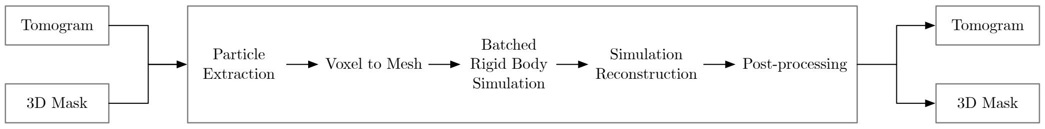
Graphical Abstract

Particle Pack Generation: Synthesise Tomograms with Physics-Based Simulation

Bogong Wang, Andrew Kingston, Philipp Lösel, Warren Creemers



Highlights

Particle Pack Generation: Synthesise Tomograms with Physics-Based Simulation

Bogong Wang, Andrew Kingston, Philipp Lösel, Warren Creemers

- We introduced a novel workflow that generates synthetic tomograms with segmentation ground truths. This method enriches existing tomographic datasets, offering substantial benefits for machine learning and data-intensive applications such as geology and materials science.

Particle Pack Generation: Synthesise Tomograms with Physics-Based Simulation

Bogong Wang[a], Andrew Kingston[b], Philipp Lösel[b], Warren Creemers[a]

[a]*School of Computing, The Australian National University, Canberra, 2617, ACT, Australia*
[b]*Research School of Physics, The Australian National University, Canberra, 2617, ACT, Australia*

Abstract

In the field of geology, 3D imaging of geological particle samples by computed tomography (CT) offers the means for non-destructive analysis. However, obtaining such tomograms with the corresponding segmentation labels remains a significant challenge. This study introduces a novel physics-based simulation workflow that generates synthetic tomograms with segmentation ground truths. The synthetic dataset enriches the existing data resources in the field, offering potential benefits for data-intensive applications, such as machine learning.

Keywords: Computed Tomography, ???

1. Introduction

In geology, the use of computed tomography (CT) to scan geological particle samples has become increasingly common [1, 2, 3]. CT scanning offers the significant advantage of providing 3D non-destructive inspection of large batches of particle samples, often referred to as particle packs. However, to perform comprehensive analyses of these particle packs, it is essential to accurately segment individual particles. This segmentation is crucial for examining specific attributes such as composition and physical properties.

As the utilisation of machine learning expands, there is an increasing demand for comprehensive datasets. A significant challenge in this context is acquiring CT volumes or tomograms and the precise segmentation ground truth. Obtaining tomograms involves complex processes, specialised users and sophisticated equipment, which can be costly to run and resource-intensive. Additionally, achieving accurate segmentation with current methods, such as the watershed algorithm, often requires laborious manual corrections.

In this work, we present a novel approach that utilises a physics-based simulation engine to synthesise tomograms with corresponding segmentation ground truths. This method generates synthetic particle pack data, enriching the available particle pack datasets in various fields.

2. Background

Synthetic data generation is a promising solution to address the inherent difficulties in collecting realistic and unbiased particle pack datasets.

Simulation-based methods are commonly used in synthetic data generation [4]. These methods employ computational models such as physics simulators or scene rendering engines to generate synthetic data that mimic real-world scenarios or phenomena for custom domains. The core idea is to create a virtual environment that replicates physical dynamics or real-world scenes accurately. Common tools include physical simulation engines such as Bullet Engine [5] and PhysX [6] that are used for their physics modeling capabilities such as rigid body simulation or particle dynamics simulation. Additionally, game engines like Unity [7] and Unreal Engine [8] are also frequently utilised, offering rendering capabilities to simulate realistic scenes or environments.

One of the main advantages of simulation-based methods is their high level of controllability [9]. By allowing precise control over both the data generation process and the characteristics of the generated data, researchers and developers can ensure that the output closely adheres to specific criteria or regulatory requirements. This transparency is crucial in understanding exactly how the data was produced, which is particularly beneficial in regulated industries where proving data integrity and compliance is essential. Additionally, simulation-based methods are advantageous because they do not necessarily require large initial datasets. Unlike methods that rely heavily on existing large volumes of data, simulations can generate valuable synthetic data from theoretical models or smaller existing datasets. This capability is particularly useful in scenarios where real data is scarce or difficult to obtain.

