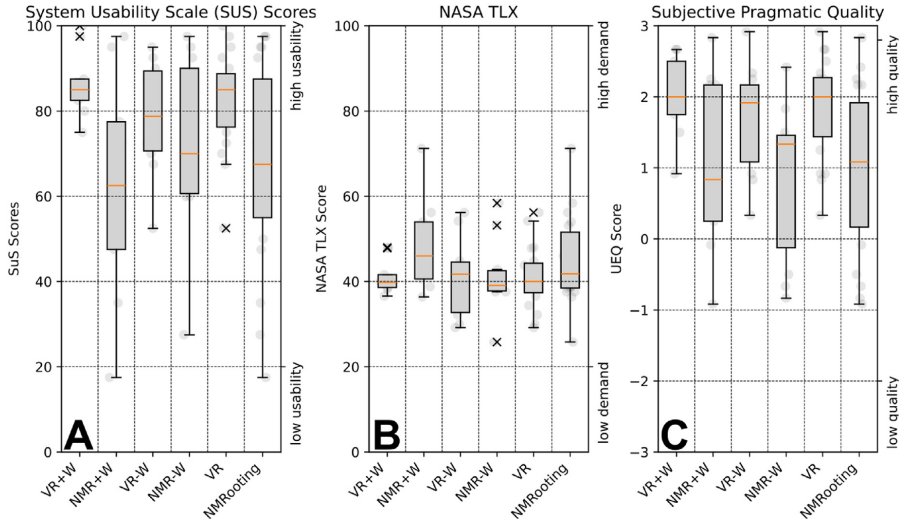

Fig. 4. Boxplot with conditions on x axis and score on y axis, orange line is the median. A: Overview of System Usability Scoring across conditions. Scoring across order conditions (within-subject) was summed. B: Overview of Task Load Index Short Questionnaire score, sum of all questions except perceived success, which was inverted. C: Overview of UEQ participant scoring for the pragmatic quality of the data.

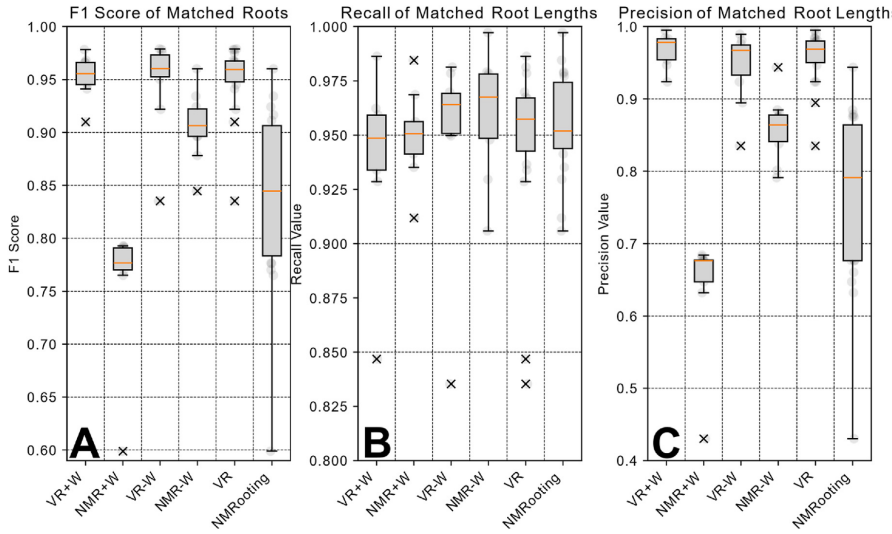

Fig. 5. Boxplots showing the median (orange) and distribution of the data with outliers marked as x. A: F_1 scores of the extracted root systems. B: Recall value of the matched root systems. C: Precision score of the matched root system.

3.2. Results regarding user study hypotheses

We summarized the statistics and hypothesis data in Table 1, which includes the most important data points. This section will give a brief overview of what results we measured on the hypotheses we had described in Sec. 2.2, and further indicators regarding ground truth comparison.

SUS: We observe an average usability of 82 for VR and 67 for NMRooting. VRoot has scores significantly better than NMRooting as measured using a one-sided *t*-test (H1). Notably, VR + W has a average usability of 86 and NMR + W of 62, with tests showing significant improvement by using VR, which is not the case for the comparison VR-W > NMR-W.

UEQ: Pragmatic experience that can be extracted from the UEQ is the average of the three statistics *Perspicuity*, *Efficiency*, *Dependability*. VR scored 1.841 on average and NMRooting scored 0.912, resulting in a significant difference (H2).

Root Length: The length measures of the root system are average and total root length, seen in Fig. 6A and B respectively. We computed the differences in extraction quality regarding these two measures, allowing for an assessment of overall extracted biomass and the correct identification of individual roots. The average and total root length contain useful information about what challenges participants encountered during the annotation. We observe a difference between NMRooting and VR in terms of extracted root lengths (H3). Deviation from ground truth was tested as well as the difference between the methods. The extraction of the average root length was, on average, correctly done in NMR-W, by a single-sample *t*-test ($t(9) = -0.979, p = .353$). The total root length extraction in VR-W yielded no significant difference to ground truth, yielding $t(9) = -0.426, p = .680$. Other conditions, including aggregates, yield significant differences. Differences between the applications exist in the case of the individual conditions (+W/-W) as well as the aggregate. We furthermore observe that the extraction of the total root length yields larger differences in the +W condition.

Root Topology: Computing the extracted number of laterals that have a minimum length of 3 cm, we find that only NMR-W found the correct number of lateral roots. For the branching density (H4), we observe a difference between the extracted number of lateral roots in the +W case, but not in the -W case nor in the aggregate case.

***F_1*:** We find that the F_1 score of the extracted root systems is significantly higher for VR in both + W and -W conditions (H5). Additionally, we find an increase in the difference between the applications once water noise is present (H6).

Influence of Water Noise: We observe a significant increase in the difference between the applications when water noise is present. This is the case for the average root length, the total root length, and the F_1 score. The computation for the F_1 score test required random pairing between individual scores.

Inter-Lateral Distance: This measure is defined as the average distance between two consecutive lateral organs O_i, O_j. VRoot users correctly extracted the inter-lateral distance, tested using a two-sided Mann-Whitney-U Test. NMRooting yielded a significant difference to ground truth in the +W condition, by single-sample *t*-test with $t(8) = 1.619, p = .002$ and an effect size of $d = 1.575$. There were no significant differences in both VR cases as well as the NMR-W case.

4. Discussion

We postulated that the VR software would yield a different usability, which has been confirmed in H1 for W+ and overall use of the software. The effect was much smaller for W-, resulting in a very small effect size and no significant difference. However, while no significant difference was measured, there is furthermore no indication that classical applications perform better in any of the study conditions. The overall measured variance for the measurement of pragmatic quality (H2) was fairly high, see Fig. 4B, resulting in no significant difference between the applications in the W+ condition.

Objective measurements were more uniformly successful, with the notable exception being the detection of the correct number of lateral roots. In that metric, all median extractions were below ground truth, with the exception of NMR-W, which was slightly higher. Interestingly, NMR + W extractions consist of less roots overall, which might be a result of the noisy data inhibiting the participant's ability to extract roots. We confirm other findings, such as by Selzner et al. [12] and Horn et al. [11]

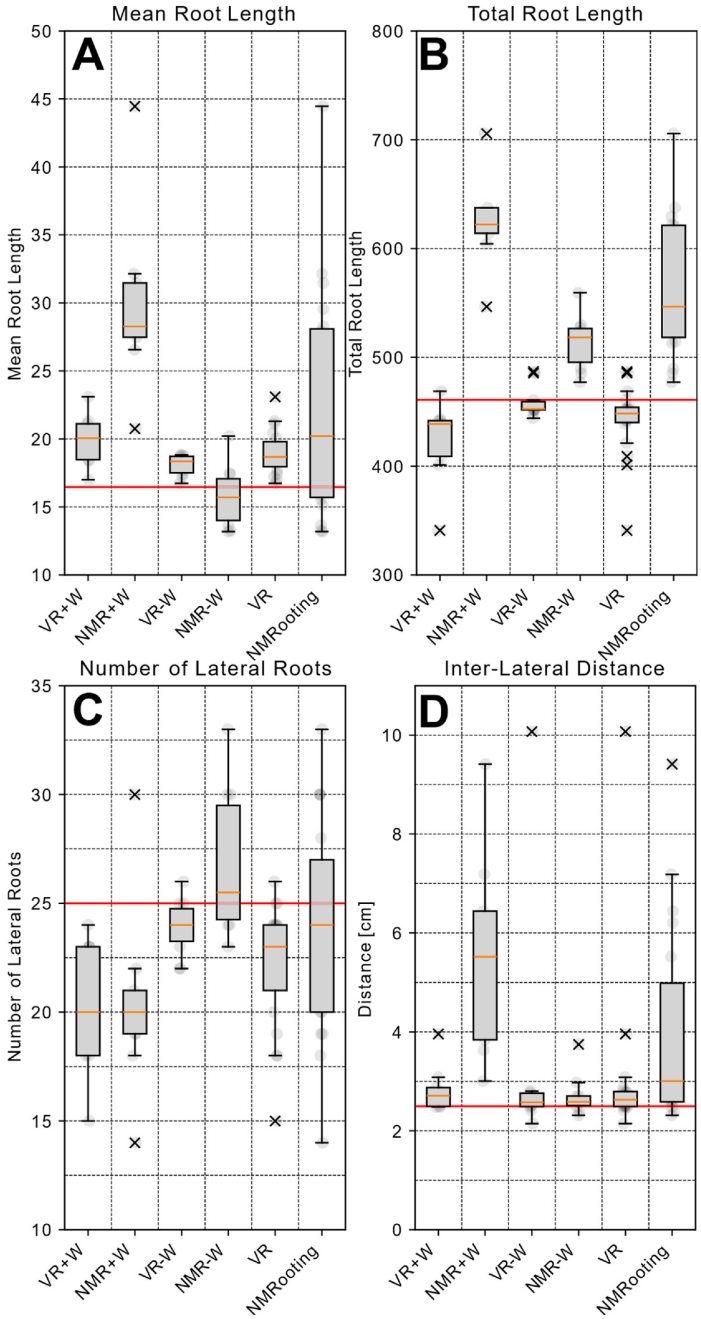

Fig. 6. Box-plot graphs show ground truth in red, median in orange, and outliers as x. A: Boxplots of average root length B: Boxplots of total root length $\sum l_i$ C: Number of lateral roots ($l_i \leq 4$ cm) D: Inter-lateral distance d_i.

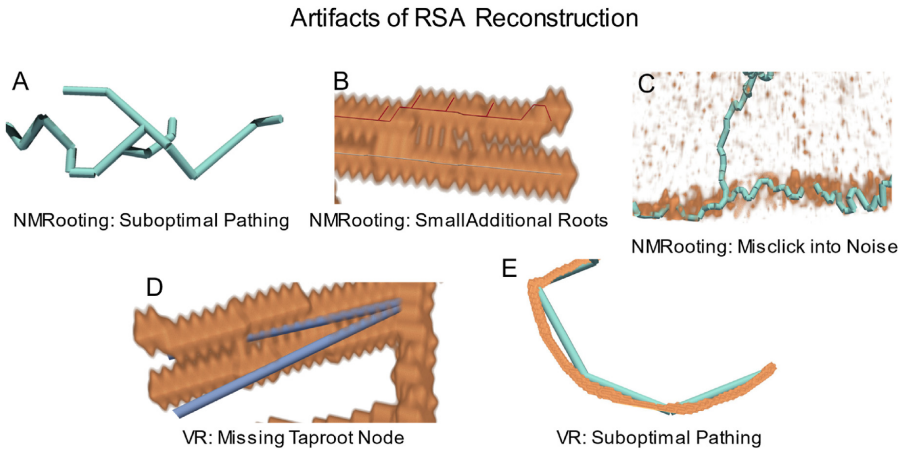

Fig. 7. Artifacts of RSA reconstruction in each application that occurred with participants.

about the impact of noise on the RSA reconstruction.

4.1. Task execution

Generally, task execution posed no problems for users. Users needed a few minutes to get accustomed to using the HMDs. Though pre-emptive measurement and calibration were done for the interpupillary distance, some users reported issues with depth perception, or depth-perception issues became apparent during the training phase. Though explicit introduction and prompting to repeat a certain interaction were done during the training phase, some users did not make use of all available options, particularly navigation, during task completion. This occurred in equal parts with NMRooting and VRoot.

4.2. Interpretation of results

The simulated root contained certain artifacts that would have made it fairly hard to trace for new users. Particularly, we note that the matching-based F_1 score is remarkably good for the VR software, which is in part due to users matching the correct root length for the individual organs successfully. The spatial distribution of the root system is more obvious in VR, which reflects in the root systems that were drawn, even under the condition that users do not edit root nodes that were already placed. The restricted set of functionalities was mitigated by the introduction of a training session, during which the procedure was explained, leading to most participants already pre-planning the taproot in a way that made it possible to achieve a high accuracy.

One aspect of the results we would like to highlight is the fact that with no water noise, the VR application still yielded better results in terms of matched length-based F_1 scoring, see Fig. 5A. Furthermore, the differences between the individuals were not very high, resulting in a standard deviation for the VR conditions of 0.02 (W+), 0.04 (W-), and 0.03 (T) respectively. However, this is partially due to the fairly 'destructive' nature of the F_1 score, leading to different user-based tracing errors to result in a similar score. The recall value R, as seen in Fig. 5, is fairly uniform across applications, with slightly more "under-tracing" done in the VR application. The F_1 score and particularly the precision P was low (but not extremely so) for NMRooting in the +W condition. The primary reason for this was the fact that each time a user clicked into the data and did not continue a root from the tip but rather from the closest segment regarding the signal strength, this induced a small lateral root at that point that users might have missed. For the actual comparison of the number of lateral roots, we discarded shorter roots to avoid comparing against this technicality. We note that the F_1 scores closely follow the power law distribution, tested by means of Cressie-Read test for goodness-of-fit at $\chi = 0.021$ and $p = 1.0$.

There is a significantly higher spread in the perceived system usability measured from NMRooting in the +W condition as well as overall. Tested variances were significantly higher according to the F-test for the overall condition ($F(18) = 4.330, p = .002$) as well as + W ($F(8) = 11.493, p = .001$) but not for the conditions -W ($F(9) = 2.571, p = .087$), even though this condition does fail the test for equal variance ($F(9) = 2.571, p = .175$). Between the VR + W and VR-W conditions, there is an increase in variance that is, while not significant, at least notable. Moreover, the inter-lateral distance has a large variance in the NMR + W condition due to several artifacts. We will note that, for the estimation of the inter-lateral distance, while there were no significant differences in VR + W, VR-W, and NMR-W, that most users (68% for VR and 84% for NMRooting) over-estimated the inter-lateral distance compared to the ground truth. We cannot make assumptions on the applicability of this regarding a real MRI scan, but our findings provide some insight into user bias when extracting parameters, particularly for FSPM simulation, from 3D imaging data. The average relative error for inter-lateral distance for VR users was 0.234 and for NMRooting it was 0.617.

The average and total root length is, in part, influenced by the presence of water noise, in both applications. NMR + W exhibited a larger total root length while still yielding a lower than ground truth number of laterals. However, the increased inter-lateral distance partially relates to misidentification later in the data. In VR, the presence of water noise caused under-identification of roots. We will note, that in the simulation data, there was one very thin root that would have been very hard to identify. There was a systematic issue with users not being able to identify that root and thus, the VR-W case was lower in median regarding the number of lateral roots users were able to identify.

4.3. Artifacts of the RSA reconstruction

Our general results show an improvement in the extraction quality using VR. More specific phenomena can often be explained taking into account observations from the study or by closer inspection of the data. There are a few instances of false positives within the NMRooting annotation that can be attributed to users clicking on surfaces they did not intend to. It is important to note that during the eventual study task, no further assistance was provided unless prompted, as opposed to the

Table 1
Hypotheses tests, reported with statistic T/U, critical value p, effect size d and degree of freedom (dof). We indicated what statistical tests were used by their abbreviation, namely U for Mann-Whitney U-Statistical Test and T for T-Test.

	W-				**W+**			
H1 SUS (T)	t	p	d	dof	t	p	d	dof
	0.716	.491	0.245	9	2.526	.017	1.249	8
	T							
	t		p		d		dof	
	5.112		< .001		0.812		18	
	W-				**W+**			
H2 UEQ (T)	t	p	d	dof	t	p	d	dof
	2.442	.017	1.019	10	1.808	.054	0.987	8
	T							
	t		p		d		dof	
	0		< .001		-2.859		9	
	W-				**W+**			
H3 ∑ L (U)	t	p	d	dof	t	p	d	dof
	0	< .001	-2.859	9	4	< .001	-2.537	8
	T							
	t		p		d		dof	
	4		< .001		-1.654		18	
	W-				**W+**			
H4 |O^1| (U)	t	p	d	dof	t	p	d	dof
	44	.789	0	8	22	.034	-1.223	9
	T							
	t		p		d		dof	
	148.5		.355		-0.386		18	
	W-				**W+**			
H5 F1 (U)	t	p	d	dof	t	p	d	dof
	84	.011	1.179	9	81	< .001	4.426	8
	T							
	t		p		d		dof	
	336		< .001		1.749		18	
	W+ and W-							
H6 F1 (U)	U		P		d		dof	
	89.821		< .001		2.494		9	
	W-				**W+**			
H7 ∑L (U)	t	p	d	dof	t	p	d	dof
	-7.898	< .001	-2.859	18	-5.239	< .001	-2.537	16
	T							
	t		p		d		dof	
	-5.021		< .001		1.749		36	
	W+ and W-							
H8 d_ij (U)	U		P		d		dof	
	13		.7		0.346		18	

training task, which included guidance and repetition until no mistake was made.

This effect was exasperated by the presence of water noise. Some users corrected their annotation to a certain degree, resulting in fewer false positives but still suboptimal pathing, as seen in Fig. 7A. In contrast, a case of supoptimal pathing in VR, which directly results from a coarser user interaction, is seen in Fig. 7E. A total of 5 users attempted to separate the proximal roots, which yielded a few perfect annotations, but also artifacts such as Fig. 7B, which includes a few small roots that are due to participants progressing the lateral by clicking in smaller steps, such that the algorithm does not use the connecting signal between the roots to path to the point indicated by the user. If water noise was present, a few users misclicked into the water volume during the study task but failed to remove such a tracing at a later point, as seen in Fig. 7C.

While the VR application had better F_1 scores on average, ultimately more training is required for users to produce high-quality RSA reconstructions. Some participants struggled with depth perception in VR, which occurred in approximately equal parts in people with corrected vision and normal vision. This is likely an experience effect that would only be resolved by further use of the VR system - participants with previous VR experience did not encounter this. Targeting objects in the virtual scene as well as the correct placement of segments was challenging for new users.

The generally low variance of individual scores might be an indicator of a systematic effect that is uniform among individuals. In the case of VR, the inability to edit nodes was likely a large contributor to the overall score of participants. This caused issues in cases where there was a node missing in the taproot, as exemplified in Fig. 7D. In the case of

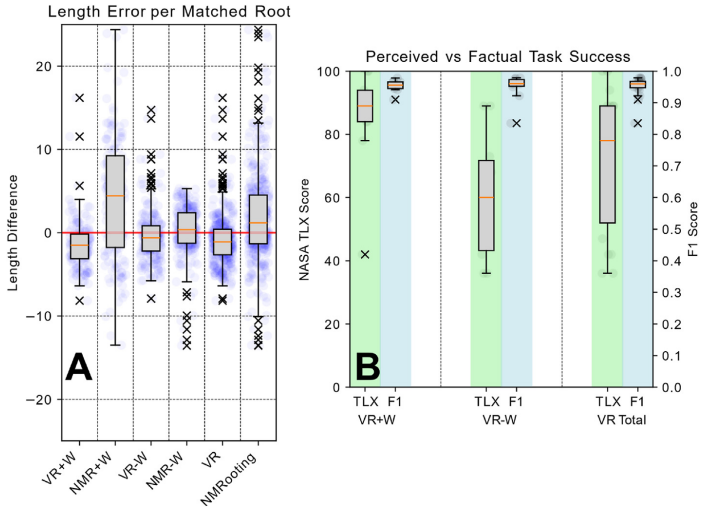

Fig. 8. A: Box-plots of length differences between ground truth and correctly annotated roots. Ground truth is red, median is orange, raw data is indicated blue, outliers are marked as x. B: Comparison of self-assessment of extraction quality in VR to actual extraction quality.

NMRooting, participants were told to manually trace the nodes and were able to delete nodes that were mistakenly inserted. The low score in NMRooting is mostly due to smaller effects accumulating to a lower score in total, including suboptimal pathing, misidentification of roots, as well as the presence of proximal roots and water noise in certain conditions.

