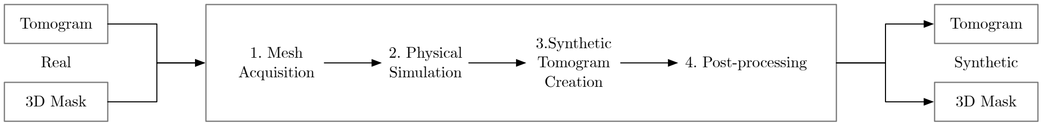
Graphical Abstract

Synthetic Data Generation for Augmentation and Testing in Geological Tomographic Segmentation

Bogong Wang, Andrew Kingston, Philipp D. Lösel, Warren Creemers



Highlights

Synthetic Data Generation for Augmentation and Testing in Geological Tomographic Segmentation

Bogong Wang, Andrew Kingston, Philipp D. Lösel, Warren Creemers

- We introduced a novel workflow that generates synthetic tomograms with segmentation ground truths. This method enriches existing tomographic datasets, offering capabilities for dataset augmentation and the evaluation of geological segmentation model.

Synthetic Data Generation for Augmentation and Testing in Geological Tomographic Segmentation

Bogong Wang[a], Andrew Kingston[b], Philipp D. 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

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 enhances existing resources in the field, providing support for data augmentation and testing for geological tomographic segmentation.

Keywords: Computed Tomography, Synthetic Data Generation

1. Introduction

In the field of 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.

A significant challenge in this context is acquiring CT volumes or tomograms and the precise segmentation ground truth. Obtaining tomograms involves complex processes, specialised technicians and sophisticated equipment, which is both costly to run and resource-intensive. Additionally,

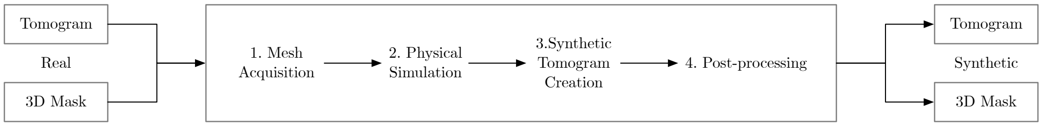


Figure 1: Particle Pack Synthesis Workflow

achieving accurate segmentation with current methods, such as the watershed algorithm, often requires laborious manual corrections.

In this work, we introduce a novel approach using a physics-based simulation engine to generate 3D particle pack tomograms alongside their segmentation ground truths. This method serves as an effective tool for augmenting tomographic segmentation datasets and validating segmentation algorithms.

2. Background

Synthetic data has been used for more than decades to enrich existing datasets and benchmark the performance of computer vision algorithms, including segmentation [4]. This process involves creating data that resemble actual samples in terms of structure, distribution, and key characteristics, yet they are synthetic and generated through computational methods.

Simulation-based methods are commonly used in synthetic data generation [5]. 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 [6] and PhysX [7] that are commonly used for their physics modeling capabilities such as rigid body simulation [8]. Additionally, game engines like Unity [9] and Unreal Engine [10] are also frequently utilised, offering rendering capabilities to simulate realistic scenes or environments [11, 12].

3. Method

To illustrate our proposed workflow for generating synthetic particle packs (see Figure 1), we designed a process that closely replicates the physical steps involved in preparing particles for CT scanning. This workflow encompasses four key stages:

1. Mesh Acquisition: Extract particles from a CT-scanned particle pack volume and convert them into meshes.
2. Physical Simulation: Conduct rigid body simulations on the particle meshes using a physics simulator to obtain their rotations and positions in the simulated tomogram.
3. Synthetic Tomogram Creation: Create a synthetic particle pack tomogram and its 3D mask by combining the original particle volumes, masks, and simulation results.
4. Post-Processing: Introduce modifications, such as noise, to make the synthetic tomograms and masks realistic proxies for real-world data.

3.1. Mesh Acquisition

Mesh acquisition involves extracting each particle's volume from a tomogram and converting it into a mesh for rigid body simulations. The process starts with a particle pack tomogram and its mask. Using the mask, we isolate each particle's volume and label volume, then convert these into mesh format with marching cubes algorithm [13]. To optimise for simulations, particle meshes can be simplified using mesh decimation techniques [14]. This would reducing vertex counts while remaining essential geometry and improve simulation efficiency.