However, simulation-based methods have disadvantages. The complexity of creating accurate simulations represents a significant challenge [4]; it

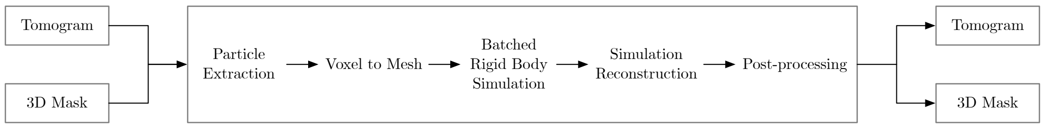


Figure 1: Particle Pack Synthesis Workflow

is often a complex and time-consuming process that demands a thorough understanding of the underlying physical or logical processes that the data aims to represent. Such complexity can limit the speed and scalability of data generation initiatives. Moreover, the novelty of data produced by simulation-based methods is inherently limited by the assumptions and rules embedded within the simulation data and models. As a result, these methods might restrict the ability to uncover unexpected patterns or anomalies within the data, potentially leading to insights that are biased towards pre-conceived notions encoded in the simulation framework. This limitation is particularly relevant in fields where discovering novel insights and understanding previously unrecognised phenomena are the primary objectives.

3. Method

To enrich existing particle pack datasets, we propose a workflow for generating synthetic particle packs, as illustrated in fig. 1. This workflow aims to replicate the physical particle packing process, which involves sifting particles based on their size and then pouring them into a cylinder for a CT scan.

The process commences with the acquisition of a real particle pack volume (from a CT scan) and the corresponding labelled volume. A labelled volume, also known as a “mask”, assigns each particle in the particle pack a unique label. To reduce the computational load in the simulation process, these extracted particle volumes are converted into meshes. A physics simulator then performs rigid body simulations on these meshes. This determines the rotations and positions of particles within the simulated tomogram. Following this, the original particle volumes, masks and the simulation outcomes are used to reconstruct a synthetic tomogram and its 3D mask. The workflow concludes with a post-processing step, where modifications such as adding noise are applied to the reconstructed tomograms and masks to ensure that the output is realistic.

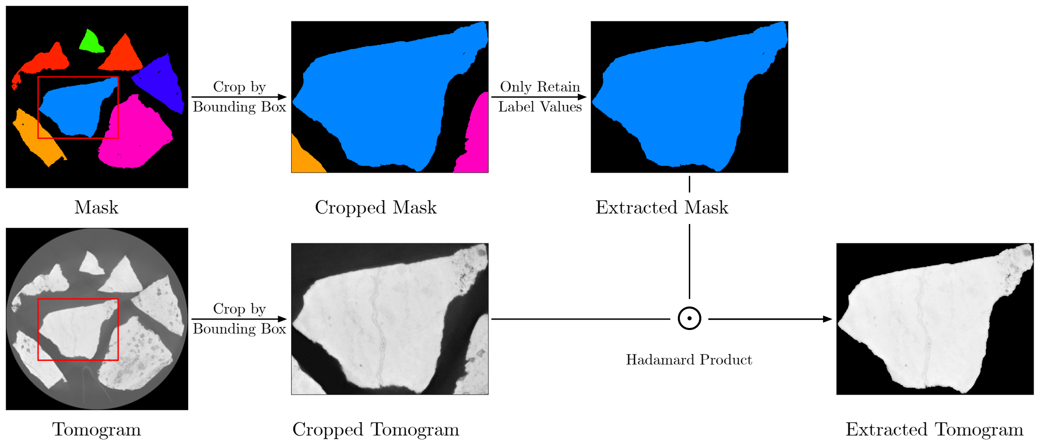


Figure 2: 2D Illustration of Particle Extraction Process

In the following subsections, we will detail each subprocess including particle extraction, voxel to mesh conversion, batched rigid body simulation, simulation reconstruction and postprocessing.

3.1. Particle Extraction

Particle extraction is the process of extracting a volume encompassing each particle from the tomogram and its mask. The input of such a process is a tomogram-label pair, the output is a list of extracted particle volumes and their label volumes.

As shown in fig. 2, the particle extraction process begins by identifying the bounding box of the particle from its mask. The mask and tomogram are then cropped using the bounding box to coarsely isolate from other particles. Next, the binary mask of the target particle is created by eliminating the masks of surrounding particles. This binary mask is subsequently applied to the cropped tomogram, producing a clean tomogram of the selected particle.