4.4. Explanations for length differences

VR users tended to estimate the ground truth total root length correctly when dealing with noise-free data. However, there is a slight indication of overestimation of the individual root length within noisy data, as found in Fig. 6. VR users might not have drawn the taproot correctly, resulting in a slight overestimation of average root lengths, but still an underestimation of the total. It has to be noted that the task was performed sitting, which meant that users were forced to use the navigation in VRoot as an alternative to bending down. Some users chose to trace the root system less effectively as parts of it were out of reach, resulting in a higher variance of the total length measurements. A few users requested to be able to stand, but for the sake of uniformity, we did not allow this.

On the other hand, the large overestimation in average root length in NMRooting was in part due to supoptimal pathing, while the overestimation in total root length was caused by false-positive lateral roots, which is indicated by the shift in distribution in the inter-lateral distance, as seen in Fig. 6. We further investigate this by filtering the roots for only true positive identifications and subsequently compute their length difference, as shown in Fig. 8. While we do observe a higher-than- zero length extraction, by means of double-sided t-test with $t(198) = 2.379$, $p = .018$, $d = 0.169$, we observe a significantly higher variance in the +W condition of the desktop software (F (192) = 0.651 and $p = .001$). We believe that a combination of issues, most notably inability to effectively navigate in a desktop setting, caused the differences in the +W condition.

4.5. Subjective measures

The quantification of responses to the questionnaires yielded mixed results. Particularly, we want to highlight the increased usability through the use of VRoot. While there was no significant increase in the no-water condition, as seen in Fig. 4, there was a larger spread of responses for the NMRooting software, resulting in a higher average usability score of using VR in comparison. NMRooting generally had a larger spread in responses in contrast to the more unified responses for VR both in total (F (18) = 4.330 and $p = .002$) and in the +W condition (F (8) = 11.493 and $p = .002$). The TLXs yielded no significant difference between NMRooting and VRoot, but it did showcase a higher variance in the -W cases.

The user experience is challenging to compare, as certain measures (such as novelty) are not appropriate in the assessment of the desktop software. This is especially true since participants will already have the expectation of using VR software even if using the desktop software first, leading to influences measurements of those metrics.

The task, while the root was generally simple with only a taproot and 25 laterals, there were particularities about the data that caused issues for certain participants, especially when using NM- Rooting. However, artifacts like proximal roots, or smaller roots further down the taproot, have gone unnoticed to some participants. We will note that there was no difference in the individual ratings depending on the order of applications tested. Generally, it is quite natural that the tasks would appear equally demanding between the individual conditions. Interestingly, the VR + W condition has the smallest variance, indicating a more universal agreement even between the within-subject conditions.

One aspect of the TLXs questionnaire is the question on the self-assessment on whether the task was completed. This assessment is shown in Fig. 8B. The F_1 score shows that users performed similarly whether there was water noise present, or not. However, self-assessment of the accuracy was much lower than the actual accuracy with no water noise present. This is likely less a self-assessment and more a comparative assessment depending on how easy the problem appears to be solvable with automatic means. We highlight this to underline the issue that the subjective assessment of data extraction done by users is seldomly representative of the actual data quality. While our users estimated their own performance as worse than it actually was, this is a more general issue, as manually annotated public data sets also contain label errors, which generally are assumed to be perfectly labeled [41]. However, manual annotation work is incredibly valuable, especially in plant science. The human self-assessment of data quality measured through

manual means, especially if used as training basis, is seldomly accurate.

4.6. Usability of VR for annotation

Even if a virtual reality workflow improves the quality of extraction, manual tasks remain more tedious and time-consuming than automatic extraction. Our workflow is ultimately aimed at correcting rather than tracing. VRoot functions best with a mixture of high throughput pre-tracing and features that assist with the analysis of root architectures as the basis. In the future, we would like to combine automated, semi-automated, and manual tracing methods. In the case of NMRooting, this would require the introduction of additional interaction metaphors suitable for tasks that are voxel-based. There are other methods, such as TopoRoot [10], that could improve the manual extraction pipeline. This would be in line with published literature on similar topics, namely by Zeng et al. [17], who improved the VR application developed in Usher et al. [14] through the use of topological features.

Furthermore, there are technical aspects to consider. Geometry visualization is ultimately constrained by the physical device memory, which primarily includes GPU memory. While geometry reduction in terms of triangle merging can help, the complexity of the data through number of roots or noise will impact rendering performance. Resolution is also a factor, which is a device attribute of the MRI scanner. While we did not observe resolution-based effects that had clear origin, Selzner et al. [12] had already studied the effects of higher resolution (through upscaling) for manual annotation. Furthermore, using commonly available software such as Unreal Engine, it is possible to target a wider range of consumer-grade hardware, and the application is portable to different HMDs in principle.

5. Conclusion

In this work, we presented a pipeline to extract RSAs from MRI images using VR. We established the need for a more immersive manual analysis tool for complex data sets and showed the advantages of using VR to enable new users to achieve high-quality reconstructions faster. We evaluated the use of our VR software in comparison to contemporary desktop applications, and semi-automated analysis. Furthermore, we quantified how well participants with a uniform knowledge base performed in these tasks, both on the desktop as well as in VR. Our results show an increased usability and accuracy through the use of VR for manual root workflows, especially in instances where automatic tools need more assistance. This enables the analysis of root systems in more diverse soil conditions. Here, immersive annotation is a very valuable method in 3D root image analysis and helps to increase the variety of analyzed data to more soil types and soil water contents. In the future the goal is to combine the tools offered by our VR application with the benefits of an automatic extraction to provide a user-friendly and fast workflow to correct automatic tracing results. A repetition experiment specifically designed to estimate the actual errors made when manually extracting data would be needed to provide a robust quantification beyond relative comparison.

6. Ethical Statement

The study conception and planning got approval from the ethics committee of Trier University. Participants provided their explicit consent and were instructed on how to contact us to have their data deleted should they wish to do so. Data is stored and published anonymously.

Author contributions

This work was primarily authored by Baker. All authors discussed the results, provided feedback, and contributed to the text of this work. Study conception and execution was done by Baker and Zielasko. The implementation of the VR application and development of its features was done by Baker, Selzner, Göbbert, Zielasko, and Schnepf. Scientific counsel and directions for data analysis, study procedure, as well as results were provided by Selzner, Scharr, Riedel, Hvannberg, Schnepf, and Zielasko. Funding for hardware was provided by Schnepf and Göbbert. Zielasko has primarily supervised this project and guided implementation of its tasks.

Data availability

The data needed to reproduce the results of this work has been uploaded to 10.26165/JUELICH-DATA/B9SBOS. We have included a meta-description of the approach but encourage researchers aiming to reproduce our results to reach out. The Software VRoot will be found on Github: dhelmrich/VRoot. The questionnaires have been attached to the supplemental material. We have published a video description of the software, seen in 10.6084/m9.figshare.26003494.

Funding

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 Excel-lence Strategy, EXC-2070 - 390732324 - PhenoRob and by the German Federal Ministry of Education and Research (BMBF) in the framework of the funding initiative Soil as a Sustainable Resource for the Bioeconomy BonaRes, the project BonaRes (Module A): Sustainable Subsoil Management - Soil3; subproject 3 (grant 031B1066C).

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.plaphe.2025.100013.

A Normal Distribution Tests

The shapiro-wilk test for normal distribution tests whether the data is drawn from a normal distri- bution. This is a test where the alternative hypothesis is exhibiting a normal distribution, meaning that the H_0 hypothesis is that $X \sim N$ for some normal distribution N, referring to a significant difference to data produced by a normal distribution. For comparative measures, the residual of the two statistics, $R: = T_1 - T_2$, needs to be normally distributed, meaning that the difference between applications is tested as opposed to the statistics T of each application itself.

As seen in Tab. 4, there are some instances of data where a normal distribution is not present, which are the +W conditions for the root lengths,

resulting in a difference from a normal distribution for the total data. The inter-lateral distance d_i does not exhibit normal distribution for the -W condition.

These effects were likely caused by a non-uniform error that is present within the data, as opposed to the subjective measurements that are mostly centered around an average value, the objective distributions are non-symmetric regarding the differences between the applications.

Tab. 2
Goodness-of-fit tests for normal distribution on different measures, including combined measures. The pairing of residuals is always within subject.

Measure	Water		No Water		Total	
	t	p	t	p	t	p
SUS	0.902	.264	0.959	.783	0.944	.311
UEQ	0.939	.577	0.938	.497	0.897	.051
TLXs	0.933	.515	0.953	.713	0.976	.897
TRL	0.798	.019	0.870	.101	0.744	< .001
ARL	0.798	.019	0.870	.101	0.744	< .001
F_1	0.722	.002	0.931	.462	0.909	.072
d_i	0.915	.357	0.490	< .001	0.901	.051

Tab. 3
Goodness-of-fit tests for individual statistics for the testing against ground truth measure-ments, for each VR condition.

Measure	VR+W		VR-W		VR	
	t	p	t	p	t	p
SUS	0.902	.264	0.959	.783	0.944	.311
UEQ	0.932	.504	0.961	.790	0.953	.074
TLXs	0.877	.157	0.907	.259	0.913	.074
TLR	0.679	.001	0.777	.011	0.719	< .001
ARL	0.986	.988	0.879	.129	0.928	.159
F_1	0.908	.308	0.686	.001	0.720	< .001
d_i	0.756	.006	0.443	< .001	0.404	< .001

Tab. 4
Goodness-of-fit tests for individual statistics for the testing against ground truth measure-ments, for each NMRooting condition.

Measure	NMR+W		NMR-W		NMR	
	t	p	t	p	t	p
SUS	0.963	.824	0.884	.146	0.931	.160
UEQ	0.920	.394	0.918	.302	0.944	.289
TLXs	0.839	.043	0.924	.390	0.882	.019
TLR	0.744	.007	0.814	.014	0.714	< .001
ARL	0.930	.453	0.854	.083	0.885	.026
F_1	0.539	.007	0.979	< .001	0.890	.032
d_i	0.949	.687	0.771	.006	0.807	.001

B Study Procedure

In the questionnaire for the study, participants were informed about what data would be stored and how. Participants gave informed consent for the participation in the study. Participants were assigned an ID at the start of the study, and we measured their inter-pupillar distance for the purposes of calibrating the HMD to their respective measurements. Depending on the ID, participants then used either the application VRoot or NMRooting to extract a root system. For both applications, participants were verbally explained the functionality of the applications and what features they were to use to extract the root system. Each participant was tasked with performing all actions on a training data set that were required later on, which included navigation, tracing, and deletion in the case of NMRooting. After task completion with the study task, during which no intervention took place, participants filled out the questionnaires attached in the supplemental material. Participants will then repeat the process with the other application.

The study aimed to assess the direct interaction with the system in cases in which manual annotation was necessary. As such, participants were told to only use these tools to ensure comparability, as otherwise the task would be more complex, and user interaction would require more functionality and a deeper understanding of the application. In the VR application, participants were taught the draw root functionality as well as data set navigation techniques to ensure that all of the data set is reachable. Successful completion of the VR training task required the following steps.

1. Move head position in VR
2. Rotate the root system

3. Move the root system up/down
4. Select a node
5. Deselect a node
6. Draw a node
7. Deselect a node and draw a lateral

In NMRooting, we restricted the functionality to the tip-to-tree option, with the correct starting point already being set pre-emptively. Users did label the root topology manually. Successful completion of the NMRooting training phase required the following steps.

Fig. 9. Removed outlier data set after subject failed to trace the taproot. While the cause of this issue is unknown, the participant did trace a fully functional root system in the training stage of the task, which included being prompted to place nodes on the taproot of the root system.

1. Turn the view
2. Zoom in/out
3. Pan the view (lateral movement respective to the screen view)
4. Click into the data to add a root
5. Click into the data to delete a root
6. Open the root order menu
7. Label roots by their root order (here just taproot and first order lateral)

Our previous tests yielded an average completion time for the tasks of 45 min. From a previous test with the application with mixed, i.e., expert and non-expert, participants ($n = 16$), we concluded that 1 h would be sufficient for untrained participants. While we did not provide a timing nor a time for task completion in the individual conditions, we did allocate time slots for participants which were an hour long.

C Outlier

The outlier data set, exemplified by the VR performance of the participant. While the participant was able to annotate both the taproot as well as annotate lateral roots in the training task, the participant has not retained the knowledge of the annotation steps and produced an annotation (red) as shown in Fig. 9. It is unclear whether this was caused by the presence of water noise.

References

[1] J.A. Atkinson, M.P. Pound, M.J. Bennett, D.M. Wells, Uncovering the hidden half of plants using new advances in root phenotyping, Curr. Opin. Biotechnol. 55 (2019) 1–8, https://doi.org/10.1016/j.copbio.2018.06.002. Analytical Biotechnology.

[2] E.D. Rogers, P.N. Benfey, Regulation of plant root system architecture: implications for crop advancement, Curr. Opin. Biotechnol. 32 (2015), https://doi.org/10.1016/j.copbio.2014.11.015. Food Biotechnology • Plant Biotechnology:93–8.

[3] E.L. Fry, A.L. Evans, C.J. Sturrock, J.M. Bullock, R.D. Bardgett, Root architecture governs plasticity in response to drought, Plant Soil 433 (2018) 189–200, https://doi.org/10.1007/s11104-018-3824-1.

[4] T.H. Nguyen, T. Gaiser, J. Vanderborght, et al., Responses of field-grown maize to different soil types, water regimes, and contrasting vapor pressure deficit, EGUsphere 2024 (2024) 1–53, https://doi.org/10.5194/egusphere-2023-2967.

[5] M. Giraud, S.L. Gall, M. Harings, et al., CPlantBox: a fully coupled modelling platform for the water and carbon fluxes in the soil–plant–atmosphere continuum, silico Plants 5 (2023) diad009, https://doi.org/10.1093/insilicoplants/diad009.

[6] F.M. Bauer, L. Lärm, S. Morandage, et al., Development and Validation of a deep learning based automated minirhizotron image analysis pipeline, Plant Phenomics 2022 (2022), https://doi.org/10.34133/2022/9758532.

[7] D van Dusschoten, R. Metzner, J. Kochs, et al., Quantitative 3D analysis of plant roots growing in soil using magnetic resonance imaging, Plant physiology 170 (2016) 1176–1188, https://doi.org/10.1104/pp.15.01388.

[8] T. Tuomainen, A. Toljamo, H. Kokko, M. Nissi, Non-invasive assessment and visualization of Phytophthora cactorum infection in strawberry crowns using quantitative magnetic resonance imaging, Sci. Rep. 14 (2024), https://doi.org/10.1038/s41598-024-52520-7.

[9] X.R. Zhou, A. Schnepf, J. Vanderborght, D. Leitner, H. Vereecken, G. Lobet, Phloem anatomy restricts root system architecture development: theoretical clues from in silico experiments, silico Plants 5 (2023) diad012, https://doi.org/10.1093/insilicoplants/diad012.

[10] D. Zeng, M. Li, N. Jiang, et al., TopoRoot: a method for computing hierarchy and fine-grained traits of maize roots from 3D imaging, Plant Methods 17 (2021) Zeng2021, https://doi.org/10.1186/s13007-021-00829-z.

[11] J. Horn, Y. Zhao, N. Wandel, M. Landl, A. Schnepf, S. Behnke, Robust skeletonization for plant root structure reconstruction from MRI, in: 2020 25th International Conference on Pattern Recognition (ICPR), 2021, pp. 10689–10696, https://doi.org/10.1109/ICPR48806.2021.9413045.

[12] T. Selzner, J. Horn, M. Landl, et al., 3D U-net segmentation improves root system recon- struction from 3D MRI images in automated and manual virtual reality work flows, Plant Phenomics 5 (2023) 76, https://doi.org/10.34133/plantphenomics.0076.

[13] L. Stingaciu, H. Schulz, A. Pohlmeier, et al., In situ root system architecture extraction from magnetic resonance imaging for water uptake modeling, Vadose zone journal 12 (2013) 1–9, https://doi.org/10.2136/vzj2012.0019.

[14] W. Usher, P. Klacansky, F. Federer, et al., A virtual reality visualization tool for neuron tracing, IEEE Trans. Visual. Comput. Graph. 24 (2018) 994–1003, https://doi.org/10.1109/TVCG.2017.2744079.

[15] T. Chandler, M. Cordeil, T. Czauderna, et al., Immersive Analytics, in: 2015 Big Data Visual Analytics (BDVA), 2015, pp. 1–8, https://doi.org/10.1109/BDVA.2015.7314296.

[16] A. Fonnet, Y. Prié, Survey of immersive Analytics, IEEE Trans. Visual. Comput. Graph. 27 (2021) 2101–2122, https://doi.org/10.1109/TVCG.2019.2929033.

[17] T. McDonald, W. Usher, N. Morrical, et al., Improving the usability of virtual reality neu- ron tracing with topological elements, IEEE Trans. Visual. Comput. Graph. (2020), https://doi.org/10.1109/TVCG.2020.3030363.

[18] D. Zielasko, B. Weyers, T.W. Kuhlen, Travel your desk? An office desk substitution and its effects on cybersickness, presence and performance in an HMD-based exploratory anal- ysis task, in: 2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR), IEEE, 2019, pp. 1285–1286, https://doi.org/10.1109/VR.2019.8798068.

[19] Alex Vlachos, Advanced VR rendering. Game Developers Conference, 2015.

[20] H. Adams, J. Stefanucci, S. Creem-Regehr, B. Bodenheimer, Depth perception in augmented reality: the effects of display, shadow, and position, in: 2022 IEEE Conference on Virtual Reality and 3D User Interfaces (VR), 2022, pp. 792–801, https://doi.org/10.1109/VR51125.2022.00101.

[21] S. Baceviciute, T. Terkildsen, G. Makransky, Remediating learning from non-immersive to immersive media: using EEG to investigate the effects of environmental embeddedness on reading in Virtual Reality, Comput. Educ. 164 (2021) 104122, https://doi.org/10.1016/j.compedu.2020.104122.

[22] R.F. Adler, R. Benbunan-Fich, The effects of task difficulty and multitasking on perfor- mance, Interact. Comput. 27 (2014) 430–439, https://doi.org/10.1093/iwc/iwu005.

[23] S. Mystakidis, J. Besharat, G. Papantzikos, et al., Design, development, and evaluation of a vir- tual reality serious game for school fire preparedness training, Educ. Sci. 12 (2022), https://doi.org/10.3390/educsci12040281.

[24] J.R. Lewis, J. Sauro, The factor structure of the system usability scale, in: International Conference on Human Centered Design, Springer, 2009, pp. 94–103, https://doi.org/10.1007/978-3-642-02806-9_12.

[25] P.T.K. Aaron Bangor, J.T. Miller, An empirical evaluation of the system usability scale, ternational Journal of Human–Computer Interaction 24 (2008) 574–594, https://doi.org/10.1080/10447310802205776.

[26] D. Pflugfelder, R. Metzner, D van Dusschoten, R. Reichel, S. Jahnke, R. Koller, Non-invasive imaging of plant roots in different soils using magnetic resonance imaging (MRI), Plant Methods 13 (2017) 102, https://doi.org/10.1186/s13007-017-0252-9.

[27] S le Gall, D van Duschoten, A. Lattacher, et al., Investigating the impact on water fluxes and physiological development of the combination of contrasted root wheat cultivars from isotopic analysis, in: EGU General Assembly 2024, 2024, https://doi.org/10.5194/egusphere-egu24-15966.

[28] R. Metzner, A. Eggert, D. Dusschoten, et al., Direct comparison of MRI and X-ray CT technologies for 3D imaging of root systems in soil: potential and challenges for root trait quantification, Plant Methods 11 (2015) 17, https://doi.org/10.1186/s13007-015-0060-z.

[29] Y. Zhao, N. Wandel, M. Landl, A. Schnepf, S. Behnke, 3D U-Net for Segmentation of Plant Root MRI Images in Super-resolution, 2020, https://doi.org/10.48550/arXiv.2002.09317.

[30] A.O. Uzman, J. Horn, S. Behnke, Learning Super-resolution 3D Segmentation of Plant Root Mri Images from Few Examples, 2019, https://doi.org/10.48550/arXiv.1903.06855 arXiv preprint arXiv:1903.06855.

[31] G. Lobet, M.P. Pound, J. Diener, et al., Root system Markup Language: toward a unified root architecture description language, Plant Physiology 167 (2015) 617–627, https://doi.org/10.1104/pp.114.253625.

[32] D. Zielasko, B.E. Riecke, Sitting or standing in VR: about comfort, conflicts, and hazards, IEEE Computer Graphics and Applications 44 (2024) 81–88, https://doi.org/10.1109/MCG.2024.3352349.

[33] W. Schroeder, K. Martin, W. Lorensen, The design and implementation of an object-oriented toolkit for 3D graphics and visualization, in: Proceedings of Seventh Annual IEEE Visualization '96, 1996, pp. 93–100, https://doi.org/10.1109/VISUAL.1996.567752.