3.2. Physical Simulation

The real particle packing process begins with sieving to separate particles by size. Selected particles are then placed into a container to form a particle pack, ready for CT scanning. Our goal in the physical simulation is to replicate this process. This is achieved by inputting particle meshes with specific size then using simulator to mimic the simulation process, lastly obtain the placement (location and rotation) of each particle in the simulated particle pack.

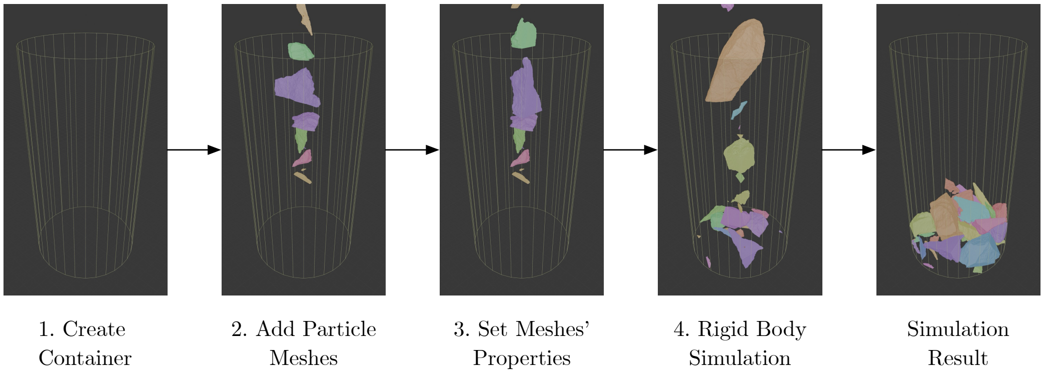


Figure 2: Pipeline of Batched Rigid Body Simulation

The batched rigid body simulation pipeline includes four steps (illustrated in Figure 2):

1. Create Container: Construct a tube-like container, similar to the one used in CT scans.
2. Add Particle Meshes: Introduce particle meshes into the simulation environment.
3. Set Meshes' Properties: Assign each particle mesh properties such as weight, initial position, and scale (details in 3.2.2).
4. Rigid Body Simulation: Start the simulation and allow the particle meshes to settle after a defined number of timesteps. The simulator will then output the final results, which contains each mesh's translation and rotation. Rigid body simulation is used to model this process because it approximates the motion and interaction of particles under realistic conditions, capturing essential dynamics such as collision, gravity, and friction without deforming the individual meshes.

3.2.1. Choice of Simulator

After evaluating multiple options, including Unity 3D [9], Unreal Engine [10], and PhysX [7], we chose Blender for conducting batched rigid body simulations. Blender stood out due to its accessible graphical user interface (GUI), which facilitated the visual inspection of packed particle formations. This helped us verify whether the simulation accurately represents the desired physical characteristics. Additionally, Blender's scripting interface allowed seamless integration into a Python-driven pipeline, ensuring that the project maintained a consistent and adaptable programming environment.

3.2.2. Implementation Detail

In simulation of the particle packing processes, specific settings and procedures are crucial to achieving accurate and reproducible results. This part provides an overview of the implementation specifics used in our simulations.

- Particle weights affects collision between particles and container in our particle packing simulation. In our implementation, we assume each particle has a uniformity and even quality. Then the mass of each particle is estimated based on particles' density and the size of the mask. This method provides a rapid approximation of weight, suitable for scenarios where detailed precision is less critical.

- The initial positioning and scaling of particle meshes are highly dependent on the experimental needs. These adjustments significantly affect the final placement of particles within the volume.

3.3. Synthetic Tomogram Creation

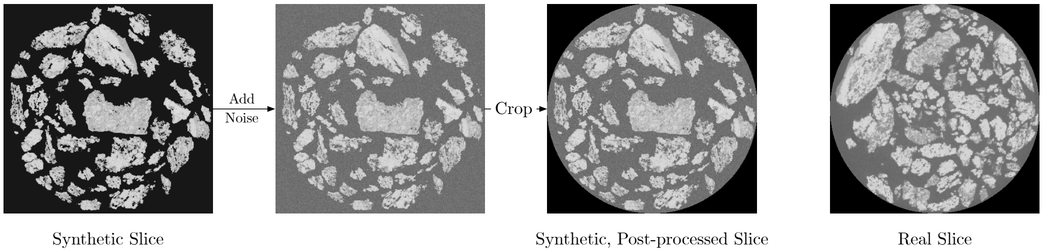
Based on the simulation results and particle volumes, the synthetic particle pack can be created. The synthetic tomogram creation involves three steps:

1. Creating the target simulation container
2. Rotating each particle volume according to the simulation results
3. Placing each rotated particle volume in the container based on the simulation results

3.4. Post-processing

Post-processing plays a crucial role in bridging the gap between synthetic and real-world data, enhancing the realism of synthetic tomograms. In our implementation, Gaussian noise are commonly added to replicate these real-world imperfections, thereby produce a more realistic synthetic data. Artefacts such as 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. 3. After post-processing, the synthetic slice appears more realistic.



Synthetic Slice

Synthetic, Post-processed Slice

Real Slice

Figure 3: 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 will be exemplified by comparing the performance gap between models trained on raw and augmented datasets.

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 [15] 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 [16], also known as Dice-Sørensen coefficient (shown in eq. (1)) 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.

$$\text{Dice Score} = 2 \times \frac{\text{Area of Overlap}}{\text{Total Area}} \tag{1}$$

	Post-processing	Real Large	Real Small
Real Large	n/a	94.5%	77.0%
Real Large + Syn	✗	94.9%	79.6%
Real Large + Syn	✓	96.0%	80.1%
Real Small	n/a	92.4%	90.1%
Real Small + Syn	✗	92.4%	88.8%
Real Small + Syn	✓	94.8%	90.9%

Table 1: Segmentation performance between models trained on different datasets. Each row represents the training set, and each column represents the testing set (except for the "Post-processing"" column). The "Post-processing" column indicates whether the augmented dataset has been post-processed. 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.

4.2. Result I: Postprocessing Is Crucial In Augmentation

In our experiment, the post-processing involved adding Gaussian noise to the synthetic data. The observed results from Table 1 indicate that after post-processing, 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 (from 90.1% to 88.8%).

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.

4.3. Result II: 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 improves segmentation performance under most scenarios. 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 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.

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 particle pack tomograms and segmentation ground truths in geology. These tomogram-mask pairs can be used to augment existing segmentation datasets and evaluating segmentation algorithms. 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 1). 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) [17] 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 [18] or generative adversarial networks (GANs) [19]. 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.

[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] S. R. Richter, V. Vineet, S. Roth, V. Koltun, Playing for data: Ground truth from computer games (2016). arXiv:1608.02192.
URL `https://arxiv.org/abs/1608.02192`

[5] 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.

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

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

[8] K. Greff, F. Belletti, L. Beyer, C. Doersch, Y. Du, D. Duckworth, D. J. Fleet, D. Gnanapragasam, F. Golemo, C. Herrmann, T. Kipf, A. Kundu, D. Lagun, I. Laradji, Hsueh-Ti, Liu, H. Meyer, Y. Miao,

D. Nowrouzezahrai, C. Oztireli, E. Pot, N. Radwan, D. Rebain, S. Sabour, M. S. M. Sajjadi, M. Sela, V. Sitzmann, A. Stone, D. Sun, S. Vora, Z. Wang, T. Wu, K. M. Yi, F. Zhong, A. Tagliasacchi, Kubric: A scalable dataset generator (2022). arXiv:2203.03570.
URL https://arxiv.org/abs/2203.03570

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

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

[11] H. Lee, J. Jeon, D. Lee, C. Park, J. Kim, D. Lee, Game engine-driven synthetic data generation for computer vision-based safety monitoring of construction workers, Automation in Construction 155 (2023) 105060. doi:https://doi.org/10.1016/j.autcon.2023.105060.
URL https://www.sciencedirect.com/science/article/pii/S0926580523003205

[12] S. Borkman, A. Crespi, S. Dhakad, S. Ganguly, J. Hogins, Y.-C. Jhang, M. Kamalzadeh, B. Li, S. Leal, P. Parisi, C. Romero, W. Smith, A. Thaman, S. Warren, N. Yadav, Unity perception: Generate synthetic data for computer vision (2021). arXiv:2107.04259.
URL https://arxiv.org/abs/2107.04259

[13] 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

[14] 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

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

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

[17] 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.

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

[19] 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.