3.2. Volume to Mesh

The conversion of extracted mask volumes into mesh structures is necessitated by the requirement for efficiency in rigid body simulations. The primary aim of these physics-based simulations is to accurately model the physics involved in the particle packing process. This include the collisions and interactions among particles. As a result, only the voxel surfaces and total mass are required in the simulation.

The conversion process for the entire tomogram can be efficiently parallelised for all particles. This is achieved by simultaneously extracting the volume of each particle and converting it to a mesh. The transformation of volumes into meshes is facilitated by employing the marching cubes algorithm [10]. Furthermore, to enhance simulation performance through a reduction in vertex points, mesh decimation techniques, as detailed by [11], can be strategically utilised to decrease the vertex count while preserving the external geometry of the particle mesh. After obtaining the meshed particles, particle packing process can then be efficiently simulated.

3.3. Batched Rigid Body Simulation

Using the particle meshes from the previous step, physics-based simulator can be applied to simulate the particle packing process before the CT scan. The particle packing process firstly involves sifting, then placing a number of geological particles into a container. Our goal in the physical simulation is to input these meshes into the simulator in several batches to improve the overall simulation performance, then obtain the placement (location and rotation) of each particle in the simulated particle pack.

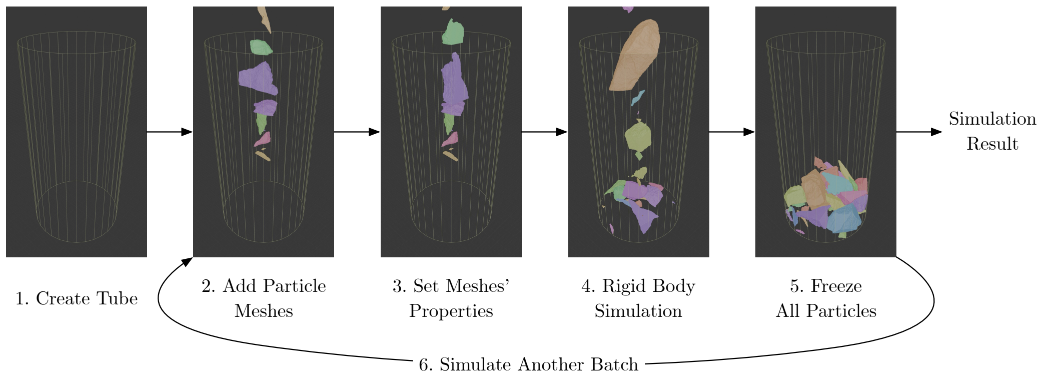


Figure 3: Pipeline of Batched Rigid Body Simulation

The batched rigid body simulation pipeline, depicted in fig. 3, begins by creating a tube—a container resembling the one used in CT scans. We then introduce a set of particle meshes into the simulation environment and assign their properties, including weight, gravitational centre, initial position, and scale (the detail will be shown in 3.3.3). Following this, the rigid body simulation is initiated. After a specified number of time steps, once the particle meshes have settled, the position and orientation of particles are fixed to prevent particles from performing minor adjustments or “damping”, which could significantly affect the performance. This step guarantees that the simulation stays both manageable and scalable, especially crucial when dealing with large quantities of particles, as minor adjustments would otherwise lead to a significant increase in unnecessary computational work. Importantly, it does not significantly compromise the effectiveness of the simulations. Lastly, another simulation batch can be initiated for larger generated tomogram or the simulation process concludes and outputs the simulation result. The final simulation result will contain the translation and the rotation of each particle mesh.

3.3.1. Choice of Simulation

Rigid body simulation is chosen for simulating particle pack processes because it efficiently handles the complexity of shapes and inter-particle dynamics essential for accurately representing hard, non-deformable particles. Unlike soft body simulations, which are better suited for scenarios involving material deformation or bending, rigid body simulation is optimal for processes where particles maintain their shape, as it avoids the unnecessary computational costs associated with modelling deformations. Additionally, while particle simulations might simplify particles to point masses [1] by ignoring the geometry of particle to make the simulation more efficient and scalable, rigid body simulation takes into account the geometric details of each particle. This enables a more precise representation of packing, collisions, and stacking behaviours critical to the process, making rigid body simulation a more efficient and realistic method for simulating particle pack processes.