[34] D.N. Baker, F.M. Bauer, M. Giraud, et al., A scalable pipeline to create synthetic datasets from functional–structural plant models for deep learning, silico Plants 6 (2023) diad022, https://doi.org/10.1093/insilicoplants/diad022.

[35] B. Laugwitz, T. Held, M. Schrepp, Construction and evaluation of a user experience ques- tionnaire, in: Holzinger A. Berlin (Ed.), HCI and Usability for Education and Work, Springer Berlin Heidelberg, Heidelberg, 2008, pp. 63–76, https://doi.org/10.1007/978-3-540-89350-9_6.

[36] S.G. Hart, L.E. Staveland, Development of NASA-TLX (task Load index): results of empiri- cal and theoretical research, in: P.A. Hancock, N. Vol 52 Meshkati (Eds.), Human Mental Workload, Advances in Psychology, North-Holland, 1988, pp. 139–183, https://doi.org/10.1016/S0166-4115(08)62386-9.

[37] D. Zielasko, B.E. Riecke, To sit or not to sit in VR: analyzing influences and (Dis) Advantages of posture and embodied interaction, Computers 10 (2021), https://doi.org/10.3390/computers10060073.

[38] A. Schnepf, K. Huber, M. Landl, F. Meunier, L. Petrich, V. Schmidt, Statistical characteriza- tion of the root system architecture model CRootBox, Vadose Zone J. 17 (2018) 170212, https://doi.org/10.2136/vzj2017.12.0212.

[39] J.R. Lewis, The system usability scale: past, present, and future, Int. J. Hum. Comput. Interact. 34 (2018) 577–590, https://doi.org/10.1080/10447318.2018.1455307.

[40] J. Cohen, A power primer, Psychol. Bull. 112 (1992) 155–159, https://doi.org/10.1037/0033-2909.112.1.155.

[41] C.G. Northcutt, M. ChipBrain, C.A. Athalye, J. Mueller, Pervasive label errors in test sets destabilize machine learning benchmarks, in: 35th Conference on Neural Information Processing Systems (NeurIPS 2021), 2021, https://doi.org/10.48550/arXiv.2103.14749.

Article III

Adapting Agricultural Virtual Environments in Game Engines to Improve HPC Accessibility

Dirk Norbert Baker, Felix Bauer, Andrea Schnepf, Hanno Scharr, Morris Riedel, Jens Henrik Göbbert, Ebba Hvannberg

DOI: 20.500.11815/4936 2024

Submitted to NeIC2024 in April 2024, Accepted May 2024

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

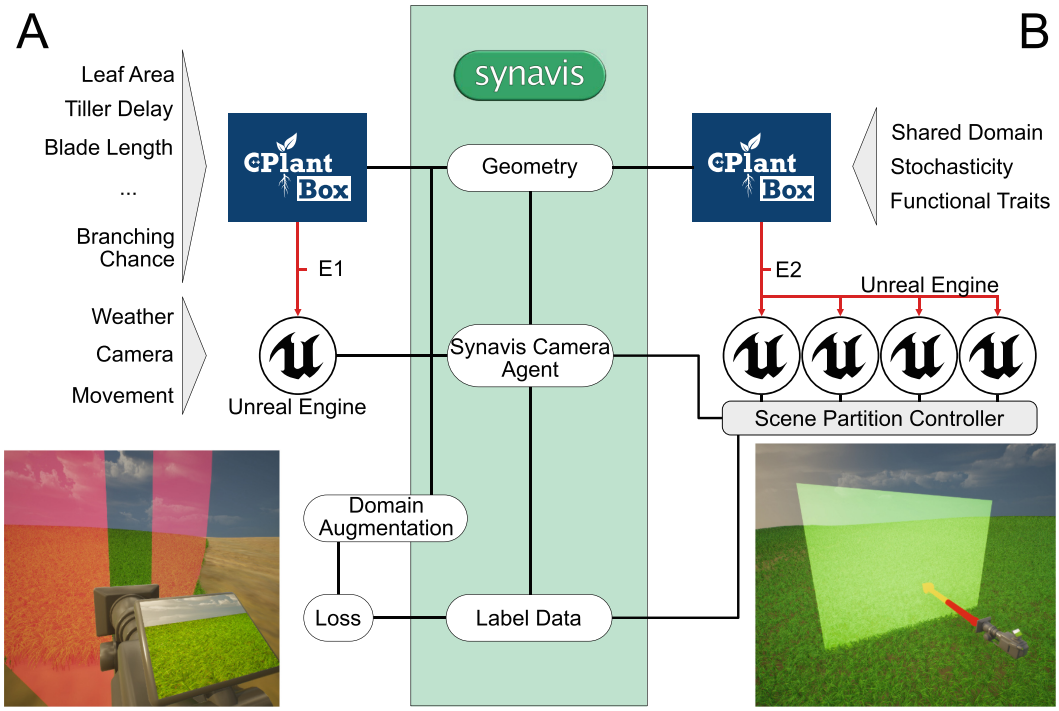

Fig. 1. Setup of pipelines in different distribution techniques. Measurements are highlighted and are also implemented in Synavis. The listed parameters are the primary steering parameters for the individual scenario. A. Continuous update with the intended use-case of adaptive data generation. B. Scene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

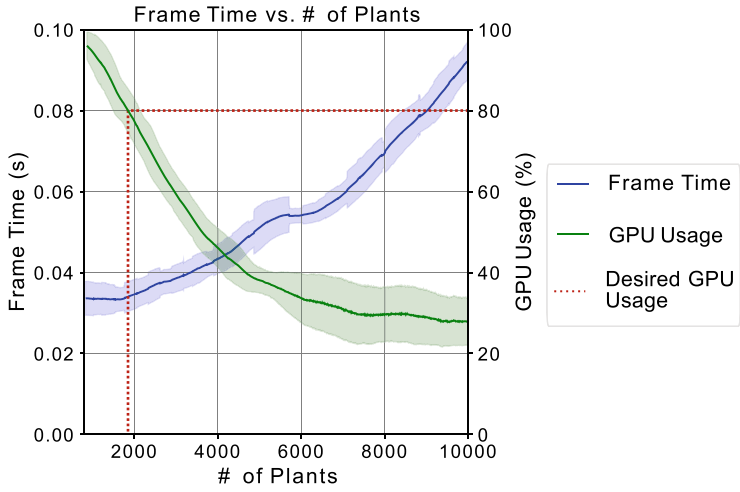

Fig. 2. Frame time measurement average for a field of N plants that are being continuously updated.

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.

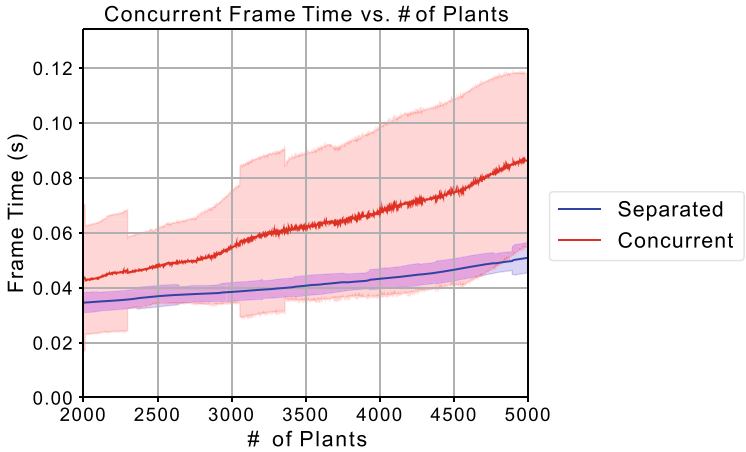

Fig. 3. Impact of rendering $N * 4$ geometries in UE concurrently on the same node as opposed to different nodes.

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

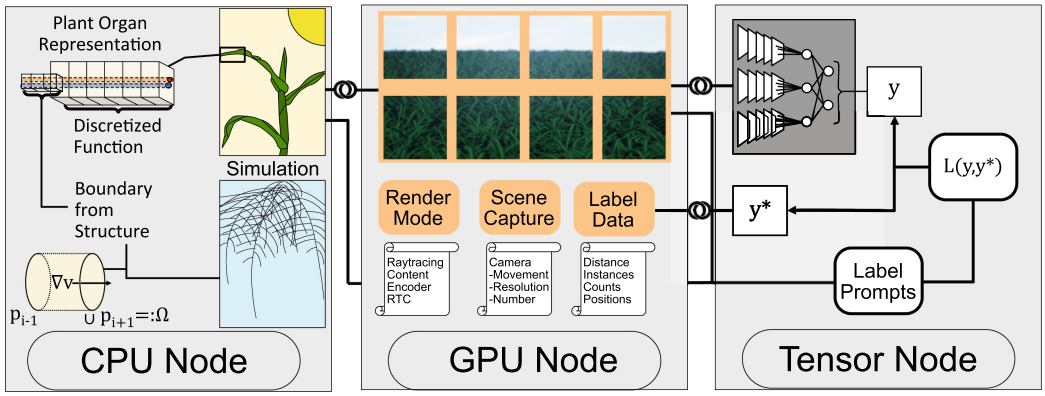

Fig. 4. Overview of technical components and data flows. The illustrated assignment to specific nodes is a performance recommendation, though individual components can share resources. A linked line indicates concurrent coupling.

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.: UnreaLCV: 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

Article IV

Virtual World Coupling with Photosynthesis Evaluation for Synthetic Data Production

Dirk Norbert Baker, Mona Giraud, Jens Henrik Göbbert, Hanno Scharr, Morris Riedel, Ebba Hvannberg, Andrea Schnepf

DOI: i10.1101/2025.02.06.633870 2025

Submitted to in silico Plants in February 2025

Virtual World Coupling with Photosynthesis Evaluation for Synthetic Data Production

Dirk N. Baker[1*], Mona Giraud[2], Jens Henrik Göbbert[3], Hanno Scharr[4], Morris Riedel[1,3], Ebba Þóra Hvannberg[1], and Andrea Schnepf[2]

[1]School of Engineering and Natural Sciences, University of Iceland
[2]Institute of Bio- and Geosciences 3: Agrosphere, Forschungszentrum Jülich GmbH
[3]Jülich Supercomputing Centre, Forschungszentrum Jülich GmbH
[4]Institute of Advanced Simulation 8: Data Analytics and Machine Learning, Forschungszentrum Jülich GmbH
[*]Address correspondence to: dnh2@hi.is

February 6, 2025

Abstract

In this work, we couple the functional-structural plant model CPlantBox to the Unreal Engine by exploiting the implemented raytracing pipeline to evaluate light influx on the plant surface. There are many approaches for photosynthesis computation and light evaluation, though they typically are limited by versatility, compute speed, or operate on much coarser resolutions. This work specifically addresses the concern that data generation pipelines tend to be unresponsive and do not include model-based knowledge as part of the generation pipeline. Using established photosynthesis solvers, we model the interaction between the Unreal Engine and the FSPM to measure physical properties in the virtual world. This is successful if we are able to reproduce experimental results using an in silico model. As part of the pipeline, we generate a surface geometry and utilize material shaders that are designed to establish a baseline surface model for light interception and transmission, based on simple parameter sets that can be calibrated. Using a Selhausen field experiment as baseline, we reproduce the photosynthesis effectiveness of the plants in the 2016 winter wheat experiments. Our pipeline is deeply intertwined with data generation and has been proven to perform well at scale. In this work, we build on our previous work by showcasing both a simulation study of a light evaluation as well as quantifying how well our system performs on high-performance computing systems.

1 Introduction

Simulating the real world is a pathway to deeper understanding of its mechanisms. This is especially true in plant science, where many individually distinct processes jointly form emergent behavior of a plant (Thornley 2011; Walia et al. 2024). Here, studying the influences of the environment, or of plant properties, becomes crucial to fully explore crop behavior as agriculture becomes more sustainable and resilient. Having access to a model of the real world, particularly in cases for which reference experimental data is available but sparse, can assist with filling gaps in the data. A use-case for this is to map the experimental results onto synthetic data for deep learning (DL) models. Particularly in plant sciences, where image analysis is a critical tool and the primary bottleneck (Minervini et al. 2015; Taiz 2013; Tsaftaris et al. 2016), using model-based data sources

can uncover hidden mechanisms. A faithful version of an experiment would include a calibrated structure of a plant, producing both images and dense training labels. The potential to uncover systematic image features that plants with a certain trait exhibit, is the first and most important step towards training models that detect such traits remotely. Embedding models into the data generation pipeline is especially impactful as a common critique of data production pipelines is their unresponsiveness and one-directional character, e.g. as reported by Li et al. (2024). In this work, we showcase a method to embed and communicate with the virtual world in a way that does not only produce more useful data labels for a concrete problem such as photosynthesis, but also explore the expressiveness of our virtual world model to accommodate these models. Functional-structural plant models (FSPMs), such as CPlantBox (Giraud et al. 2023) seldomly represent only one functional process and are instead a coupling of different models (Sievänen et al. 2014). Plant structural definitions can inform boundaries of plant processes, such as leaf surface, or root-shoot-interface, where e.g. flux parameters can be defined that are communicated between the organs. A key feature, however, is the fact that the plant structure is incorporated explicitly rather than using heuristics to signify plant behavior. Though heuristics can be established using plant simulations, particularly for upscaling as investigated by Vanderborght et al. (2024), plant simulations are typically performed to gain insights into plant mechanisms.

Photosynthetic simulations with stomatal regulation generally solve a fixed-point iteration (Von Caemerer 2013) and the reactions involved in the uptake of co_2 are solved on the basis of a steady reaction flow limited by availability of nutrients (Giraud et al. 2023) or stomatal metabolism (Yin et al. 2004). The CPlantBox module for photosynthesis, developed by Giraud et al. (2023), combines different models (Farquhar et al. 1980; Leuning 1995; Tuzet et al. 2003) into a whole-plant framework that couples the water and carbon dynamics from root to leaf. Although the model can use specific input data for each leaf section, Giraud et al. (2023) only used mean atmospheric data as basic heuristical approach. Models that calculate the actual light influx in a plant involve radiative transfer modeling (RTM), which describes the light propagation and eventual impact onto the surface of the plant. RTM has standardized concepts (John V. Martonchik and Strahler 2000; Nicodemus et al. 1977) and is implemented in a variety of models, particularly DART (J. P. Gastellu-Etchegorry and Gascon 2004). For the retrieval of functional properties from remote sensing data, empirical models such as LESS developed by Qi et al. (2019) and extended e.g. by Zhao et al. (2024), have been successfully implemented. Another example of a high-quality model for radiative transfer available as open-source is `prospect`, developed by Féret and Boissieu (2024). Worth noting is that similar concepts and models have been implemented when a rendering technique called *physics-based rendering* was established, and the similarity between the two fields regarding this specific use-case is of particular note, as described by Salesin et al. (2024).

In experiments to parameterize FSPMs, particularly the measurement of the structural portion can yield much information on the performance of the crop. Diversity of structure, e.g. in root systems, has been identified to be one of the primary indicators of adaptation of a crop to its environment by Yu et al. (2024). As such, many data generation pipelines make use of a plant's structure to accommodate variability (Bailey 2019; Baker et al. 2023; Ward et al. 2018). The key factor for the embedding of an FSPM for data production is a reasonable reproduction of the target space, which is RGB images for many applications. In our pipeline, we are using the Unreal Engine (UE) to compute the virtual world. UE is a multimedia 3D graphics engine, which has seen use across industries, including its origin in gaming, simulation science (Agarwal et al. 2023), plant science (Li et al. 2024), and virtual reality (Krüger et al. 2024). Our own recent work brought forth the Synavis framework, a library designed to dynamically setup, steer, and measure within the UE virtual world (Baker et al. 2023, 2024).

An embedding recently implemented in the HELIOS pipeline by Lei et al. (2024) shows the potential for pipelines to accommodate functional information. We believe that the inclusion of an FSPM that can simulate functions in the whole plant (roots and shoot) and at a sub-organ level provides a valuable extension to this by allowing the models to communicate between domains (soil

- plant - atmosphere) and scales (plant section - field). Thus, in this work, we extend the Synavis framework to work together with the photosynthesis module of CPlantBox, which until recently employed a uniform light influx heuristic. Our implementation of this pipeline uses the physics-based rendering employed in UE to simulate light exposure to achieve an accurate light exposure computation for CPlantBox, which we validate by modeling a digital twin of an experiment at the Field Minirhizotron Facility in Selhausen. The data-based pipeline that is enabled in Synavis is extended by using a model-view which takes physical properties to allow radiative transfer modelling and a data view that is able to map functional photosynthesis data onto a synthetic data pipeline, which we showcase by the example of light exposure efficiency. In our replication, we make use of experimental data provided by the TERENO platform (Bogena 2016; Reichenau et al. 2020) to re-create a field experiment while also retaining a high level of structural and functional information through the use of a full-scale plant model. Lastly, our pipeline is HPC compatible, with a focus on ensuring that parallelization is feasible, and we include full scaling experiments with this manuscript.

2 Methods

We will be highlighting selected aspects of the FSPM along with the geometric embedding into the virtual scene as part of our solution to compute light influx. Note that for certain parts, this paper includes expressions and terms that are otherwise written as render code, and for further resources, we need to refer to the open-source implementation of UE[1].

2.1 Virtual World Generation in Synavis

We used the Synavis framework to create a virtual world in UE. An in-depth description of the Synavis framework can be found in our recent paper (Baker et al. 2023). Synavis is a framework for the coupling of simulations with virtual environments, allowing the sampling and measuring of values through the framework for the purposes of training neural networks. In this work, we extended this functionality by allowing the FSPM simulation to access data from the virtual environment directly, allowing the superimposition of plant function onto the virtual environment and the subsequent measurement physical properties in the environment. Within UE, we used the base rendering framework as well as the entity-component system implemented in UE. We developed a framework to measure light intensity, a pipeline to embed model data into UE to generate synthetic data based on FSPM simulation results, as well as a pipeline to setup and manage distributed virtual scene handling through Synavis.

Our setup has three distinct aspects: the CPlantBox simulation, the UE model scene, and the resulting visualization. These aspects of the setup can be separated and communicated asynchronously, but in our setups we typically assume that each application is running concurrently (Baker et al. 2024). The field setup configures and runs the CPlantBox simulations. The visualizer carries vertex-triangle information that can be used to map functional properties. Through Synavis, plant geometries are inserted into the virtual scene once every time step. A texture containing surface properties is being generated per organ because of how the texture wraps around the plant, see Fig. 1 and Sec.2.2. Light influx measurement is done after the radiation calibration, submitted as point measurement command to the `ALightMeter` class in UE, which carries out the measurement. Necessary parameters are meter resolution, measurement delay to accommodate scene update and measurement duration to accommodate path tracing. Path tracing is a UE term that refers to the physics-based raytracing of image pixels without heuristic. UE does implement a heuristic for raytracing, called Lumen, which uses a reduced-detail scene to complete the light information and might not correctly reflect the physical properties of the surfaces, as reported by Agarwal et al. (2023) in a remote sensing context. The measurement is passed back through Synavis as influx per segment,

[1] Unreal Engine Source Documentation, account needed for source code access

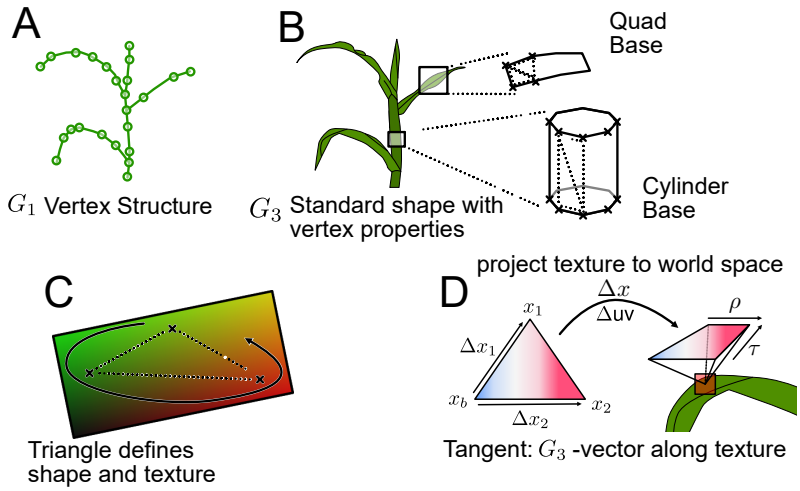

Fig. 1: Relationship between functional properties on the texture level, according to the mapping employed by Synavis (Baker et al. 2023). **A.** We denote G_1 to be the graph structure produced by the FSPM, including diameters, organ shape parameters, as well as functional aspects. **B.** Geometrization is done through shape extrusion, taking uniform shapes as template and placing them in a way that they encase the plant. Triangulization is the connection between the shapes that are used to build the geometry. **C.** On top of a triangle, the texture map is further defined on each vertex. **D.** To be able to fully map the texture onto the geometry, the tangents need to be computed, which indicate for the texture is wrapped locally around the geometry, being as long as one pixel is wide on that vertex.

which is passed to the Photosynthesis module. The parametrization is entirely on the Python side, which retains all functional parameters and values. Mapping of the photosynthesis output is done exactly like the mapping of functional properties.

2.2 CPlantBox Model Embedding

CPlantBox produces the structure of a plant, using a graph ($G_1 := (V, E, P)$) that encodes one-dimensional properties along the edges. G_1 is made of vertices (V) connected by edges (E) and stores a set of properties (P) defined on either the edges or the vertices. This structure can be seen outlined in Fig. 1.A. Our embedding typically uses an inferred 3D structure, called G_3, which contains shape information informed by G_1. G_3 is a meshing based on shape representations (cylinder or quad) that is implicitly generated from vertex positions and organ type, as shown in Fig. 1.B. We make certain assumptions to define the output geometry, e.g., the leaf blades are perpendicular to the branching direction, and remain perpendicular to successive vertex differences. This can be calibrated empirically through the CPlantBox visualizer (Baker et al. 2023). This construction means that in most circumstances the properties on the surface triangles are defined through the edge in G_1 they were constructed from. We interpolate properties to vertex level and implicitly further by encoding properties into textures, seen in Fig. 1.C, which can be accessed through tangent information like Fig. 1.D.

UE uses physics-based rendering, utilizing a bi-directional radiative transfer function (BRDF) $f(\omega_i, \omega_j)$, which yields the outgoing light intensity for the angle ω_j according to the incident angle

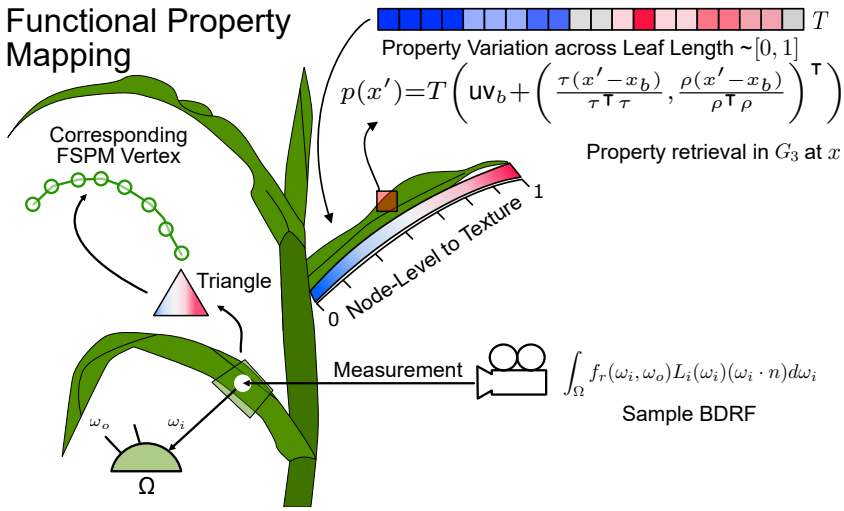

Fig. 2: Matrix-notation description on how the functional properties are being sampled by the rendering pipeline, if defined through the Synavis framework in the prescribed way.

ω_i. The function f is defined either through geometric or texture properties encoded in the Material shader that dictates how the surface reacts to light. A surface can also emit light, and the final color at the hit point x' where the pixel ray intersects the surface will include both direct emission as well as reflectance from other surfaces' emission (indirect light). The reflectance of a surface is represented semi-empirically via a single factor, representing implicitly the interaction of the light with the different layers of an object. As we are measuring the brightness, we are reversing certain aspects of the light simulation - e.g., a surface that would have absorbed a band of the light spectrum would now be brighter instead. This was implemented as such mostly for convenience, allowing the values to be representing of the actual absorption rather than its inverse. The BRDF is sampled across the incident angles of possible other light source contributions.

The sampling range is locally hemispheric, but the effective brightness is scaled with the surface normal. The surface normal n is interpolated from its vertex definition much like other properties (cf. Fig. 1.C). The space of surface angles Ω needs to be thoroughly sampled to gain a fully informed view of the scene brightness, also illustrated in Fig. 2. While there are certain optimizations implemented in Lumen, there are no intrinsic tricks that can be utilized to speed up the sampling, as it is entirely dependent on the geometric content of the scene. Texture properties rely on a sparse map $\text{UV} : G_3 \ni x \mapsto p \in \mathbb{R}^2$, illustrated on triangles of G_3 in Fig. 1.C, showing potentially different scales between vertices and texture maps. As described in Baker et al. (2023), the texture mapping of the FSPM structure fills the space of the texture map, thus resulting in a stretching of the properties contributing to the transfer function. Since the texture mapping is done on vertex level, but the rendering will densely prompt for surface properties at every location, the evaluation of a specific point x' relies on the texture map. Fig. 1.D shows how the texture (and thus functional) information is wrapped around the geometry, while the corresponding value retrieval for the texture is shown in Fig. 2. This estimation is linear, can be done before rendering (thus saving the tangents at vertex level) and allows for seamless stitching of a texture covering unconnected surfaces by explicit tangent declaration. The computation of a value from the functional properties at a specific surface location is described in Fig. 2. The primary surface properties[2] that are implemented in UE that are of

[2]Unreal Engine official documentation on the topic of Physically Based Materials, Accessed February 6, 2025

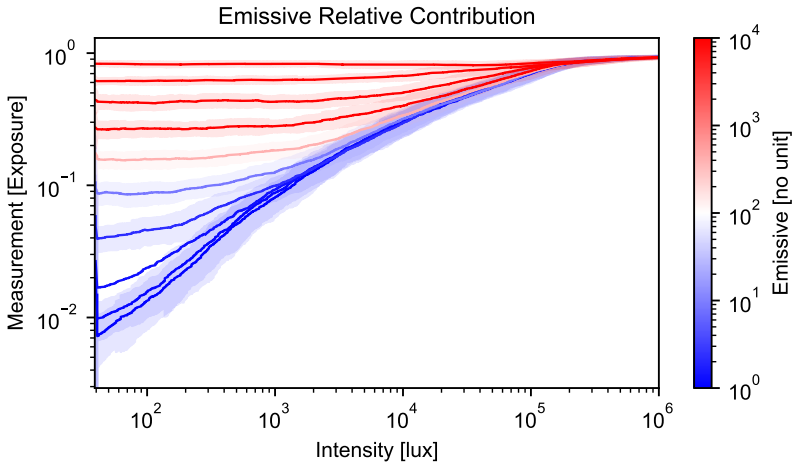

Fig. 3: Characterization: relative contribution of emissive property on the BRDF rendering of the surface model in UE, depending on the brightness of the directional (sun) light. The measurement is non-scaled, meaning that the initial rendering system will output to a uniform $[0, 1]$ range. On the x axis, the intensity is provided in *lux* units. We generally calibrate against an above-field sensor that is set to a direct exposure measurement reference.

interest are diffuse, which is set to grey for the purpose of measurement, specular, roughness, and auxiliary properties such as subsurface, translucency, or masking. Evidently, our implementation uses strong heuristics that need continuous verification by full-spectrum models for radiative transfer. We break down the most relevant contributions to photosynthesis and light absorption to material properties. The diffuse property (set to $(0.5, 0.5, 0.5)$) is a measurement tool that does not contain any physical relevance. However, although we are using a single heuristic, one could separate three channels for the transfer model. The virtual world model that is being used here is shown in the flowchart in Fig. 4.

Emissive properties, such as chlorophyll fluorescence, are added to the total luminescence in the scene. This effect is smaller compared to the direct light influx in the scene, as characterized in Fig. 3. This characterization is primarily done to ascertain whether the impact of the emissive property can be calibrated. Surface light emission is being added outside of the hit-based sampling of the surface, meaning that the initial contact of the measurement ray yields a base luminescent contribution by the plant surface. In certain wavelengths of light, this behavior contributes to the total light absorption of the surface of the plants. As shown in Miao et al. (2018), this effect can have significant impact on the absorbed PAR rate ($mol/m^2/s$) of the plant. Fig. 3 shows that the base emissive contribution will increase the base measured value, extinguishing in terms of impact with the increased directional intensity.

2.3 Domain Partition and HPC

Since we are aiming to simulate field-scale setups, the actual workload of simulating and measuring in the virtual environment needs to be distributed. For this purpose, we utilize field partitioning by allocating instances of Unreal Engine dynamically through Synavis (Baker et al. 2024). Field distribution is based on direct partitioning. Because CPlantBox uses stochastic structures that are

random-number-based, we assign a specific starting seed to the field in total, varying the individual plants deterministically. This ensures that the specific realization of a plant organ does not need to be communicated between nodes, as all nodes have implicit knowledge about the neighboring structures and could access their specific realization by simulating the corresponding field seed + plant position. All simulations that are being run have unique plant structures, but these structures can be inferred from the starting time of the cluster job.

We implement our field distribution for the JURECA-DC (Thörnig 2021) cluster, equiped with four NVIDIA A100 GPUs in addition to two AMD EPYC 7742 CPUs. These nodes are connected with InfiniBand Interfaces (NVIDIA Mellanox Connect-X6). Within each node, the field consists of a realization of a grid of plants, growing in a separate CPlantBox configuration. It is entirely optional, and independent of the virtual scene, whether the boundaries of the individual FSPM instances are being coupled, or not. In fact, as we are measuring the light influx on the geometry in the scene, the competition and individual exposure of individual plants can be estimated without running the FSPM in a continuous system. Nevertheless, for a full view of the plant field, a continuous system of plants, particularly regarding their soil properties, would be necessary.

The field-scale embedding further needs to be distributed over nodes to ensure efficient computation of plant instances. This is the case for all field-scale setups, irrespective of the soil-related simulation setup, since the amount of individual segments that need to be measured exceeds the memory of common GPUs (Baker et al. 2024). Our scaling experiment, results of which are shown in Sec. 3.3, consists of a partitioned field using a total number of plants N^2 arranged in a square. Based on total amount and MPI world size W, this square is subdivided into sets of squares of N/W plants per side. The seed numbers act as IDs for the plants, with the added randomness introduced by the starting time. The simulation can use one coherent global starting seed that is concatenated with the local ID of the plants. The grid of plants is scaled with the world embedding scale (which can be calibrated if needed for numerical precision) and the plant density. However, custom placements of plants are generally encouraged when producing light simulations, as our partitioning setup uses simple idioms to distribute the simulation as a showcase for HPC partitioning.

Our partitioning and scaling experiments are based on a plant grid that is being set and defined remotely. There are many hyperparameters that contribute to performance and results. In our tests, the total virtual screen resolution, i.e., number of light meters, is a factor in overall performance. For the hard scaling, we tested the progression using three steps of meter count: $50 \mapsto 3200$Pixels, $100 \mapsto 6400$Pixels, and $200 \mapsto 12800$Pixels. Virtual resolution values outside of these yield either decreased render or measurement performance. We increase the number of nodes, resulting in an increased field size or an increased distribution of the field size. In terms of UE rendering configuration, we do not utilize culling and will keep the reference point for render optimization[3] close to the target for measurement. When path tracing[4] is employed, lightmap settings do not matter, and we are not utilizing virtual textures[5], though it would not impact the measurement if this was added.

2.4 Embedding of Experimental Data

Our reference data has been taken from the Field Minirhizotron Facility in Selhausen[6] (Lärm et al. 2023; Nguyen et al. 2024). The facility is an experimental site that includes field data as well as extensive sensory data, including a minirhizotron setup that includes tubing to measure root distribution (Bauer et al. 2022). These data sets are hosted on the TERENO platform (Bogena 2016), which is an open data platform that enables us to access sensory as well as experimental data across parts of central Europe.

[3] Occlusion Culling Description
[4] Path Tracing Description
[5] Virtual Texturing Description
[6] 50.865°N, 6.447°E, 203m a.s.l. in 2024

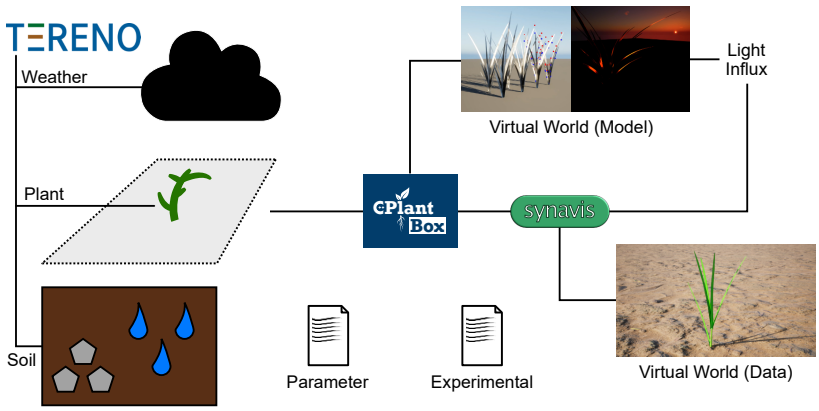

Fig. 4: Data sources and pathways of the UE virtual world coupling for photosynthetic radiation influx measurement. CPlantBox is the central data source for UE, while on the left are static data sources. The model world provides light influx, which can be mapped onto the data world.

Weather is introduced both as environmental condition influencing the atmospheric boundary of the photosynthetic reaction and as precursor to the quantization of other parameters. From top to bottom, air pressure and humidity, temperature, and radiation are being derived from Selhausen measurements. These values are partially resolved at the 10-minute rate. Using the calibration in Fig. 2, we can linearly scale the light measurement using a clearsky measurement value as reference. This value can stem from singular measurements done as reference, or a densely resolved radiation flux measurement as present in the Selhausen above-ground data. From a data generation standpoint, PAR radiation can be directly mapped onto the pixel space by using the relative radiance at each point without needing to consult measurement data. An illustration of this approach is shown in Fig. 4. We vary the directional light brightness based on the `RadiationGlobal` $[W/m^2]$ data collected in the Selhausen facility, as illustrated in Fig. 4. Our site data was partially extracted from the TERENO data hub (Bogena 2016) for the climate station Selhausen 3 (Schmidt 2024). The soil characterization was done in Lärm et al. (2023), which features matric potential measurements at different depths. The soil parameters were implemented as one-dimensional model for simplicity, while the root structure was simulated explicitly. We chose not to infer air co_2 molar fraction from the experimental measurements as the sensors we would have available are not representative of the chamber measurements in Nguyen et al. (2024). Nevertheless, the full weather model can be referenced to in the simulation script, see Sec. 5.4.

For the 2016 measurement, winter wheat was sown on October 26th, 2015, while first emergence was logged at November 1st, 2015. Nguyen et al. (2024) describe the gas chamber measurement in the field regarding the photosynthetic activity of the plants. The measurements were done at multiple time of days, and the PAR was always measured concurrently, along with other indicators. Our comparison focuses on the PAR adsorption and the co_2 uptake. We assumed the start of the growing season to be 5th March 2016 and simulate the plant growth up to the measurement, at which time we run the photosynthesis module in combination with the UE measurement of light influx. We are using a pre-flowering gas chamber measurement for the photosynthesis activity, including a sunlit wheat plant. We measure the light photon influx per segment on the leaf surface. This value has been relayed to CPlantBox, following a photosynthesis evaluation with simplistic soil coupling. We assumed no limitation by soil water content. The leaf surface parameters are kept simple, not including subsurface scattering or fluorescence. These values are included in the model, but

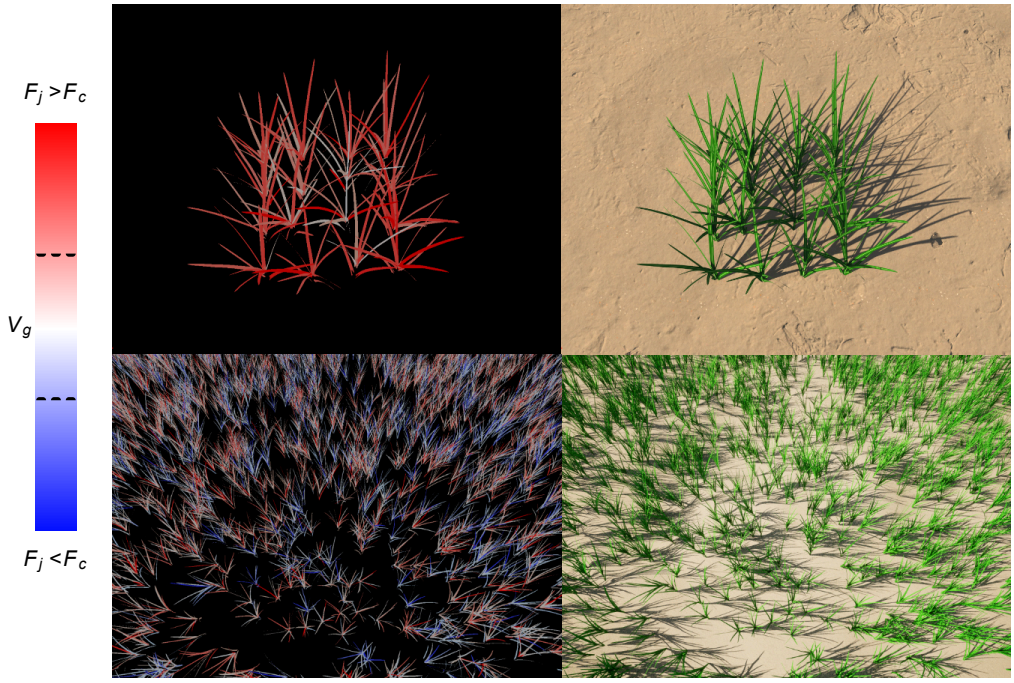

Fig. 5: Mapping of functional properties, in this case light assimilation effectiveness, onto the leaf area of the FSPM simulation. These values are scaled mostly categorical, with the red/blue distinction describing the flipping point between nitrogen limitation A_n and photoelectric limitation V_j.

need to be parameterized, and our TERENO-based parametrization is illustrated in Fig. 4. Both the virtual world models are steered through Synavis and implemented in UE, while CPlantBox runs as a precursor to the data scene, needing the model scene as light influx reference. In UE, the measurement was done over 3 frames following a 2 frame waiting period after the measurement point has been registered. This is to ensure that the virtual scene is fully updated before the measurement begins.

3 Results

Model coupling is largely based on the combination of the virtual world embedding (Baker et al. 2023) and the photosynthesis calculation developed by Giraud et al. (2023). Here, we highlight our results regarding the primary evaluation of our coupling's main contributions. This section focuses on the accuracy of the framework, the scalability of its implementation, as well as the production of synthetic light exposure training data from FSPM simulations.

3.1 Synthetic Light Exposure Data

Mapping the functional data onto a virtual measurement space ensures the increased usefulness of the photosynthesis data. It allows the extension of the synthetic data pipeline to uncover mechanistic insights into plant development in a pipeline otherwise restricted to other information that is being rendered in the engine, such as implemented by Mousavi et al. (2020).

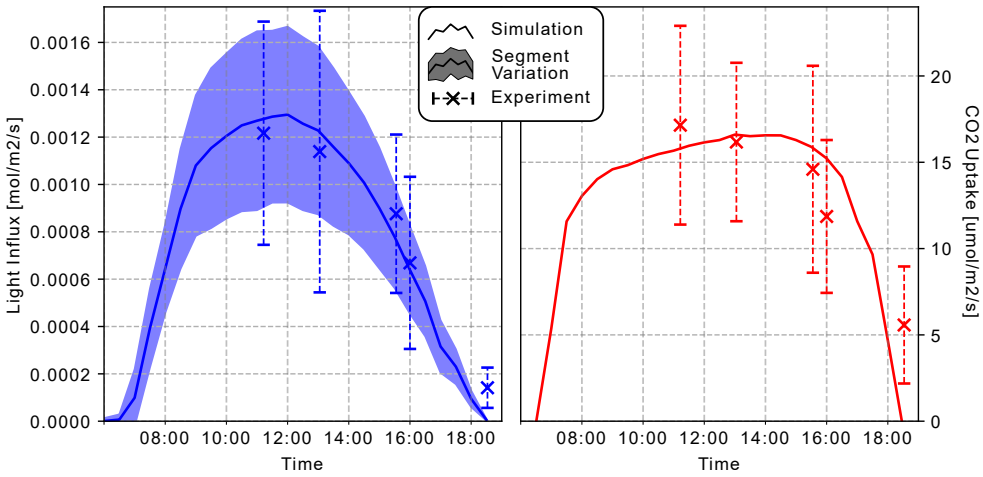

Fig. 6: Absorbed PAR in blue/left, and net co_2 uptake in red/right. The light flux standard deviation is segment based, i.e., depending on where on the plant the light is measured, the light measurement has different results, resulting in a high standard variation. The experimental data is marked with an x and a dotted line for measured variance.