3.3.2. Choice of Simulator

Blender was selected to perform batched rigid body simulation, driven by the need for a tool that offered a graphical user interface (GUI) for visualisation, a scripting interface for automation, and was relatively efficient.

Compared to PyBullet, Blender’s GUI enables visualisation of simulations, facilitating a more intuitive interaction with the simulation environment. In our study, Python served as the primary programming environment

[1] A point mass is an object of negligible size, treated as if all its mass is concentrated at a single point in space for the purposes of modelling.

for the broader project, using Python ensuring a consistent and unified programming interface. Unity 3D and Unreal Engine, with their comprehensive suites of development tools and GUIs, are ideal for projects that require advanced visualisation and interactive simulation capabilities, though with some trade-offs in scripting flexibility and open-source accessibility. PhysX, known for its high-performance via GPU acceleration in various simulations, is best suited for projects that require high efficiency, but its platform dependency can potentially hinder cross-platform compatibility.

3.3.3. Implementation Detail

In simulation of the particle packing processes, specific settings and procedures are crucial to achieving accurate and reproducible results. This part provides a detailed overview of the implementation specifics used in our simulations, focusing on the settings of the gravitational center, particle properties, and the locking mechanism of the particles.

- The gravitational center plays a pivotal role in our simulation. It is established at the center of the mesh, determined by calculating the object's surface area [12]. This configuration ensures that gravitational effects are realistically simulated, mirroring the natural behavior of particles under the influence of gravity. For ease of reconstruction after exporting the simulation results, the center of the object is repositioned to align with the center of the original bounding box.

- In our implementation, the mass of each particle is estimated based on particles density and the size of the mask, as illustrated in eq. (1)

  $$\text{mass} = \rho V \tag{1}$$

  where ρ is the density of a particle, V is the volume of a particle mask. This method provides a rapid approximation of weight, suitable for scenarios where detailed precision is less critical.

- The initial positioning and scaling of particles are highly dependent on the experimental setup. Adjustments to the initial position can significantly affect the final placement of particles within the volume. Similarly, changing the scale influences the packing density (how tightly particles are packed together), resembling different particle packing configurations.

- As noted above, simulation in batches will enhance the overall performance and manageability. A unique feature of our workflow is the locking mechanism employed after each batch. The particle locking is realised by key frame feature in Blender. Key frames are used extensively in animation and simulation to mark specific points at which certain properties of an object are saved. In our context, we introduce a key frame between two consecutive time steps, t and $t+1$, At time t, particles from the previous batch are still moveable, allowing them to perform rigid body simulation. At time $t+1$, a new batch of particles is introduced into the simulation environment; Simultaneously, movement of particles from the previous batch is disabled. This "locking" of particles after each batch prevents any further adjustments, ensuring stability and reducing computational overhead, particularly when handling a large number of particles.

3.4. Simulation Reconstruction

Given the results of the simulation, the particle pack can be constructed within the tomogram by repositioning each particle. The construction process involves initially gathering volumetric data from the particles' tomograms and masks, alongside the simulation outcomes from batched rigid body simulations. Subsequently, arrays that represents the reconstructed tomogram and its mask are generated. The reconstruction then proceeds in parallel, commencing with the creation of each particle's rotation and translation matrix, followed by the acquisition of volumetric data for the tomogram and mask of each particle. Finally, each particle is rotated and positioned within the resulting tomogram and mask.

Building on the process of reconstructing the particle pack within the tomogram, we next focus on the methodology for rotating tomograms and masks. To achieve rotation, a rotation matrix is constructed based on the specified rotation angles along the x, y, and z axes, employing separate rotation matrices for each axis. The matrix for rotation about the x, y and

z-axis is defined as (2), (3) and (4) respectively:

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix}; \tag{2}$$

$$R_y(\phi) = \begin{bmatrix} \cos(\phi) & 0 & \sin(\phi) \\ 0 & 1 & 0 \\ -\sin(\phi) & 0 & \cos(\phi) \end{bmatrix}; \tag{3}$$

$$R_z(\psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{4}$$

The comprehensive rotation matrix, integrating rotations about all three axes, is derived by multiplying the individual matrices in the sequence $R = R_z(\psi)R_y(\phi)R_x(\theta)$. To retain the original 3D volume after rotation and ensure that no part of the image is lost, padding is necessary. The process can be understood in three main steps:

1. Each vertex V_i of the bounding box of the volume is transformed using the combined rotation matrix R to obtain its new position V_i':

$$V_i' = R \cdot V_i, \tag{5}$$

   where V_i is a vector representing the coordinates of the i-th vertex of the volume's bounding box.
2. After the rotation, the dimensions of the bounding box are determined by calculating the minimum and maximum values of the transformed vertices' coordinates along each axis. The differences between these values give us the dimensions of the bounding box:

$$L = \lceil \max(X') - \min(X') \rceil; \tag{6}$$

$$W = \lceil \max(Y') - \min(Y') \rceil; \tag{7}$$

$$H = \lceil \max(Z') - \min(Z') \rceil, \tag{8}$$

   where X', Y' and Z' are the x, y, and z-coordinates of rotated vertices.
3. Finally, the size of padding applied to each direction can be calculated as:

$$x \text{ direction padding} = 2 \times (L - L_0); \tag{9}$$

$$y \text{ direction padding} = 2 \times (W - W_0); \tag{10}$$

$$z \text{ direction padding} = 2 \times (H - H_0), \tag{11}$$

where L_0, W_0 and H_0 are the original size of the volume's bounding box.

This padding step ensures that the original volume is completely enclosed within the new dimensions post-rotation, preserves the integrity of the 3D volume. Lastly, having this rotation matrix and the size after rotation, an affine transformation with interpolation is then applied to the original tomogram and mask to generate a rotated particle volume.

3.5. Post-processing

Post-processing plays a crucial role in bridging the gap between synthetic and real-world data, enhancing the realism of synthetic tomograms. As real tomograms are rarely perfect, they often contain defects resulting from equipment limitations, photon scarcity, or environmental interference.

One fundamental post-processing step is the addition of noise. Gaussian noise, speckle noise, and Poisson noise are commonly added to replicate these real-world imperfections, thereby produce a more realistic synthetic data. Another aspect of realism is the presence of artefacts, which can also be deliberately introduced into synthetic tomograms. These artefacts might include motion blur, ring effects, or beam hardening, which are common in tomographic imaging. By simulating these artefacts, synthetic datasets can closely mimic the challenges faced when processing real-world data, providing an invaluable ground truth for developing algorithms robust to these artefacts. An example of the post-processed synthesised slice is shown in fig. 4. After post-processing, the synthetic slice appears more realistic.

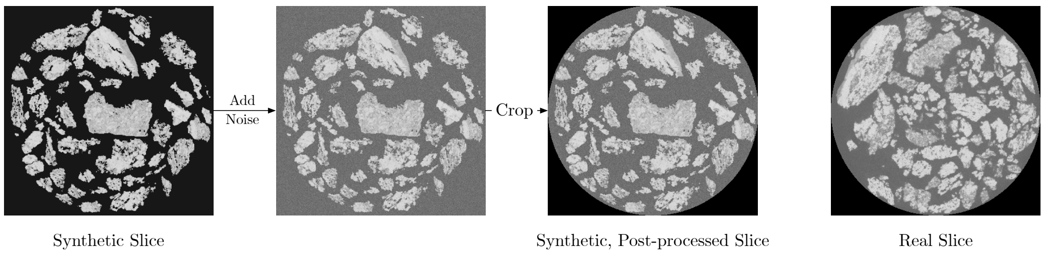


Figure 4: An Example of Postprocessing Slice in Simulated Particle Pack Data

4. Experiment & Discussions

Having constructed synthetic particle packs as described in Section 3. We will conduct a couple of experiments to investigate the efficacy of using synthetic particle packs to augment existing training datasets for 2D particle segmentation. This augmentation approach is particularly valuable for scenarios where acquiring large volumes of real data is impractical due to constraints such as cost, time, or accessibility. As synthetic data can be designed to cover a wide range of variations and complexities not always present in real-world data, potentially filling gaps in the dataset that hinder the model's ability to learn diverse features and scenarios.

4.1. Experiment Setup

To effectively train and evaluate our model, we partitioned each tomogram into training and testing sets. We specifically avoided random selection of slices for the testing set due to the high similarity between consecutive slices within the particle pack. This similarity can lead to overestimation of model performance, as the model might simply be recognising similar instances it has already encountered in adjacent slices, rather than generalising from diverse examples. Therefore, we allocated the central 20% of the particle pack slices to the testing set, and the remaining 80% for training. This will avoid the similarity between training set and testing set.