Our UE implementation measures exactly around the submitted points, an array which can be mapped onto the geometry. As the rendering needs to switch modes for this task, the workload needs to be separated from the measurement or performed subsequently. The photosynthesis module of CPlantBox, if not restricted through other mechanisms, imposes a limitation on carbon uptake A_g, which is $A_g = \min(F_j, F_c)$ (Giraud et al. 2023), where F_j is the electron transport rate and F_c is the carboxylation rate, which is largely restricted by Nitrogen uptake. These rates are output by the simulation, particularly the photosynthesis module, when solving for the photosynthesis reactions. They are given by segment, which we interpolated to organ level for the visualization in Fig. 5. Different populations, one large and one rather small are shown in Fig. 5, showcasing how this kind of scaling would function in the Synavis pipeline. An output of this simulation is the fact that we can scale such an otherwise very mechanical value to a more clearer result, showing a potential limitation by F_c which could indicate the need for more nitrogen fertilization.

Our implementation for this work furthermore allows a full export of the FSPM geometry for multiple file formats, including VTK-compatible files, or Wavefront Object files that can be used with software such as DART (J. P. Gastellu-Etchegorry and Gascon 2004). The higher compatibility also allows users to employ more coupling techniques that can be used with other RT models, including lightweight models that run on CPU-only devices. The coupling is furthermore entirely steerable within Python, which also contains the simulation information provided by CPlantBox.

3.2 Light Influx Accuracy

We compare the experimentally measured values for light influx and carbon uptake to our virtual world model. Our comparison is shown in Fig. 6, which highlights both light influx as well as co_2 uptake per area. Our calibration yielded a close match to the experimental data, though we will note that multiple parameters could yield similar results. The calibration of the individual parameters used for the light evaluation can be referenced to in the resources that accompany this manuscript,

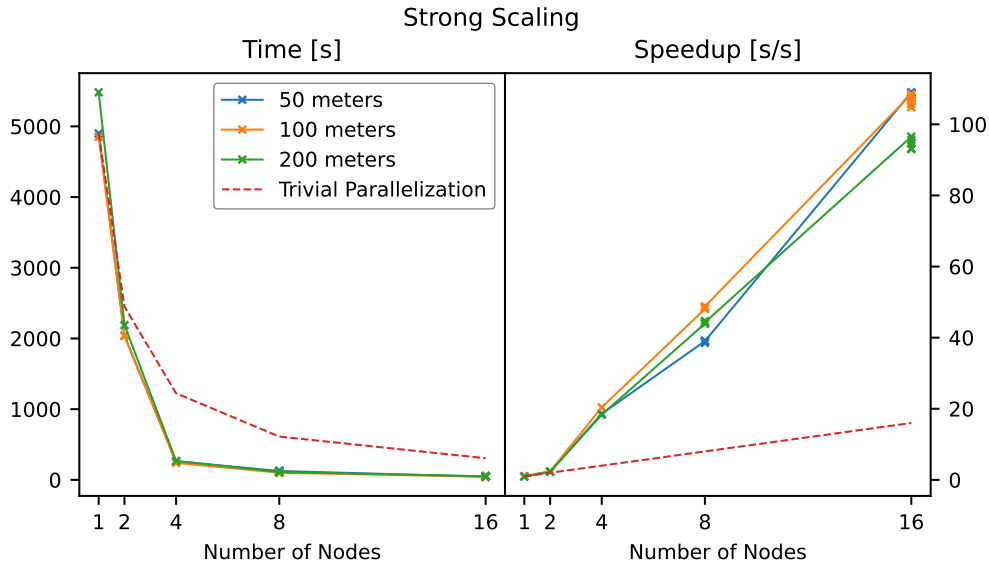

Fig. 7: Scaling of number of nodes while not scaling total problem size. A: Time of execution for the photosynthesis pipeline for a field of 500 plants, measuring the performance of the light evaluation only, from submission to reception of the exposure data. B: Derived speedup values for the parallel performance. We compare our results against the 'trivial parallelization' baseline, representing a speedup directly proportional to the number of nodes used. Our memory-bound applications has a superscaling behavior, resulting in an overall faster computation by means of parallelization and reduction of wait cycles for the computation.

see Sec. 5.4.

For our simulation study, we sub-sampled the 10-minute intervals to 30-minute intervals to reduce simulation time to allow for a better progression of the FSPM simulation. This is mostly a reduction of the computational workload to replicate the data, while still maintaining similar expressiveness. The base experimental time resolution of the experiment is non-uniform, with the measurements taking place in specific time slots with multiple measurements. We averaged measurements that were taken in the same interval, though on different plants labeled the same, resulting in the variance seen in Fig. 6. Similarly we highlight that while the measurements that were deemed *sunlit* in the experiments described in Nguyen et al. (2024). The light absorption measured in both the experiment and in our simulation study yielded high variance. This is likely caused by differences in measurement positions on the plants. The increase in the mornings was steeper than the decrease in the evening. This is also present in the raw data we used for the modeling (Schmidt 2024). Similarly, the co_2 uptake increased in such a fashion, though in the evening both radiation and uptake values are outside the experimental range. Our simulated co_2 uptake at around 11 ó clock was lower than the average experimental value, though within the variance. Our light influx variance was peculiarly low in the range of 7 to 8 ó clock, which we could attribute to the scattering values that were used in the virtual environment, in combination with the fact that overall illumination was low.

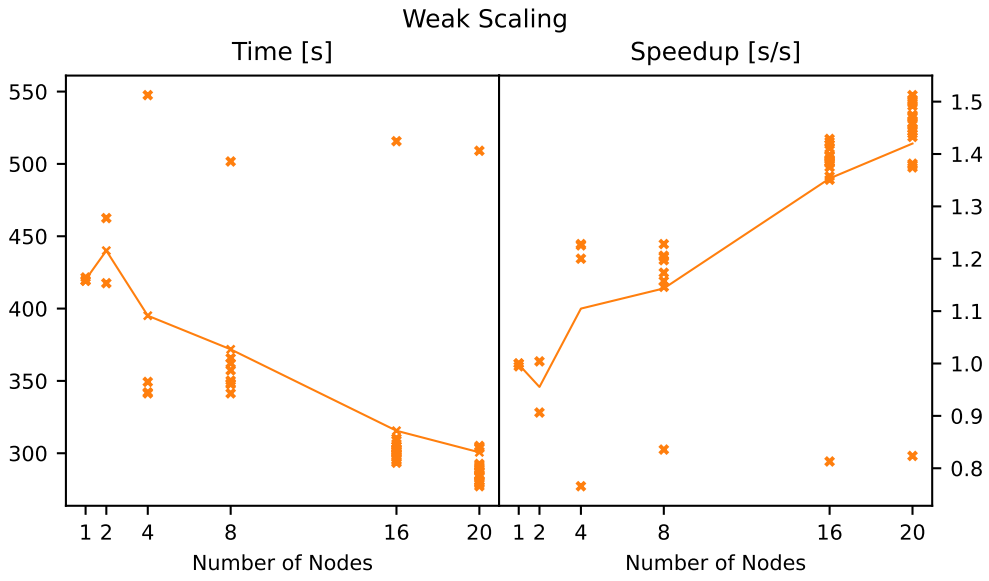

Fig. 8: Scaling of both problem size and number of nodes. A: Time of execution if we increase the data in the same way as the number of nodes. B: Derived speedup values for the parallel performance, again calculated as $T(1)/T(N)$.

3.3 Scaling and Performance

We have investigated the scalability of the coupling to determine the performance of the method on HPC systems. We show both timings and resulting speedup in Fig. 7. In it, we observe what appears to be super-scaling behavior, which requires closer investigation. First, the light evaluation being distributed across nodes will make the measurement faster than on a single node, with N/P measurement points as opposed to N. We note that there *is* overhead in the shape of the surrounding plants that get rendered as to not distort the boundaries between the compartments of the field, even though these plants act as geometric inclusion only. However, because the overall amount of triangles in the virtual scene is significantly lower, the engine also performs better *per render pass*, which results in an increased performance of a single evaluation. Thus, the parallel performance appears to be so steeply increasing with the number of nodes. This is commonly referred to as memory-bound performance (see e.g., Allande et al. (2014)).

Fig 8 shows the weak scaling of our system. The speedup appears to be slightly increasing, though this is constraint by the controlling node that handles the same area. From measurement, the added speedup even though the problem size remains constant might have different reasons, as investigated in Sec. 4.1. The measurement is taken through the framework itself, and as such it does not account for effects due to MPI setup and the initialization phase of the photosynthesis simulation. In our evaluation, we found that there was not much difference in the performance regarding the total (virtual) resolution of the light measurement, based on the top three performing settings in this context. An increased resolution, i.e., an increased number of light meters, will result in a decreased frame performance, resulting in a relatively similar performance overall. From the hard scaling comparison in Fig. 7, we found that 100 light meters yields best performance, resulting in a total resolution of $8^2 \cdot 100 = 6400$ measurement points.

4 Discussion

This work has been an investigation into the validity of using the Synavis framework for simulated embeddings, lifting the data production pipeline to a true embedding of simulated data. The contribution on a technical level is the coupling to the FSPM as well as the implementation of a functional embedding into UE such that it can be used together with other models beyond plant models.

4.1 Distributed Simulation of FSPM Embeddings

It is generally no issue to distribute plant simulations across nodes in cases where the simulation is entirely at organ scale. Scale conversions, such as between cells and plant level, always require a form of heuristics, as it would be cumbersome to simulate a plant on cell level entirely. As such, the data scale conversion is an important tool to not only use varying resolutions of experimental data, but also to distribute the plant simulation. We use this to estimate the size of the buffer region in our tests. Typically, with our parameters regarding the size and density of the field, two rows of plants around our measurement area is sufficient on average. During midday, and depending on geographic location, the buffer plants might even be omitted, which is also the case for cloudy days. In the first case, this is because the shadows are very close to the plant and might not significantly touch neighboring plants, and in the second case, most light arrives in so many scattered pathways that shadows have decreased contrast.

The scaling experiments in Fig. 7 and Fig. 8 show a clear speedup through HPC use. We note, however, that we do not believe the apparent trend in Fig. 8 regarding the speedup between 8 and 20 GPU nodes has a systematic cause. There are components regarding the order of operations, particularly the spawning of plants and the initialization of the measurement, that cause a greater variance in the sampling. Particularly we note that the main compute node did not speedup, but rather remained somewhat constant regarding its sampling and the collection of light influxes. Uncovering the hyperparameters that contribute to the eventual parallel efficiency is beyond the scope of this work, but would be the next step to complete the HPC integration of the Synavis framework, particularly in combination with AI workloads. To accommodate DNN pipelines and to reduce over-fitting, the number of visualizers should be lower than the number of learners, as the learning algorithms needs to use image-based augmentations in addition to domain-based augmentations to fully make use of the synthetic data, as image-based augmentations have uses beyond adding more pictures to the data space.

Distributed performance using the Unreal Engine also would have some limitations regarding inter-module connectivity, as scaling up the virtual world rendering on a dedicated visualization module might impact the latency. Users of the framework would need to be cognizant of the local properties of the HPC systems they operate on, as it could introduce topology or module-depended differences in communication times. Our communication tests typically yielded latencies between modules (through Ethernet, e.g. between JURECA-DC and other clusters on-site) of around 200ms. We do highlight the usefulness enabling graphics on user-exclusive GPU nodes as opposed to offloading the capability of rendering onto interactive nodes that cannot tolerate high workloads.

4.2 In-Silico Recreation of a Field Experiment

The gas chamber measurements can be referred to in Nguyen et al. (2024), and we will note that it is not an isolated chamber, and thus the concentrations are susceptible to wind. We are comparing distribution-free measurements, as the reported photosynthesis measurements are provided per area and time. Our pipeline is able to replicate the experiments of this specific configuration well. We have to note that there are many configurations of parameters that might yield similar results, some of which are outside what are realistic properties for plants. The overfitting of simulation models has been previously reported in other domains, such as radiotherapy in silico simulation (van der Schaaf

et al. 2012). The calibration of the light influx is not only dependent on the actual sunlight intensity, but also on the atmospheric scattering of specific wavelengths, as well as leaf surface scattering. To truly provide an embedding for more different FSPM simulations, a full study, including experimental measurements of leaf surfaces, would be required, along with a study on the mapping of wavelengths into the virtual world. We have fitted and highlighted that we can represent experimental data, but the full parametrization and analysis of the virtual world is beyond the scope of this paper, particularly regarding its use as data generation pipeline. The soil parametrization regarding light propagation is an important factor that contributes to total illumination.

The experimental replication furthermore yields a type of bottleneck that correlates with the production of synthetic data. In future work, less light influx settings, i.e. day, time, and weather, should be computed to speedup evaluation, in addition to only computing a representative subset of the field. This would speedup the total evaluation significantly, and more time can be allocated for the actual data generation, after deriving a heuristic of the mapping between the parameter space and the light influx or carbon uptake limitation on the plant. For the domain composition, the challenge poses itself that a camera either needs to be facing downwards, as this is how the light evaluation is parallelized, or the render scene needs to be setup by reducing the decomposed scene into one.

4.3 Functional Synthetic Data using UE

One factor contributing to the increased use of UE in certain domains, specifically in plant science, is the ability to very quickly prototype scenes. The key aspect is that visually, a user of these engines can achieve a good general setting very quickly, beyond the need for texturing and geometrization, which are issues that are largely independent of the concrete rendering back-end. Examples of this are Helios or DART with subsequent rendering, or any alternative to UE for synthetic data implementation. These models also require calibration of surface properties, even if respective rendering pipeline use multi-spectral or physics-based models.

There are certain aspects of the pipeline that required adjustment seeing as UE is historically a game engine, which goes beyond terminology. Certain parts of our workflow are implemented on top of basic optimizations we explicitly turn off, or circumvent using procedural geometry. Similarly, certain features might not be as fleshed out as others, particularly regarding the implementation of simulation models, requiring some fine-tuning in cases that extend beyond what a game engine typically delivers. In Sec. 3.2 we showcase the connection between the simulation and the eventual training output that could be used for inference in field settings. There are caveats, but generally, our method adds to the virtual scene in a graphics engine information that is otherwise not available, by employing simulation models to act as data source.

While this work focuses on the central validity and applicability of the workflow for photosynthesis problems, this pipeline is a generic data production pipeline. More functional information within the virtual world would enable a more informed embedding and ultimately strengthen the ability of deep neural networks to estimate plant health through remote sensing techniques. Camera footage together with distance sensors are comparatively low-cost and easily reproducible in experiments, and they are also always computed in UE, as pixel shaders depend on pixel depth and relative velocity information. Camera settings, particularly within the data production, but also for the real world, are extremely important. In Fig. 5 we showcase a functional mapping and visualize the plants in a way that makes them easily distinguishable for this manuscript, but the scene is technically overlit, as two pixels yielding different light absorption information might have the same scene color value. This is an issue that is a vital component of any functional analysis that depends on consumer camera images. Unless the exposure is configured to take only leaf surfaces into account, there will be some loss of information due to the restricted brightness range of the images. This is the case even in UE, which internally uses a high dynamic range system but is flattened into pictures to produce synthetic data. We have previously reported the dependency of threshold-based relative

leaf area on image capturing conditions in Baker et al. (2023).

5 Conclusion

This paper presents a coupling approach that is directly embedded into a data generation framework. Recent advances in data generation, particularly for agricultural science, have highlighted the need for embedded models in these frameworks. Here, we proposed a coupling with an FSPM model, highlighting the usefulness of its functional simulation for the enrichment of the data generation. A new implementation of radiative transfer representation in UE was developed to couple the FSPM CPlantBox to a light influx evaluation for photosynthesis evaluation. We show that we can embed functional properties into the virtual scene accurately, scalable, and usable for synthetic data.

We have included a replication of an experiment, a data production pipeline for light exposure efficiency and the carbon uptake limitation in plants, as well as the scalability of our method on HPC systems. Exploring how to best utilize the tools available, especially when implementing the simulations into a wider workflow, is essential.

5.1 Author Contributions

This work was written, revised, and edited by all authors. DB is primary author and conducted the implementation, data analysis, visualization, and drafted this work. DB,MG,JHG adapted the Synavis/UE software for this work. MR,JHG provided access to supercomputing resources. MG,HS,AS provided domain expertise and guided the implementation of the plant simulation and experimental replication. EH,AS guided the data analysis and research aims of this work. MR,EH are the primary supervisors for the University of Iceland. AS is primary supervisor of this work.

5.2 Acknowledgments

The authors thank Thuy Huu Nguyen for his advise on data interpretation and use. Furthermore, the authors acknowledge compute time granted to the JURECA-DC Supercomputer in the project visforai.

The authors would like to acknowledge funding provided by the German government to the Gauss Centre for Supercomputing via the InHPC-DE project (01H17001). 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 bio-economy (Rhizo4Bio), subproject CROP (ref. FKZ 031B0909A).

5.3 Conflicts of Interest

The author declare that there is no conflict of interest regarding the publication of this article.

5.4 Data Availability

The Selhausen data sets are described in Lärm et al. 2023 and Nguyen et al. 2024, and all relevant download information can be seen in those articles. Our framework, Synavis, is Open Source and the relevant codes can be seen in `dhelmrich/synavis`. CPlantBox is open source and available at `Plant-Root-Soil-Interactions-Modelling/CPlantBox`.

A video description is available at 10.6084/m9.figshare.28280780. The simulation code that is associated with this manuscript is available in the `feature/experiment` branch and is scheduled to be merged.

References

Agarwal, D., T. Kucukpinar, J. Fraser, J. Kerley, A. R. Buck, D. T. Anderson, and K. Palaniappan (2023). "Simulating City-Scale Aerial Data Collection Using Unreal Engine". In: *2023 IEEE Applied Imagery Pattern Recognition Workshop (AIPR)*, pp. 1–9 DOI: 10.1109/AIPR60534.2023.10440697.

Allande, C., J. Jorba, A. Sikora, and E. Cèsar (2014). "A Performance Model for OpenMP Memory Bound Applications in Multisocket Systems". In: *Procedia Computer Science* 29. 2014 International Conference on Computational Science, pp. 2208–2218. ISSN: 1877-0509 DOI: https://doi.org/10.1016/j.procs.2014.05.206.

Bailey, B. N. (2019). "Helios: A Scalable 3D Plant and Environmental Biophysical Modeling Framework". In: *Frontiers in Plant Science* 10. ISSN: 1664-462X DOI: 10.3389/fpls.2019.01185.

Baker, D. N., F. M. Bauer, M. Giraud, A. Schnepf, J. H. Göbbert, H. Scharr, E. Þ. Hvannberg, and M. Riedel (Dec. 2023). "A scalable pipeline to create synthetic datasets from functionalstructural plant models for deep learning". In: *in silico Plants* 6.1, diad022. ISSN: 2517-5025 DOI: 10.1093/insilicoplants/diad022.

Baker, D. N., F. M. Bauer, A. Schnepf, H. Scharr, M. Riedel, J. H. Göbbert, and E. Hvannberg (May 27, 2024). "Adapting Agricultural Virtual Environments in Game Engines to Improve HPC Accessibility". In: *Communications in Computer and Information Science*. nordic e-Infrastructure Collaboration Conference, Tallinn (Estonia), 27 May 2024 - 29 May 2024. Springer, pp. 1–15 DOI: 20.500.11815/4936.

Bauer, F. M., L. Lärm, S. Morandage, G. Lobet, J. Vanderborght, H. Vereecken, and A. Schnepf (2022). "Development and Validation of a Deep Learning Based Automated Minirhizotron Image Analysis Pipeline". In: *Plant Phenomics* 2022 DOI: 10.34133/2022/9758532.

Bogena, H. R. (Feb. 2016). "TERENO: German network of terrestrial environmental observatories". In: *J. Large-scale Res. Facil. JLSRF* 2.A52 DOI: 10.17815/jlsrf-2-98.

Farquhar, G. D., S. von Caemmerer, and J. A. Berry (June 1980). "A biochemical model of photosynthetic CO2 assimilation in leaves of C3 species". In: *Planta* 149.1, pp. 78–90. ISSN: 1432-2048 DOI: 10.1007/BF00386231.

Féret, J.-B. and F. de Boissieu (2024). "prospect: an R package to link leaf optical properties with their chemical and structural properties with the leaf model PROSPECT". In: *Journal of Open Source Software* 9.94, p. 6027 DOI: 10.21105/joss.06027.

Giraud, M., S. Le Gall, M. Harings, M. Javaux, D. Leitner, F. Meunier, Y. Rothfuss, D. van Dusschoten, J. Vanderborght, H. Vereecken, G. Lobet, and A. Schnepf (July 2023). "CPlantBox: a fully coupled modelling platform for the water and carbon fluxes in the soilplantatmosphere continuum". In: *in silico Plants* 5.2, diad009. ISSN: 2517-5025 DOI: 10.1093/insilicoplants/diad009.

J. P. Gastellu-Etchegorry, E. M. and F. Gascon (2004). "DART: a 3D model for simulating satellite images and studying surface radiation budget". In: *International Journal of Remote Sensing* 25.1, pp. 73–96 DOI: 10.1080/0143116031000115166.

John V. Martonchik, C. J. B. and A. H. Strahler (2000). "A review of reflectance nomenclature used in remote sensing". In: *Remote Sensing Reviews* 19.1-4, pp. 9–20 DOI: 10.1080/02757250009532407.

Krüger, M., D. Gilbert, T. W. Kuhlen, and T. Gerrits (July 2024). "Game Engines for Immersive Visualization: Using Unreal Engine Beyond Entertainment". In: *PRESENCE: Virtual and Augmented Reality* 33, pp. 31–55. ISSN: 1054-7460 DOI: 10.1162/pres_a_00416.

Lärm, L., F. M. Bauer, N. Hermes, J. van der Kruk, H. Vereecken, J. Vanderborght, T. H. Nguyen, G. Lopez, S. J. Seidel, F. Ewert, A. Schnepf, and A. Klotzsche (Oct. 2023). "Multi-year belowground data of minirhizotron facilities in Selhausen". In: *Nature Scientific Data* 10.1, p. 672. ISSN: 2052-4463 DOI: 10.1038/s41597-023-02570-9.

Lei, T., J. Graefe, I. K. Mayanja, M. Earles, and B. N. Bailey (2024). "Simulation of Automatically Annotated Visible and Multi-/Hyperspectral Images Using the Helios 3D Plant and Radiative Transfer Modeling Framework". In: *Plant Phenomics* 6, pp. 1–16 DOI: 10.34133/plantphenomics.0189.

Leuning, R. (1995). "A critical appraisal of a combined stomatal-photosynthesis model for C3 plants". In: *Plant, Cell and Environment* 18.4, pp. 339–355. ISSN: 0140-7791 DOI: 10.1111/j.1365-3040.1995.tb00370.x.

Li, X., J. Park, C. Reberg-Horton, S. Mirsky, E. Lobaton, and L. Xiang (2024). "Photorealistic Arm Robot Simulation for 3D Plant Reconstruction and Automatic Annotation using Unreal Engine 5". In: *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 5480–5488 DOI: 10.1109/CVPRW63382.2024.00557.

Miao, G., K. Guan, X. Yang, C. J. Bernacchi, J. A. Berry, E. H. DeLucia, J. Wu, C. E. Moore, K. Meacham, Y. Cai, B. Peng, H. Kimm, and M. D. Masters (2018). "Sun-Induced Chlorophyll Fluorescence, Photosynthesis, and Light Use Efficiency of a Soybean Field from Seasonally Continuous Measurements". In: *Journal of Geophysical Research: Biogeosciences* 123.2, pp. 610–623 DOI: 10.1002/2017JG004180.

Minervini, M., H. Scharr, and S. A. Tsaftaris (2015). "Image analysis: the new bottleneck in plant phenotyping". In: *IEEE signal processing magazine* 32.4, pp. 126–131 DOI: 10.1109/MSP.2015.2405111.

Mousavi, M., A. Khanal, and R. Estrada (2020). "AI Playground: Unreal Engine-Based Data Ablation Tool for Deep Learning". In: *Advances in Visual Computing*. Ed. by G. Bebis, Z. Yin, E. Kim, J. Bender, K. Subr, B. C. Kwon, J. Zhao, D. Kalkofen, and G. Baciu. Cham: Springer International Publishing, pp. 518–532 DOI: 10.1007/978-3-030-64559-5_41.

Nguyen, T. H., G. Lopez, S. J. Seidel, L. Lärm, F. M. Bauer, A. Klotzsche, A. Schnepf, T. Gaiser, H. Hüging, and F. Ewert (June 2024). "Multi-year aboveground data of minirhizotron facilities in Selhausen". In: *Nature Scientific Data* 11.1, p. 674. ISSN: 2052-4463 DOI: 10.1038/s41597-024-03535-2.

Nicodemus, F. E., J. C. Richmond, J. J. Hsia, I. W. Ginsberg, T. Limperis, S. Harman, and J. J. Baruch (1977). "Geometrical considerations and nomenclature for reflectance". In: DOI: 10.5555/136913.136929.

Qi, J., D. Xie, T. Yin, G. Yan, J.-P. Gastellu-Etchegorry, L. Li, W. Zhang, X. Mu, and L. K. Norford (2019). "LESS: LargE-Scale remote sensing data and image simulation framework over heterogeneous 3D scenes". In: *Remote Sensing of Environment* 221, pp. 695–706. ISSN: 0034-4257 DOI: https://doi.org/10.1016/j.rse.2018.11.036.

Reichenau, T. G., W. Korres, M. Schmidt, A. Graf, G. Welp, N. Meyer, A. Stadler, C. Brogi, and K. Schneider (2020). "A comprehensive dataset of vegetation states, fluxes of matter and energy, weather, agricultural management, and soil properties from intensively monitored crop sites in western Germany". In: *Earth System Science Data* 12.4, pp. 2333–2364 DOI: 10.5194/essd-12-2333-2020.

Salesin, K., K. D. Knobelspiesse, J. Chowdhary, P.-W. Zhai, and W. Jarosz (2024). "Unifying radiative transfer models in computer graphics and remote sensing, Part I: A survey". In: *Journal of Quantitative Spectroscopy and Radiative Transfer* 314, p. 108847. ISSN: 0022-4073 DOI: 10.1016/j.jqsrt.2023.108847.

Schmidt, M. (2024). *EC/Climate station Selhausen 3* DOI: 20.500.11952/TERENO/1056009555.

Sievänen, R., C. Godin, T. M. DeJong, and E. Nikinmaa (Sept. 2014). "Functionalstructural plant models: a growing paradigm for plant studies". In: *Annals of Botany* 114.4, pp. 599–603. ISSN: 0305-7364 DOI: 10.1093/aob/mcu175.

Taiz, L. (2013). "Agriculture, plant physiology, and human population growth: past, present, and future". In: *Theoretical and Experimental Plant Physiology* 25.3, pp. 167–181 DOI: 10.1590/S2197-00252013000300001.

Thörnig, P. (2021). "JURECA: Data Centric and Booster Modules implementing the Modular Supercomputing Architecture at Jülich Supercomputing Centre". In: *Journal of large-scale research facilities JLSRF* DOI: 10.17815/jlsrf-7-182.

Thornley, J. H. M. (Sept. 2011). "Plant growth and respiration re-visited: maintenance respiration defined it is an emergent property of, not a separate process within, the system and why the respiration:photosynthesis ratio is conservative". In: *Annals of Botany* 108.7, pp. 1365–1380. ISSN: 0305-7364 DOI: 10.1093/aob/mcr238.

Tsaftaris, S. A., M. Minervini, and H. Scharr (2016). "Machine learning for plant phenotyping needs image processing". In: *Trends in plant science* 21.12, pp. 989–991 DOI: 10.1016/j.tplants.2016.10.002.

Tuzet, A., A. Perrier, and R. Leuning (2003). "A coupled model of stomatal conductance, photosynthesis and transpiration". In: *Plant, Cell & Environment* 26.7, pp. 1097–1116 DOI: 10.1046/j.1365-3040.2003.01035.x.

van der Schaaf, A., C.-J. Xu, P. van Luijk, A. A. vant Veld, J. A. Langendijk, and C. Schilstra (2012). "Multivariate modeling of complications with data driven variable selection: Guarding against overfitting and effects of data set size". In: *Radiotherapy and Oncology* 105.1, pp. 115–121. ISSN: 0167-8140 DOI: 10.1016/j.radonc.2011.12.006.

Vanderborght, J., J. Baca Cabrera, G. lobet, D. Leitner, M. Javaux, V. Couvreur, and A. Schnepf (Mar. 2024). "Upscaling of 3D root hydraulic architectures of trees to 1D root hydraulic models". In: *EGU24*. Copernicus GmbH DOI: 10.5194/egusphere-egu24-7397.

Von Caemerer, S. (2013). "Steady-state models of photosynthesis". In: *Plant, Cell & Environment* 36.9, pp. 1617–1630 DOI: 10.1111/pce.12098.

Walia, A., R. Carter, R. Wightman, E. M. Meyerowitz, H. Jönsson, and A. M. Jones (Dec. 2024). "Differential growth is an emergent property of mechanochemical feedback mechanisms in curved plant organs". In: *Developmental Cell* 59.24, 3245–3258.e3. ISSN: 1534-5807 DOI: 10.1016/j.devcel.2024.09.021.

Ward, D., P. Moghadam, and N. Hudson (2018). "Deep leaf segmentation using synthetic data". In: *arXiv preprint arXiv:1807.10931* DOI: 10.48550/arXiv.1807.10931.

Yin, X., M. van Oijen, and A. H. C. M. Schapendonk (2004). "Extension of a biochemical model for the generalized stoichiometry of electron transport limited C3 photosynthesis". In: *Plant, Cell & Environment* 27.10, pp. 1211–1222 DOI: 10.1111/j.1365-3040.2004.01224.x.

Yu, P., C. Li, M. Li, X. He, D. Wang, H. Li, C. Marcon, Y. Li, S. Perez-Limón, X. Chen, M. Delgado-Baquerizo, R. Koller, R. Metzner, D. van Dusschoten, D. Pflugfelder, L. Borisjuk, I. Plutenko, A. Mahon, M. F. R. Resende, S. Salvi, A. Akale, M. Abdalla, M. A. Ahmed, F. M. Bauer, A. Schnepf, G. Lobet, A. Heymans, K. Suresh, L. Schreiber, C. M. McLaughlin, C. Li, M. Mayer, C.-C. Schön, V. Bernau, N. von Wirén, R. J. H. Sawers, T. Wang, and F. Hochholdinger (June 2024). "Seedling root system adaptation to water availability during maize domestication and global expansion". In: *Nature Genetics* 56.6, pp. 1245–1256. ISSN: 1546-1718 DOI: 10.1038/s41588-024-01761-3.

Zhao, X., J. Qi, J. Jiang, S. Liu, H. Xu, S. Lin, Z. Yu, L. Li, and H. Huang (2024). "Fine-scale retrieval of leaf chlorophyll content using a semi-empirically accelerated 3D radiative transfer model". In: *International Journal of Applied Earth Observation and Geoinformation* 135, p. 104285. ISSN: 1569-8432 DOI: 10.1016/j.jag.2024.104285.

A Data Availability

This thesis is based on the Synavis framework, which is available in the public GitHub repository, https://github.com/dhelmrich/Synavis.

The repository includes the Synavis framework `synavis/` and the code examples needed to reproduce the publications in `python/`. `SynavisUE` contains the plugin that can be installed on a UE project. Lastly, example projects can be found in https://github.com/dhelmrich/SynavisUEexample, which has branches that accommodate certain use cases, including a laboratory and a field use case.

VRoot is a public application available at https://github.com/dhelmrich/VRoot. The application is a UE application to be built with a Head-Mounted Display system. The folder `Server` contains the backend data analysis server that is being used with the application.

The data repository containing anonymized study data has been uploaded to an online repository to comply with FAIR data principles. The data is anonymized and includes all participant answers and root tracings. The study has received ethical clearance from the ethics committee of Trier University.

D. **Baker** and D. Zielasko. *Study Data to VRoot: An XR-Based Application for Manual Root System Architecture Reconstruction*. Version V1. 2024 DOI: `10.26165/JUELICH-DATA/B9SBOS`.

Video material has been published for all publications. The videos can be found in references in the individual paper summaries and here:

D. N. Baker. *Supplemental Video for our Manuscript ISPLANTS-2023-026.R1*. FigShare. 2023 DOI: `10.6084/m9.figshare.24179136`.

D. N. Baker. *Video on VRoot: An XR-Based Application for Manual Root System Architecture Reconstruction*. FigShare. 2024 DOI: `10.6084/m9.figshare.26003494`.

D. N. Baker. *Scene Distribution using Synavis for Game Engine HPC Use-Cases*. FigShare. 2024 DOI: `10.6084/m9.figshare.25723773`.

D. N. Baker. *Video to: Virtual World Coupling with Photosynthesis Evaluation for Synthetic Data Production*. FigShare. 2025 DOI: `10.6084/m9.figshare.28280780`.

References

[Aac+22] M. Aach, R. Sedona, A. Lintermann, G. Cavallaro, H. Neukirchen, and M. Riedel. "Accelerating Hyperparameter Tuning of a Deep Learning Model for Remote Sensing Image Classification." In: *IGARSS 2022 - 2022 IEEE International Geoscience and Remote Sensing Symposium*. 2022, pp. 263–266 DOI: 10.1109/IGARSS46834.2022.9883257.

[ARS10] J. A. Acebrón, Á. Rodríguez-Rozas, and R. Spigler. "Efficient Parallel Solution of Nonlinear Parabolic Partial Differential Equations by a Probabilistic Domain Decomposition." In: *Journal of Scientific Computing* 43.2 (May 2010), pp. 135–157. ISSN: 1573-7691 DOI: 10.1007/s10915-010-9349-2.

[ACT21] H. Achicanoy, D. Chaves, and M. Trujillo. "StyleGANs and Transfer Learning for Generating Synthetic Images in Industrial Applications." In: *Symmetry* 13.8 (2021). ISSN: 2073-8994 DOI: 10.3390/sym13081497.

[Agh+19] H. H. Aghdam, A. Gonzalez-Garcia, J. v. d. Weijer, and A. M. Lopez. "Active Learning for Deep Detection Neural Networks." In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2019 DOI: 10.48550/arXiv.1911.09168.

[And15] A. Andrade. "Game engines: a survey." In: *EAI Endorsed Transactions on Serious Games* 2.6 (Nov. 2015) DOI: 10.4108/eai.5-11-2015.150615.

[Atk+19] J. A. Atkinson, M. P. Pound, M. J. Bennett, and D. M. Wells. "Uncovering the hidden half of plants using new advances in root phenotyping." In: *Current Opinion in Biotechnology* 55 (2019). Analytical Biotechnology, pp. 1–8. ISSN: 0958-1669 DOI: 10.1016/j.copbio.2018.06.002.

[Aum15] M. Aumüller. "The Architecture of Vistle, a Scalable Distributed Visualization System." In: *Solving Software Challenges for Exascale*. Ed. by S. Markidis and E. Laure. Cham: Springer International Publishing, 2015, pp. 141–147 DOI: 10.1007/978-3-319-15976-8_11.

[Bai19] B. N. Bailey. "Helios: A Scalable 3D Plant and Environmental Biophysical Modeling Framework." In: *Frontiers in Plant Science* 10 (2019). ISSN: 1664-462X DOI: 10.3389/fpls.2019.01185.

[Bak+25a] D. N. **Baker**, F. Bauer, A. Schnepf, H. Scharr, M. Riedel, J. H. Göbbert, and E. Hvannberg. “Adapting Agricultural Virtual Environments in Game Engines to Improve HPC Accessibility.” In: *Nordic e-Infrastructure Tomorrow*. Ed. by A. Azab and T. Malkiewicz. Cham: Springer Nature Switzerland, 2025, pp. 152–167. ISBN: 978-3-031-86240-3 DOI: 10.1007/978-3-031-86240-3_11.

[Bak+25b] D. N. **Baker**, T. Selzner, J. H. Göbbert, H. Scharr, M. Riedel, E. þ. Hvannberg, A. Schnepf, and D. Zielasko. “VRoot: A VR-Based Application for Manual Root System Architecture Reconstruction.” In: *Plant Phenomics* (2025), p. 100013. ISSN: 2643-6515 DOI: 10.1016/j.plaphe.2025.100013.

[BZ24] D. **Baker** and D. Zielasko. *Study Data to VRoot: An XR-Based Application for Manual Root System Architecture Reconstruction*. Version V1. 2024 DOI: 10.26165/JUELICH-DATA/B9SBOS.

[Bak23] D. N. Baker. *Supplemental Video for our Manuscript ISPLANTS-2023-026.R1*. FigShare. 2023 DOI: 10.6084/m9.figshare.24179136.

[Bak24a] D. N. Baker. *Scene Distribution using Synavis for Game Engine HPC Use-Cases*. FigShare. 2024 DOI: 10.6084/m9.figshare.25723773.

[Bak24b] D. N. Baker. *Video on VRoot: An XR-Based Application for Manual Root System Architecture Reconstruction*. FigShare. 2024 DOI: 10.6084/m9.figshare.26003494.

[Bak25] D. N. Baker. *Video to: Virtual World Coupling with Photosynthesis Evaluation for Synthetic Data Production*. FigShare. 2025 DOI: 10.6084/m9.figshare.28280780.

[Bak+23] D. N. **Baker**, F. M. Bauer, M. Giraud, A. Schnepf, J. H. Göbbert, H. Scharr, E. Þ. Hvannberg, and M. Riedel. “A scalable pipeline to create synthetic datasets from functional–structural plant models for deep learning.” In: *in silico Plants* 6.1 (Dec. 2023), diad022. ISSN: 2517-5025 DOI: 10.1093/insilicoplants/diad022.

[Bak+25c] D. N. **Baker**, M. Giraud, J. H. Goebbert, H. Scharr, M. Riedel, E. T. Hvannberg, and A. Schnepf. “Virtual World Coupling with Photosynthesis Evaluation for Synthetic Data Production.” 2025 DOI: 10.1101/2025.02.06.633870.

[Bak+24] D. N. **Baker**, T. Selzner, J. H. Göbbert, H. Scharr, M. Riedel, E. Þ. Hvannberg, A. Schnepf, and D. Zielasko. “Hands-On Plant Root System Reconstruction in Virtual Reality.” In: *Proceedings of the 30th ACM Symposium on Virtual Reality Software and Technology*. VRST ’24. Trier, Germany: Association for Computing Machinery, 2024. ISBN: 9798400705359 DOI: 10.1145/3641825.3689494.

[Bau+16] A. C. Bauer, H. Abbasi, J. Ahrens, H. Childs, B. Geveci, S. Klasky, K. Moreland, P. O'Leary, V. Vishwanath, B. Whitlock, and E. W. Bethel. "In Situ Methods, Infrastructures, and Applications on High Performance Computing Platforms." In: *Computer Graphics Forum* 35.3 (2016), pp. 577–597 DOI: https://doi.org/10.1111/cgf.12930.

[Bau+24] F. M. Bauer, D. N. **Baker**, M. Giraud, J. C. Baca Cabrera, J. Vanderborght, G. Lobet, and A. Schnepf. "Root system architecture reorganization under decreasing soil phosphorus lowers root system conductance of Zea mays." In: *Annals of Botany* (Nov. 2024), mcae198. ISSN: 0305-7364 DOI: 10.1093/aob/mcae198.

[Bau+22] F. M. Bauer, L. Lärm, S. Morandage, G. Lobet, J. Vanderborght, H. Vereecken, and A. Schnepf. "Development and Validation of a Deep Learning Based Automated Minirhizotron Image Analysis Pipeline." In: *Plant Phenomics* 2022 (2022) DOI: 10.34133/2022/9758532.

[Bau+23] F. M. Bauer, G. Lobet, D. N. Helmrich, A. Galinski, Z. Kahlilova, L. Zaner, M. Kuczkowska, P. Yu, P. Dörmann, G. Schaaf, and A. Schnepf. "In silico investigation on phosphorus efficiency of Zea mays: An experimental whole plant model parametrization approach." English. In: Berlin, Germany, 29 March 2023 DOI: 10.34734/FZJ-2023-04031.

[Bec+24] T. Beck, A. Baroni, R. Bennink, G. Buchs, E. A. C. Pérez, M. Eisenbach, R. F. da Silva, M. G. Meena, K. Gottiparthi, P. Groszkowski, T. S. Humble, R. Landfield, K. Maheshwari, S. Oral, M. A. Sandoval, A. Shehata, I.-S. Suh, and C. Zimmer. "Integrating quantum computing resources into scientific HPC ecosystems." In: *Future Generation Computer Systems* 161 (2024), pp. 11–25. ISSN: 0167-739X DOI: 10.1016/j.future.2024.06.058.

[Ber+24] E. M. Berrigan, L. Wang, H. Carrillo, K. Echegoyen, M. Kappes, J. Torres, A. Ai-Perreira, E. McCoy, E. Shane, C. D. Copeland, L. Ragel, C. Georgousakis, S. Lee, D. Reynolds, A. Talgo, J. Gonzalez, L. Zhang, A. B. Rajurkar, M. Ruiz, E. Daniels, L. Maree, S. Pariyar, W. Busch, and T. D. Pereira. "Fast and Efficient Root Phenotyping via Pose Estimation." In: *Plant Phenomics* 6 (2024), p. 0175 DOI: 10.34133/plantphenomics.0175.