In this experiment, the YOLOv8m-seg model from Ultralytics [13] will be employed due to its versatility across a wide range of applications as well as optimal balance between computational speed and segmentation accuracy.

Lastly, we selected the Dice score [14], also known as Dice-Sørensen coefficient (shown in eq. (12)) to evaluate segmentation performance of trained models. As it effectively evaluating segmentation accuracy by accounting for both false positives and false negatives, providing a balanced view of under and over-segmentation.

$$\text{Dice Score} = 2 \times \frac{\text{Area of Overlap}}{\text{Total Area}} \tag{12}$$

4.2. Result I: Synthetic Data Improves Segmentation Accuracy

To assess performance improvements with synthetic particle packs, we trained and tested segmentation models using real data alone, and real data augmented with synthetic particle pack data. The results for each model, presented in Table 1, demonstrate that incorporating synthetic data consistently improves segmentation performance for both large and small particles. Specifically, when trained with real large particles augmented with synthetic data, the accuracy increased from 94.5% to 96.0% for large particles and from

	Real Large	Real Small
Real Large	94.5%	77.0%
Real Large + Syn	96.0%	80.1%
Real Small	92.4%	90.1%
Real Small + Syn	94.8%	90.9%

Table 1: Testing results for different tomogram datasets. Each row represents the training set and each column represents the testing set. Real Large refers to a real tomogram dataset with large particles, while Real Small refers to a real tomogram dataset with small particles. Rows include results from training on only the real datasets and combinations with synthetic data (Real Large + Syn, Real Small + Syn). Columns indicate testing on real datasets with large or small particles.

92.4% to 94.8% for small particles. Similarly, training with real small particles combined with synthetic data improved the accuracy on large particles from 77.0% to 80.1%, while the improvement for small particles was more modest, rising from 90.1% to 90.9%. These results indicate that synthetic data enriches the training set with diverse examples, thereby enhances the model's ability to segment tomograms with different particle sizes.

4.3. Result II: Postprocessing Is Crucial In Augmentation

In our experiment, the postprocessing involved adding gaussian noise to the synthetic data. The observed results from Table 2 indicate that after adding noise, the training dataset was successfully augmented, leading to improvements in the models' performance across various test sets. Conversely, without adding noise, the models' performance sometimes degraded, as evidenced by decreases in Dice scores in certain cases.

This can be attributed to the noise making the synthetic slices more realistic and closer to the actual data the model will encounter, thereby reducing the domain gap between synthetic and real data. Without noise, the synthetic data lacks the necessary realism, leading the model to learn features that do not transfer well to real-world scenarios. This discrepancy can cause the model to perform poorly on real data, effectively learning in a different domain than the one it is tested on. This highlights the importance of appropriate postprocessing to make synthetic datasets more realistic, which enhances its utility for augmenting existing datasets.

	Real Large	Real Small
Real Large + Syn w/ Postprocessing	96.0%	80.1%
Real Large + Syn w/o Postprocessing	94.9%	76.9%
Real Small + Syn w/ Postprocessing	94.8%	90.9%
Real Small + Syn w/o Postprocessing	92.4%	88.8%

Table 2: Testing results for real tomogram datasets with large and small particles using different training sets, with and without postprocessing. Each row indicates a different training configuration: Real Large + Syn or Real Small + Syn, combined with postprocessing (w/) or without postprocessing (w/o). Columns show testing results on real datasets: Real Large and Real Small. Performance varies depending on the use of postprocessing and particle size of the testing set.

5. Conclusion

This paper presents a comprehensive workflow for generating synthetic particle pack datasets through physics-based simulation. Our approach addresses the challenges of acquiring accurate CT tomograms and segmentation ground truths for training and evaluation of segmentation algorithms by leveraging simulations to produce enriched and realistic datasets. The integration of noise and artefacts in post-processing further bridges the gap between synthetic and real-world data, offering a data generation platform for future research.

Our experimental results validate the effectiveness of our synthetic particle pack generation workflow. By augmenting real datasets with synthetic data, we achieved consistent improvements in segmentation accuracy across different particle sizes, as demonstrated by higher Dice scores in Table 1. The integration of post-processing techniques, specifically the addition of noise, was crucial in bridging the gap between synthetic and real-world data, leading to enhanced model performance (Table 2). These findings highlight the potential of our approach to enrich annotated particle pack data for training and validation of segmentation algorithms.