[Bie+17] T. Biedert, K. Werner, B. Hentschel, and C. Garth. "A task-based parallel rendering component for large-scale visualization applications." In: *Proceedings of the 17th Eurographics Symposium on Parallel Graphics and Visualization*. PGV '17. Barcelona, Spain: Eurographics Association, 2017, pp. 63–71 DOI: 10.2312/pgv.20171094.

[Bon+18] E. Bondi, D. Dey, A. Kapoor, J. Piavis, S. Shah, F. Fang, B. Dilkina, R. Hannaford, A. Iyer, L. Joppa, and M. Tambe. "AirSim-W: A Simulation Environment for Wildlife Conservation with UAVs." In: COMPASS '18. Menlo Park and San Jose, CA, USA: Association for Computing Machinery, 2018. ISBN: 9781450358163 DOI: 10.1145/3209811.3209880.

[Bou+17] P. Bourhis, J. L. Reutter, F. Suárez, and D. Vrgoč. “JSON: Data model, Query languages and Schema specification.” In: *Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*. PODS ’17. Chicago, Illinois, USA: Association for Computing Machinery, 2017, pp. 123–135. ISBN: 9781450341981 DOI: 10.1145/3034786.3056120.

[Bow+17] D. Bowman, J. J. LaViola Jr, E. Kruijff, R. P. McMahan, and I. Poupyrev. *3D user interfaces: theory and practice*. Addison-Wesley Professional, 2017 DOI: 10.5555/993837.

[Cao+20] X. Cao, J. Yao, Z. Xu, and D. Meng. “Hyperspectral Image Classification With Convolutional Neural Network and Active Learning.” In: *IEEE Transactions on Geoscience and Remote Sensing* 58.7 (2020), pp. 4604–4616 DOI: 10.1109/TGRS.2020.2964627.

[Cas+23] S. Casao, A. Otero, Á. Serra-Gómez, A. C. Murillo, J. Alonso-Mora, and E. Montijano. “A Framework for Fast Prototyping of Photo-realistic Environments with Multiple Pedestrians.” In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. 2023, pp. 9083–9089 DOI: 10.1109/ICRA48891.2023.10160586.

[CCP18] E. Castro, J. S. Cardoso, and J. C. Pereira. “Elastic deformations for data augmentation in breast cancer mass detection.” In: *2018 IEEE EMBS International Conference on Biomedical Health Informatics (BHI)*. 2018, pp. 230–234 DOI: 10.1109/BHI.2018.8333411.

[Cha+22] G. Chance, A. Ghobrial, K. McAreavey, S. Lemaignan, T. Pipe, and K. Eder. “On Determinism of Game Engines Used for Simulation-Based Autonomous Vehicle Verification.” In: *IEEE Transactions on Intelligent Transportation Systems* 23.11 (2022), pp. 20538–20552 DOI: 10.1109/TITS.2022.3177887.

[Che+20] J. Chen, J. Chen, D. Zhang, Y. Sun, and Y. Nanehkaran. “Using deep transfer learning for image-based plant disease identification.” In: *Computers and Electronics in Agriculture* 173 (2020), p. 105393. ISSN: 0168-1699 DOI: https://doi.org/10.1016/j.compag.2020.105393.

[Chi+20] H. Childs, S. D. Ahern, J. Ahrens, A. C. Bauer, J. Bennett, E. W. Bethel, P.-T. Bremer, E. Brugger, J. Cottam, M. Dorier, S. Dutta, J. M. Favre, T. Fogal, S. Frey, C. Garth, B. Geveci, W. F. Godoy, C. D. Hansen, C. Harrison, B. Hentschel, J. Insley, C. R. Johnson, S. Klasky, A. Knoll, J. Kress, M. Larsen, J. Lofstead, K.-L. Ma, P. Malakar, J. Meredith, K. Moreland, P. Navrátil, P. O’Leary, M. Parashar, V. Pascucci, J. Patchett, T. Peterka, S. Petruzza, N. Podhorszki, D. Pugmire, M. Rasquin, S. Rizzi, D. H. Rogers, S. Sane, F. Sauer, R. Sisneros, H.-W. Shen, W. Usher, R. Vickery, V. Vishwanath, I. Wald, R. Wang, G. H. Weber, B. Whitlock, M. Wolf, H. Yu, and S. B. Ziegeler. “A terminology for in situ visualization and analysis systems.” In: *The International Journal of High Performance Computing Applications* 34.6 (2020), pp. 676–691 DOI: 10.1177/1094342020935991.

[Cie+21] M. Cieslak, N. Khan, P. Ferraro, R. Soolanayakanahally, S. J. Robinson, I. Parkin, I. McQuillan, and P. Prusinkiewicz. “L-system models for image-based phenomics: case studies of maize and canola.” In: *in silico Plants* 4.1 (Dec. 2021). diab039. ISSN: 2517-5025 DOI: 10.1093/insilicoplants/diab039.

[CK09] K. Ciosek and P. Kotowski. “Generating 3D Plants using Lindenmayer System.” In: *GRAPP*. 2009, pp. 76–81 DOI: 10.5220/0001785300760081.

[De +20] P. De Bauw, T. H. Mai, A. Schnepf, R. Merckx, E. Smolders, and J. Vanderborght. “A functional–structural model of upland rice root systems reveals the importance of laterals and growing root tips for phosphate uptake from wet and dry soils.” In: *Annals of Botany* 126.4 (June 2020), pp. 789–806. ISSN: 0305-7364 DOI: 10.1093/aob/mcaa120.

[Dem+22] A. C. Demiralp, D. N. **Helmrich**, J. Protze, T. W. Kuhlen, and T. Gerrits. “Performance Assessment of Diffusive Load Balancing for Distributed Particle Advection.” In: *CSRN*. Západočeská univerzita, 2022 DOI: 10.24132/csrn.3201.2.

[Den+09] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. “ImageNet: A large-scale hierarchical image database.” In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009 DOI: 10.1109/CVPR.2009.5206848.

[Den+17] Y. Deng, Y. Ni, Z. Li, S. Mu, and W. Zhang. “Toward Real-Time Ray Tracing: A Survey on Hardware Acceleration and Microarchitecture Techniques.” In: *ACM Comput. Surv.* 50.4 (Aug. 2017). ISSN: 0360-0300 DOI: 10.1145/3104067.

[Den90] A. R. Dennis. “An overview of rendering techniques.” In: *Computers & Graphics* 14.1 (1990), pp. 101–115. ISSN: 0097-8493 DOI: 10.1016/0097-8493(90)90014-O.

[Dum23] M. Dumiak. “Exascale Comes to Europe: Germany will Host JUPITER, Europe’s Entry Into the Realm of Exascale Supercomputing.” In: *IEEE Spectrum* 60.1 (2023), pp. 50–51 DOI: 10.1109/MSPEC.2023.10006668.

[EZL89] D. Eager, J. Zahorjan, and E. Lazowska. “Speedup versus efficiency in parallel systems.” In: *IEEE Transactions on Computers* 38.3 (1989), pp. 408–423 DOI: 10.1109/12.21127.

[FAO24] FAO, IFAD, UNICEF, WFP and WHO. *The State of Food Security and Nutrition in the World 2024 – Financing to end hunger, food insecurity and malnutrition in all its forms*. Rome: FAO, 2024 DOI: 10.4060/cd1254en.

[FCB80] G. D. Farquhar, S. von Caemmerer, and J. A. Berry. “A biochemical model of photosynthetic CO2 assimilation in leaves of C3 species.” In: *Planta* 149.1 (1980), pp. 78–90. ISSN: 1432-2048 DOI: 10.1007/BF00386231.

[Fol95] J. D. Foley. *Computer graphics: principles and practice*. Addison-Wesley, 1995. ISBN: 978-0-201-84840-3 DOI: 10.5555/83821.

[FP21] A. Fonnet and Y. Prié. "Survey of Immersive Analytics." In: *IEEE Transactions on Visualization and Computer Graphics* 27.3 (Mar. 2021), pp. 2101–2122. ISSN: 1941-0506 DOI: 10.1109/TVCG.2019.2929033.

[FRS19] S. Friston, T. Ritschel, and A. Steed. "Perceptual rasterization for head-mounted display image synthesis." In: *ACM Trans. Graph.* 38.4 (July 2019). ISSN: 0730-0301 DOI: 10.1145/3306346.3323033.

[Gao+20] J. Gao, A. P. French, M. P. Pound, Y. He, T. P. Pridmore, and J. G. Pieters. "Deep convolutional neural networks for image-based Convolvulus sepium detection in sugar beet fields." In: *Plant Methods* 16.1 (Mar. 2020), p. 29. ISSN: 1746-4811 DOI: 10.1186/s13007-020-00570-z.

[Gir+23] M. Giraud, S. L. Gall, M. Harings, M. Javaux, D. Leitner, F. Meunier, Y. Rothfuss, D. van Dusschoten, J. Vanderborght, H. Vereecken, G. Lobet, and A. Schnepf. "CPlantBox: a fully coupled modelling platform for the water and carbon fluxes in the soil–plant–atmosphere continuum." In: *in silico Plants* 5.2 (July 2023), diad009. ISSN: 2517-5025 DOI: 10.1093/insilicoplants/diad009.

[Har+21] Z. K. J. Hartley, A. S. Jackson, M. Pound, and A. P. French. "GANana: Unsupervised Domain Adaptation for Volumetric Regression of Fruit." In: *Plant Phenomics* 2021 (2021) DOI: 10.34133/2021/9874597.

[Hel+21] D. N. **Helmrich**, J. H. Göbbert, M. Giraud, H. Scharr, A. Schnepf, and M. Riedel. "Towards Large-Scale Rendering of Simulated Crops for Synthetic Ground Truth Generation on Modular Supercomputers." In: *IEEE Symposium on Large Data Analysis and Visualization (LDAV)*. 2021 DOI: 10.48550/arXiv.2110.14946.

[Her+24] A. Herten, S. Achilles, D. Alvarez, J. Badwaik, E. Behle, M. Bode, T. Breuer, D. Caviedes-Voullième, M. Cherti, A. Dabah, S. E. Sayed, W. Frings, A. Gonzalez-Nicolas, E. B. Gregory, K. H. Mood, T. Hater, J. Jitsev, C. M. John, J. H. Meinke, C. I. Meyer, P. Mezentsev, J.-O. Mirus, S. Nassyr, C. Penke, M. Römmer, U. Sinha, B. von St. Vieth, O. Stein, E. Suarez, D. Willsch, and I. Zhukov. *Application-Driven Exascale: The JUPITER Benchmark Suite*. 2024 DOI: 10.48550/arXiv.2408.17211.

[Hor+21] J. Horn, Y. Zhao, N. Wandel, M. Landl, A. Schnepf, and S. Behnke. "Robust Skeletonization for Plant Root Structure Reconstruction from MRI." In: *2020 25th International Conference on Pattern Recognition (ICPR)*. 2021, pp. 10689–10696 DOI: 10.1109/ICPR48806.2021.9413045.

[Hou+25] Z. Hou, X. Lv, R. Lu, J. Zhang, Y. Li, Z. Yao, J. Li, J. Tang, and Y. Dong. *Advancing Language Model Reasoning through Reinforcement Learning and Inference Scaling*. 2025 DOI: 10.48550/arXiv.2501.11651.

[Huo+21] Y. Huo, A. Yang, Q. Jia, Y. Chen, B. He, and J. Li. “Efficient Visualization of Large-Scale Oblique Photogrammetry Models in Unreal Engine.” In: *ISPRS International Journal of Geo-Information* 10.10 (2021). ISSN: 2220-9964 DOI: 10.3390/ijgi10100643.

[Ise+17] P. Isenberg, T. Isenberg, M. Sedlmair, J. Chen, and T. Möller. “Visualization as Seen through its Research Paper Keywords.” In: *IEEE Transactions on Visualization and Computer Graphics* 23.1 (2017), pp. 771–780 DOI: 10.1109/TVCG.2016.2598827.

[JG04] E. M. J. P. Gastellu-Etchegorry and F. Gascon. “DART: a 3D model for simulating satellite images and studying surface radiation budget.” In: *International Journal of Remote Sensing* 25.1 (2004), pp. 73–96 DOI: 10.1080/0143116031000115166.

[JBB21] C. Jennings, H. Boström, and J.-I. Bruaroey. “WebRTC 1.0: Real-Time Communication Between Browsers.” In: *Recommendation of the World Wide Web Consortium* (Jan. 2021).

[Jim+15] J. Jimenez, K. Zsolnai, A. Jarabo, C. Freude, T. Auzinger, X.-C. Wu, J. der Pahlen, M. Wimmer, and D. Gutierrez. “Separable Subsurface Scattering.” In: *Comput. Graph. Forum* 34.6 (Sept. 2015), pp. 188–197. ISSN: 0167-7055 DOI: 10.1111/cgf.12529.

[Jon+20] D. Jones, C. Snider, A. Nassehi, J. Yon, and B. Hicks. “Characterising the Digital Twin: A systematic literature review.” In: *CIRP Journal of Manufacturing Science and Technology* 29 (2020), pp. 36–52. ISSN: 1755-5817 DOI: https://doi.org/10.1016/j.cirpj.2020.02.002.

[Jun+21] J. Jung, M. Maeda, A. Chang, M. Bhandari, A. Ashapure, and J. Landivar-Bowles. “The potential of remote sensing and artificial intelligence as tools to improve the resilience of agriculture production systems.” In: *Current Opinion in Biotechnology* 70 (2021). Food Biotechnology - Plant Biotechnology, pp. 15–22. ISSN: 0958-1669 DOI: 10.1016/j.copbio.2020.09.003.

[Jyo+16] S. A. Jyothi, A. Singla, P. B. Godfrey, and A. Kolla. “Measuring and Understanding Throughput of Network Topologies.” In: *SC ’16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. 2016, pp. 761–772 DOI: 10.1109/SC.2016.64.

[KP18] A. Kamilaris and F. X. Prenafeta-Boldú. “A review of the use of convolutional neural networks in agriculture.” In: *The Journal of Agricultural Science* 156.3 (2018), pp. 312–322 DOI: 10.1017/S0021859618000436.

[Kes+21] S. Kesselheim, A. Herten, K. Krajsek, J. Ebert, J. Jitsev, M. Cherti, M. Langguth, B. Gong, S. Stadtler, A. Mozaffari, G. Cavallaro, R. Sedona, A. Schug, A. Strube, R. Kamath, M. G. Schultz, M. Riedel, and T. Lippert. “JUWELS Booster – A Supercomputer for Large-Scale AI Research.” In: *High Performance Computing*. Ed. by H. Jagode, H. Anzt, H. Ltaief,

and P. Luszczek. Cham: Springer International Publishing, 2021. ISBN: 978-3-030-90539-2 DOI: 10.1007/978-3-030-90539-2_31.

[KB17] D. P. Kingma and J. Ba. *Adam: A Method for Stochastic Optimization*. 2017 DOI: 10.48550/arXiv.1412.6980.

[KS23] E. H. Korkut and E. Surer. "Visualization in virtual reality: a systematic review." In: *Virtual Reality* 27.2 (Jan. 2023), pp. 1447–1480. ISSN: 1434-9957 DOI: 10.1007/s10055-023-00753-8.

[KM22] T. W. Kuhlen and G. Marthys. "The aixCAVE at RWTH Aachen University." In: *Virtual and Augmented Reality - Foundations and Methods of Extended Realities*. Springer, 2022. Chap. 2 DOI: 10.1007/978-3-030-79062-2.

[Lar+15] M. Larsen, J. S. Meredith, P. A. Navrátil, and H. Childs. "Ray tracing within a data parallel framework." In: *2015 IEEE Pacific Visualization Symposium (PacificVis)*. 2015, pp. 279–286 DOI: 10.1109/PACIFICVIS.2015.7156388.

[LaV00] J. J. LaViola. "A discussion of cybersickness in virtual environments." In: *SIGCHI Bull.* 32.1 (Jan. 2000), pp. 47–56. ISSN: 0736-6906 DOI: 10.1145/333329.333344.

[Lec+19] C. Lecarpentier, R. Barillot, E. Blanc, M. Abichou, I. Goldringer, P. Barbillon, J. Enjalbert, and B. Andrieu. "WALTer: a three-dimensional wheat model to study competition for light through the prediction of tillering dynamics." In: *Annals of botany* 123.6 (2019), pp. 961–975 DOI: 10.1093/aob/mcy226.

[LBH15] Y. LeCun, Y. Bengio, and G. Hinton. "Deep learning." In: *Nature* 521.7553 (May 2015), pp. 436–444. ISSN: 1476-4687 DOI: 10.1038/nature14539.

[LeC+89] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel. "Handwritten Digit Recognition with a Back-Propagation Network." In: *Advances in Neural Information Processing Systems*. Ed. by D. Touretzky. Vol. 2. Morgan-Kaufmann, 1989 DOI: 10.5555/109230.109279.

[Lei+24] T. Lei, J. Graefe, I. K. Mayanja, M. Earles, and B. N. Bailey. "Simulation of Automatically Annotated Visible and Multi-/Hyperspectral Images Using the Helios 3D Plant and Radiative Transfer Modeling Framework." In: *Plant Phenomics* 6 (2024), p. 0189 DOI: 10.34133/plantphenomics.0189.

[Leu95] R. Leuning. "A critical appraisal of a combined stomatal-photosynthesis model for C3 plants." In: *Plant, Cell and Environment* 18.4 (1995), pp. 339–355. ISSN: 0140-7791 DOI: https://doi.org/10.1111/j.1365-3040.1995.tb00370.x.

[Li+24] X. Li, J. Park, C. Reberg-Horton, S. Mirsky, E. Lobaton, and L. Xiang. "Photorealistic Arm Robot Simulation for 3D Plant Reconstruction and Automatic Annotation using Unreal Engine 5." In: *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2024, pp. 5480–5488 DOI: 10.1109/CVPRW63382.2024.00557.

[Lip21] T. Lippert. "Modular Supercomputing for Neuroscience." In: *Brain-Inspired Computing: 4th International Workshop, BrainComp 2019, Cetraro, Italy, July 15–19, 2019, Revised Selected Papers*. Vol. 12339. Springer Nature. 2021, p. 63 DOI: 10.1007/978-3-030-82427-3.

[Liu+22] D. Liu, L. Wei, Q. Zheng, P. Ding, and Y. Shen. "Design and Implementation of Distributed Rendering System." In: *2022 IEEE Symposia and Workshops on Ubiquitous, Autonomic and Trusted Computing*. 2022, pp. 2366–2371 DOI: 10.1109/SmartWorld-UIC-ATC-ScalCom-DigitalTwin-PriComp-Metaverse56740.2022.00332.

[Lob+17] G. Lobet, I. T. Koevoets, M. Noll, P. E. Meyer, P. Tocquin, L. Pagès, and C. Périlleux. "Using a Structural Root System Model to Evaluate and Improve the Accuracy of Root Image Analysis Pipelines." In: *Frontiers in Plant Science* 8 (2017) DOI: 10.3389/fpls.2017.00447.

[Lob+15] G. Lobet, M. P. Pound, J. Diener, C. Pradal, X. Draye, C. Godin, M. Javaux, D. Leitner, F. Meunier, P. Nacry, T. P. Pridmore, and A. Schnepf. "Root System Markup Language: Toward a Unified Root Architecture Description Language." In: *Plant Physiology* 167.3 (Jan. 2015), pp. 617–627. ISSN: 0032-0889 DOI: 10.1104/pp.114.253625.

[LS20] G. Louarn and Y. Song. "Two decades of functional–structural plant modelling: now addressing fundamental questions in systems biology and predictive ecology." In: *Annals of Botany* 126.4 (2020), pp. 501–509. ISSN: 0305-7364 DOI: 10.1093/aob/mcaa143.

[Mai+19] T. H. Mai, A. Schnepf, H. Vereecken, and J. Vanderborght. "Continuum multiscale model of root water and nutrient uptake from soil with explicit consideration of the 3D root architecture and the rhizosphere gradients." In: *Plant and Soil* 439.1 (June 2019), pp. 273–292. ISSN: 1573-5036 DOI: 10.1007/s11104-018-3890-4.

[Mal+21] Z. Malenovský, O. Regaieg, T. Yin, N. Lauret, J. Guilleux, E. Chavanon, N. Duran, R. Janoutová, A. Delavois, J. Meynier, G. Medjdoub, P. Yang, C. van der Tol, D. Morton, B. D. Cook, and J.-P. Gastellu-Etchegorry. "Discrete anisotropic radiative transfer modelling of solar-induced chlorophyll fluorescence: Structural impacts in geometrically explicit vegetation canopies." In: *Remote Sensing of Environment* 263 (2021), p. 112564. ISSN: 0034-4257 DOI: 10.1016/j.rse.2021.112564.

[MB19] P. Mania and M. Beetz. "A Framework for Self-Training Perceptual Agents in Simulated Photorealistic Environments." In: *2019 International Conference on Robotics and Automation (ICRA)*. 2019, pp. 4396–4402 DOI: 10.1109/ICRA.2019.8793474.

[Mar+22] G. S. Markomanolis, A. Alpay, J. Young, M. Klemm, N. Malaya, A. Esposito, J. Heikonen, S. Bastrakov, A. Debus, T. Kluge, K. Steiniger, J. Stephan, R. Widera, and M. Bussmann. “Evaluating GPU Programming Models for the LUMI Supercomputer.” In: *Supercomputing Frontiers*. Ed. by D. K. Panda and M. Sullivan. Springer International Publishing, 2022 DOI: 10.1007/978-3-031-10419-0_6.

[MBS00] J. Martonchik, C. Bruegge, and A. Strahler. “A review of reflectance nomenclature used in remote sensing.” In: *Remote Sensing Reviews* 19.1-4 (2000), pp. 9–20 DOI: 10.1080/02757250009532407.

[Mat+14] R. C. Mat, A. R. M. Shariff, A. N. Zulkifli, M. S. M. Rahim, and M. H. Mahayudin. “Using game engine for 3D terrain visualisation of GIS data: A review.” In: *IOP Conference Series: Earth and Environmental Science* 20.1 (June 2014), p. 012037 DOI: 10.1088/1755-1315/20/1/012037.

[McC+17] J. McCormac, A. Handa, S. Leutenegger, and A. J. Davison. “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)*. Oct. 2017 DOI: 10.1109/ICCV.2017.292.

[MP43] W. S. McCulloch and W. Pitts. “A logical calculus of the ideas immanent in nervous activity.” In: *The bulletin of mathematical biophysics* 5.4 (Dec. 1943), pp. 115–133. ISSN: 1522-9602 DOI: 10.1007/BF02478259.

[McD+20] T. McDonald, W. Usher, N. Morrical, A. Gyulassy, S. Petruzza, F. Federer, A. Angelucci, and V. Pascuccis. “Improving the Usability of Virtual Reality Neuron Tracing with Topological Elements.” In: *IEEE Transactions on Visualization and Computer Graphics* (2020) DOI: 10.1109/TVCG.2020.3030363.

[Mil+19] B. Mildenhall, P. P. Srinivasan, R. Ortiz-Cayon, N. K. Kalantari, R. Ramamoorthi, R. Ng, and A. Kar. “Local Light Field Fusion: Practical View Synthesis with Prescriptive Sampling Guidelines.” In: *ACM Transactions on Graphics (TOG)* (2019) DOI: 10.1145/3306346.3322980.

[MST15] M. Minervini, H. Scharr, and S. A. Tsaftaris. “Image analysis: the new bottleneck in plant phenotyping.” In: *IEEE signal processing magazine* 32.4 (2015), pp. 126–131 DOI: 10.1109/MSP.2015.2405111.

[Mor+21a] S. Morandage, E. Laloy, A. Schnepf, H. Vereecken, and J. Vanderborght. “Bayesian inference of root architectural model parameters from synthetic field data.” In: *Plant and Soil* 467.1 (2021), pp. 67–89 DOI: 10.1007/s11104-021-05026-4.

[Mor+21b] S. Morandage, E. Laloy, A. Schnepf, H. Vereecken, and J. Vanderborght. “Bayesian inference of root architectural model parameters from synthetic field data.” In: *Plant and Soil* 467.1 (2021), pp. 67–89 DOI: 10.1007/s11104-021-05026-4.

[Mor+16] K. Moreland, C. Sewell, W. Usher, L.-t. Lo, J. Meredith, D. Pugmire, J. Kress, H. Schroots, K.-L. Ma, H. Childs, M. Larsen, C.-M. Chen, R. Maynard, and B. Geveci. "VTK-m: Accelerating the Visualization Toolkit for Massively Threaded Architectures." In: *IEEE Computer Graphics and Applications* 36.3 (2016), pp. 48–58 DOI: 10.1109/MCG.2016.48.

[MKE20] M. Mousavi, A. Khanal, and R. Estrada. "AI Playground: Unreal Engine-Based Data Ablation Tool for Deep Learning." In: *Advances in Visual Computing*. Ed. by G. Bebis, Z. Yin, E. Kim, J. Bender, K. Subr, B. C. Kwon, J. Zhao, D. Kalkofen, and G. Baciu. Cham: Springer International Publishing, 2020, pp. 518–532. ISBN: 978-3-030-64559-5 DOI: 10.1007/978-3-030-64559-5_41.

[Mu+20] J. Mu, W. Qiu, G. D. Hager, and A. L. Yuille. "Learning From Synthetic Animals." In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020 DOI: 10.48550/arXiv.1912.08265.

[Mwa+22] C. Mwase, Y. Jin, T. Westerlund, H. Tenhunen, and Z. Zou. "Communication-efficient distributed AI strategies for the IoT edge." In: *Future Generation Computer Systems* 131 (2022), pp. 292–308. ISSN: 0167-739X DOI: 10.1016/j.future.2022.01.013.

[NH10] V. Nair and G. E. Hinton. "Rectified linear units improve restricted boltzmann machines." In: *Proceedings of the 27th International Conference on International Conference on Machine Learning*. ICML'10. Haifa, Israel: Omnipress, 2010, pp. 807–814. ISBN: 9781605589077 DOI: 10.5555/3104322.3104425.

[Nor+21] C. G. Northcutt, M. ChipBrain, C. A. Athalye, and J. Mueller. "Pervasive Label Errors in Test Sets Destabilize Machine Learning Benchmarks." In: *35th Conference on Neural Information Processing Systems (NeurIPS 2021)*. 2021 DOI: 10.48550/arXiv.2103.14749.

[Pau19] S. Paulus. "Measuring crops in 3D: using geometry for plant phenotyping." In: *Plant Methods* (Sept. 2019) DOI: 10.1186/s13007-019-0490-0.

[Pfl+17] D. Pflugfelder, R. Metzner, D. van Dusschoten, R. Reichel, S. Jahnke, and R. Koller. "Non-invasive imaging of plant roots in different soils using magnetic resonance imaging (MRI)." In: *Plant Methods* 13.8-9 (2017), p. 102 DOI: 10.1186/s13007-017-0252-9.

[PH10] M. Pharr and G. Humphreys. *Physically based rendering: From theory to implementation*. Elsevier, 2010 DOI: 10.1016/C2009-0-30446-8.

[Pou+17a] M. P. Pound, J. A. Atkinson, A. J. Townsend, M. H. Wilson, M. Griffiths, A. S. Jackson, A. Bulat, G. Tzimiropoulos, D. M. Wells, E. H. Murchie, T. P. Pridmore, and A. P. French. "Deep machine learning provides state-of-the-art performance in image-based plant phenotyping." In: *GigaScience* 6.10 (Aug. 2017). gix083. ISSN: 2047-217X DOI: 10.1093/gigascience/gix083.

[Pou+17b] M. P. Pound, J. A. Atkinson, D. M. Wells, T. P. Pridmore, and A. P. French. “Deep Learning for Multi-task Plant Phenotyping.” In: *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*. 2017, pp. 2055–2063 DOI: 10.1109/ICCVW.2017.241.

[Pul+12] K. Pulli, A. Baksheev, K. Kornyakov, and V. Eruhimov. “Real-time computer vision with OpenCV.” In: *Commun. ACM* 55.6 (June 2012), pp. 61–69. ISSN: 0001-0782 DOI: 10.1145/2184319.2184337.

[Qiu+17] W. Qiu, F. Zhong, Y. Zhang, S. Qiao, Z. Xiao, T. S. Kim, and Y. Wang. “Unrealcv: Virtual worlds for computer vision.” In: *Proceedings of the 25th ACM international conference on Multimedia*. 2017, pp. 1221–1224 DOI: 10.48550/arXiv.1609.01326.

[Rag+17] M. Raghu, B. Poole, J. Kleinberg, S. Ganguli, and J. Sohl-Dickstein. “On the Expressive Power of Deep Neural Networks.” In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by D. Precup and Y. W. Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, June 2017, pp. 2847–2854 DOI: 10.48550/arXiv.1606.05336.

[Raj+18] P. Rajpura, A. Aggarwal, M. Goyal, S. Gupta, J. Talukdar, H. Bojinov, and R. Hegde. “Transfer Learning by Finetuning Pretrained CNNs Entirely with Synthetic Images.” In: *Computer Vision, Pattern Recognition, Image Processing, and Graphics*. Ed. by R. Rameshan, C. Arora, and S. Dutta Roy. Singapore: Springer Singapore, 2018, pp. 517–528. ISBN: 978-981-13-0020-2 DOI: 10.1007/978-981-13-0020-2_45.

[Rob+17] M. Roberts, D. Dey, A. Truong, S. Sinha, S. Shah, A. Kapoor, P. Hanrahan, and N. Joshi. “Submodular Trajectory Optimization for Aerial 3D Scanning.” In: (2017) DOI: 10.48550/arXiv.1705.00703.

[Rob+21] M. Roberts, J. Ramapuram, A. Ranjan, A. Kumar, M. A. Bautista, N. Paczan, R. Webb, and J. M. Susskind. “Hypersim: A Photorealistic Synthetic Dataset for Holistic Indoor Scene Understanding.” In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2021, pp. 10912–10922 DOI: 10.48550/arXiv.2011.02523.

[Rod+22] M. Rodekamp, E. Berkowitz, C. Gäntgen, S. Krieg, T. Luu, and J. Ostmeyer. “Mitigating the Hubbard Sign Problem with Complex-Valued Neural Networks.” In: (Mar. 2022). arxiv preprint DOI: 10.48550/arXiv.2203.00390.

[RFB15] O. Ronneberger, P. Fischer, and T. Brox. “U-net: Convolutional networks for biomedical image segmentation.” In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241 DOI: 10.1007/978-3-319-24574-4_28.

[RM20] M. Rostamza and G. G. McNickle. “A global database of photosynthesis model parameters, and phylogenetically controlled analysis of photosynthetic responses from every major terrestrial plant clade.” In: *bioRxiv* (2020), p. 2020.10.06.328682 DOI: 10.1101/2020.10.06.328682.

[Sal+24] K. Salesin, K. D. Knobelspiesse, J. Chowdhary, P.-W. Zhai, and W. Jarosz. "Unifying radiative transfer models in computer graphics and remote sensing, Part I: A survey." In: *Journal of Quantitative Spectroscopy and Radiative Transfer* 314 (2024), p. 108847. ISSN: 0022-4073 DOI: https://doi.org/10.1016/j.jqsrt.2023.108847.

[SCN06] A. Saxena, S. Chung, and A. Ng. "Learning Depth from Single Monocular Images." In: *Advances in Neural Information Processing Systems*. Ed. by Y. Weiss, B. Schölkopf, and J. Platt. Vol. 18. MIT Press, 2006 DOI: book/10.5555/109230.

[Sch+18a] A. Schnepf, K. Huber, M. Landl, F. Meunier, L. Petrich, and V. Schmidt. "Statistical Characterization of the Root System Architecture Model CRootBox." In: *Vadose Zone Journal* 17.1 (2018), p. 170212 DOI: 10.2136/vzj2017.12.0212.

[Sch+18b] A. Schnepf, D. Leitner, M. Landl, G. Lobet, T. H. Mai, S. Morandage, C. Sheng, M. Zörner, J. Vanderborght, and H. Vereecken. "CRootBox: a structural–functional modelling framework for root systems." In: *Annals of botany* 121.5 (2018), pp. 1033–1053 DOI: 10.1093/aob/mcx221.

[Sch+20] A. Schnorr, D. N. Helmrich, D. Denker, T. W. Kuhlen, and B. Hentschel. "Feature Tracking by Two-Step Optimization." In: *IEEE Transactions on Visualization and Computer Graphics* 26.6 (2020), pp. 2219–2233 DOI: 10.1109/TVCG.2018.2883630.

[Sch+03] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. "RTP: A Transport Protocol for Real-Time Applications." In: *Standard of the Internet Engineering Task Force* (2003) DOI: 10.17487/RFC1889.

[Sed+19] R. Sedona, G. Cavallaro, J. Jitsev, A. Strube, M. Riedel, and J. A. Benediktsson. "Remote Sensing Big Data Classification with High Performance Distributed Deep Learning." In: *Remote Sensing* 11.24 (2019). ISSN: 2072-4292 DOI: 10.3390/rs11243056.

[Sel+25] T. Selzner, D. N. **Baker**, M. Landl, D. Leitner, G. Lobet, and A. Schnepf. "Coupling of MRI and modelling." In: *NMR in Plants and Soils*. Ed. by A. Pohlmeier, S. Haber-Pohlmeier, and S. Stapf. Royal Society of Chemistry, 2025. Chap. 16, in press.

[Sel+23] T. Selzner, J. Horn, M. Landl, A. Pohlmeier, D. N. **Helmrich**, K. Huber, J. Vanderborght, H. Vereecken, S. Behnke, and A. Schnepf. "3D U-Net Segmentation Improves Root System Reconstruction from 3D MRI Images in Automated and Manual Virtual Reality Work Flows." In: *Plant Phenomics* 5 (2023), p. 0076 DOI: 10.34133/plantphenomics.0076.

[SB14] S. Shalev-Shwartz and S. Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014 DOI: doi/10.5555/2621980.

[SS17] P. Sharma and A. Singh. “Era of deep neural networks: A review.” In: *2017 8th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*. 2017, pp. 1–5 DOI: 10.1109/ICCCNT.2017.8203938.

[Shu+19] D. Shu, J. Cunningham, G. Stump, S. W. Miller, M. A. Yukish, T. W. Simpson, and C. S. Tucker. “3D Design Using Generative Adversarial Networks and Physics-Based Validation.” In: *Journal of Mechanical Design* 142.7 (Nov. 2019), p. 071701. ISSN: 1050-0472 DOI: 10.1115/1.4045419.

[Sie+14] R. Sievänen, C. Godin, T. M. DeJong, and E. Nikinmaa. “Functional–structural plant models: a growing paradigm for plant studies.” In: *Annals of Botany* 114.4 (Sept. 2014), pp. 599–603. ISSN: 0305-7364 DOI: 10.1093/aob/mcu175.

[Sil+24] I. Silva, H. Silva, F. Botelho, and C. Pendão. “Realistic 3D Simulators for Automotive: A Review of Main Applications and Features.” In: *Sensors* 24.18 (2024). ISSN: 1424-8220 DOI: 10.3390/s24185880.

[Sko+18] P. Skobelev, D. Budaev, N. Gusev, and G. Voschuk. “Designing Multi-agent Swarm of UAV for Precise Agriculture.” In: *Highlights of Practical Applications of Agents, Multi-Agent Systems, and Complexity: The PAAMS Collection*. Ed. by J. Bajo, J. M. Corchado, E. M. Navarro Martínez, E. Osaba Icedo, P. Mathieu, P. Hoffa-Dąbrowska, E. del Val, S. Giroux, A. J. Castro, N. Sánchez-Pi, V. Julián, R. A. Silveira, A. Fernández, R. Unland, and R. Fuentes-Fernández. Cham: Springer International Publishing, 2018, pp. 747–59. ISBN: 978-3-319-94779-2 DOI: 10.1007/978-3-319-94779-2_5.

[Smi+20] A. G. Smith, E. Han, J. Petersen, N. A. F. Olsen, C. Giese, M. Athmann, D. B. Dresbøll, and K. Thorup-Kristensen. “RootPainter: Deep Learning Segmentation of Biological Images with Corrective Annotation.” In: *BioRxiv* (2020) DOI: 10.1101/2020.04.16.044461.

[Sou+21] S. Soualiou, Z. Wang, W. Sun, P. de Reffye, B. Collins, G. Louarn, and Y. Song. “Functional–Structural Plant Models Mission in Advancing Crop Science: Opportunities and Prospects.” In: *Frontiers in Plant Science* 12 (2021). ISSN: 1664-462X DOI: 10.3389/fpls.2021.747142.

[SKD97] K. M. Stanney, R. S. Kennedy, and J. M. Drexler. “Cybersickness is not simulator sickness.” In: *Proceedings of the Human Factors and Ergonomics Society annual meeting*. Vol. 41. 2. SAGE Publications Sage CA: Los Angeles, CA. 1997, pp. 1138–1142 DOI: 10.1177/107118139704100292.

[SEL19] E. Suarez, N. Eicker, and T. Lippert. “Modular supercomputing architecture: from idea to production.” In: *Contemporary high performance computing*. CRC Press, 2019, pp. 223–255 DOI: 10.1201/9781351036863-9.

[Tai13] L. Taiz. “Agriculture, plant physiology, and human population growth: past, present, and future.” In: *Theoretical and Experimental Plant Physiology* 25.3 (2013), pp. 167–181 DOI: 10.1590/S2197-00252013000300001.

[Thö21] P. Thörnig. “JURECA: Data Centric and Booster Modules implementing the Modular Supercomputing Architecture at Jülich Supercomputing Centre.” In: *Journal of large-scale research facilities JLSRF* (2021) DOI: 10.17815/jlsrf-7-182.

[Tre+18] J. Tremblay, A. Prakash, D. Acuna, M. Brophy, V. Jampani, C. Anil, T. To, E. Cameracci, S. Boochoon, and S. Birchfield. “Training Deep Networks With Synthetic Data: Bridging the Reality Gap by Domain Randomization.” In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. June 2018 DOI: 10.1109/CVPRW.2018.00143.

[Ush+18] W. Usher, P. Klacansky, F. Federer, P.-T. Bremer, A. Knoll, J. Yarch, A. Angelucci, and V. Pascucci. “A Virtual Reality Visualization Tool for Neuron Tracing.” In: *IEEE Transactions on Visualization and Computer Graphics* 24.1 (2018), pp. 994–1003 DOI: 10.1109/TVCG.2017.2744079.

[Vos+09] J. Vos, J. B. Evers, G. H. Buck-Sorlin, B. Andrieu, M. Chelle, and P. H. B. de Visser. “Functional–structural plant modelling: a new versatile tool in crop science.” In: *Journal of Experimental Botany* 61.8 (Dec. 2009), pp. 2101–2115. ISSN: 0022-0957 DOI: 10.1093/jxb/erp345.

[WMH18] D. Ward, P. Moghadam, and N. Hudson. “Deep leaf segmentation using synthetic data.” In: *arXiv preprint arXiv:1807.10931* (2018) DOI: 10.48550/arXiv.1807.10931.

[Wey+22] J. Weyler, F. Magistri, P. Seitz, J. Behley, and C. Stachniss. “In-Field Phenotyping Based on Crop Leaf and Plant Instance Segmentation.” In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2022, pp. 2725–2734 DOI: 10.1109/WACV51458.2022.00302.

[Xia+23] S. Xiao, S. Fei, Q. Li, B. Zhang, H. Chen, D. Xu, Z. Cai, K. Bi, Y. Guo, B. Li, Z. Chen, and Y. Ma. “The Importance of Using Realistic 3D Canopy Models to Calculate Light Interception in the Field.” In: *Plant Phenomics* 5 (2023), p. 0082 DOI: 10.34133/plantphenomics.0082.

[Yat+22] M. Yates, G. Hart, R. Houghton, M. Torres Torres, and M. Pound. “Evaluation of synthetic aerial imagery using unconditional generative adversarial networks.” In: *ISPRS Journal of Photogrammetry and Remote Sensing* 190 (2022), pp. 231–251. ISSN: 0924-2716 DOI: https://doi.org/10.1016/j.isprsjprs.2022.06.010.

[Zha+20] T. Zhang, L. Xie, L. Wei, Z. Zhuang, Y. Zhang, B. Li, and Q. Tian. *UnrealPerson: An Adaptive Pipeline towards Costless Person Re-identification*. 2020 DOI: arXiv:2012.04268.

[Zha+18] Y. Zhang, W. Qiu, Q. Chen, X. Hu, and A. Yuille. *UnrealStereo: Controlling Hazardous Factors to Analyze Stereo Vision*. Los Alamitos, CA, USA, Sept. 2018 DOI: 10.1109/3DV.2018.00035.

[Zha+24] X. Zhao, J. Qi, J. Jiang, S. Liu, H. Xu, S. Lin, Z. Yu, L. Li, and H. Huang. "Fine-scale retrieval of leaf chlorophyll content using a semi-empirically accelerated 3D radiative transfer model." In: *International Journal of Applied Earth Observation and Geoinformation* 135 (2024), p. 104285. ISSN: 1569-8432 DOI: https://doi.org/10.1016/j.jag.2024.104285.

[Zho+20] X.-R. Zhou, A. Schnepf, J. Vanderborght, D. Leitner, A. Lacointe, H. Vereecken, and G. Lobet. "CPlantBox, a whole-plant modelling framework for the simulation of water- and carbon-related processes." In: *in silico Plants* 2.1 (Jan. 2020). diaa001. ISSN: 2517-5025 DOI: 10.1093/insilicoplants/diaa001.

[Zhu+17] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. "Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks." In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 2242–2251 DOI: 10.1109/ICCV.2017.244.