In the current synthetic particle-pack-generation workflow, we rely on manually labelled particle packs, which limits the versatility of the particle data. To eliminate this dependency and increase data diversity, we propose two improvement directions. The first approach involves using classical methods such as free-form deformation (FFD) [15] to alter particles' shape while retaining their core characteristics. This method is relatively fast, easy, and deterministic but still relies on the existing particle data. The second approach involves employing advanced machine-learning techniques, such as diffusion models [16] or generative adversarial networks (GANs) [17]. These models would first learn the features of the particle data and then generate entirely synthetic particle data.

References

[1] M. Van Geet, R. Swennen, M. Wevers, Quantitative analysis of reservoir rocks by microfocus x-ray computerised tomography, Sedimentary Geology 132 (1) (2000) 25–36. doi:https://doi.org/10.1016/S0037-0738(99)00127-X.
URL `https://www.sciencedirect.com/science/article/pii/S003707389900127X`

[2] V. Cnudde, J. Dewanckele, W. De Boever, L. Brabant, T. De Kock, 3d characterization of grain size distributions in sandstone by means of x-ray computed tomography, in: Quantitative mineralogy and microanalysis of sediments and sedimentary rocks, Vol. 42, Mineralogical Association of Canada (MAC), 2012, pp. 99–113.

[3] M. Warlo, G. Bark, C. Wanhainen, A. Butcher, F. Forsberg, H. Lycksam, J. Kuva, Multi-scale x-ray computed tomography analysis to aid automated mineralogy in ore geology research, Frontiers in Earth Science 9 (2021) 789372. doi:10.3389/feart.2021.789372.

[4] C. M. De Melo, A. Torralba, L. Guibas, J. DiCarlo, R. Chellappa, J. Hodgins, Next-generation deep learning based on simulators and synthetic data, Trends in Cognitive Sciences 26 (2) (2022) 174–187. doi:10.1016/j.tics.2021.11.008.

[5] E. Coumans, Y. Bai, PyBullet, a Python module for physics simulation for games, robotics and machine learning, `http://pybullet.org` (2016–2024).

[6] NVIDIA Corporation, NVIDIA PhysX, `https://developer.nvidia.com/physx-sdk` (2024).

[7] Unity Technologies, Unity Game Engine, `https://unity.com` (2024).

[8] Epic Games, Unreal Engine, `https://www.unrealengine.com` (2024).

[9] M. Müller, V. Casser, J. Lahoud, N. Smith, B. Ghanem, Sim4CV: A Photo-Realistic Simulator for Computer Vision Applications, International Journal of Computer Vision 126 (9) (2018) 902–919. arXiv:1708.05869, doi:10.1007/s11263-018-1073-7.

[10] W. E. Lorensen, H. E. Cline, Marching cubes: A high resolution 3d surface construction algorithm, SIGGRAPH Comput. Graph. 21 (4) (1987) 163–169. doi:10.1145/37402.37422.
URL `https://doi.org/10.1145/37402.37422`

[11] M. Garland, P. S. Heckbert, Surface simplification using quadric error metrics, in: Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '97, ACM Press/Addison-Wesley Publishing Co., USA, 1997, pp. 209–216. doi:10.1145/258734.258849.
URL `https://doi.org/10.1145/258734.258849`

[12] Blender Foundation, Blender Python API: Object Operators, `https://docs.blender.org/api/current/bpy.ops.object.html#bpy.ops.object.origin` (2023).

[13] G. Jocher, A. Chaurasia, J. Qiu, Ultralytics YOLO (Jan. 2023).
URL `https://github.com/ultralytics/ultralytics`

[14] L. R. Dice, Measures of the amount of ecologic association between species, Ecology 26 (1945) 297–302.
URL `https://api.semanticscholar.org/CorpusID:53335638`

[15] T. W. Sederberg, S. R. Parry, Free-form deformation of solid geometric models, in: Proceedings of the 13th annual conference on Computer graphics and interactive techniques, 1986, pp. 151–160.

[16] J. Ho, A. Jain, P. Abbeel, Denoising diffusion probabilistic models (2020). arXiv:2006.11239.
URL `https://arxiv.org/abs/2006.11239`

[17] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative Adversarial Networks (Jun. 2014). arXiv:1406.2661.