CGI2023 manuscript No.151
(will be inserted by the editor)

XProtoSphere: an eXtended multi-sized sphere packing algorithm driven by particle size distribution

Xu Wang[1] · Makoto Fujisawa[1] · Masahiko Mikawa[1]

Abstract The sphere packing problem, which involves filling an arbitrarily shaped geometry with the maximum number of non-overlapping spheres, is a critical research challenge. ProtoSphere is a prototype-oriented algorithm designed for solving sphere packing problems. Due to its easily parallelizable design, it exhibits high versatility and has wide-ranging applications. However, the controllable regulation of particle size distribution (PSD) produced by ProtoSphere is often neglected, which limits its application on algorithm. This paper proposes a novel PSD-driven technique that extends the ProtoSphere algorithm to achieve multi-sized sphere packing with distribution-specific characteristics, as dictated by a pre-defined cumulative distribution function. The proposed approach improves the controllability and flexibility of the packing process, and enables users to generate packing configurations that meet their specific requirements. In addition, by combining the relaxation method with the ProtoSphere algorithm, we can further improve the packing density and ensure the average overlap below 1%. Our method generates multi-sized particles that can be used to simulate the behavior of various granular materials, including sand-like and clay-like soils.

Keywords ProtoSphere · Multi-Sized Sphere Packing · Particle Size Distribution · Discrete Element-Based Relaxation · Physically-Based Simulation

Xu Wang[1]
University of Tsukuba, Ibaraki, Japan
E-mail: wang.xu.lm@alumni.tsukuba.ac.jp

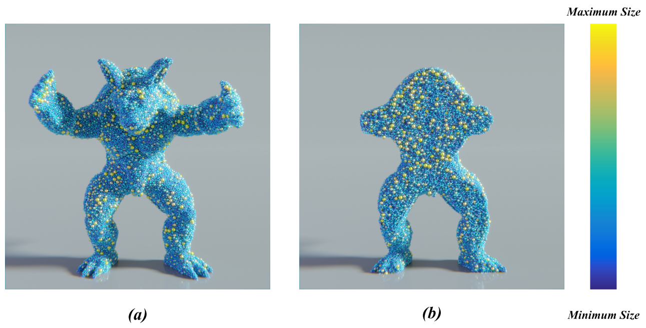

Fig. 1: Armadillo's XProtoSphere packing results (a), as well as corresponding cross-sectional views (b)

1 Introduction

The sphere packing algorithm has broad application to the process of filling particles densely within a given boundary without overlapping. It is commonly utilized to solve the problem of optimal sphere packing in three dimensions in mathematics [15], which is relevant to many fields such as coding theory and cryptography. In materials science, sphere packing algorithms aid in designing materials with desirable physical properties [29]. In wireless communication, they play a critical role in setting up the closest arrangement of antennas to optimize signal quality and coverage [9], and so forth [17].

In computer graphics community, sphere packing is frequently used for efficient spatial segmentation, collision detection [33], automatic rigging [3] and physically based simulation for granular materials [5,31]. In particular, granular material simulations require non-overlapping spheres for computational stability and a high packing density to simulate realistic sediment structures. Many of these applications utilize multi sized sphere packing algorithms rather than uniform sphere

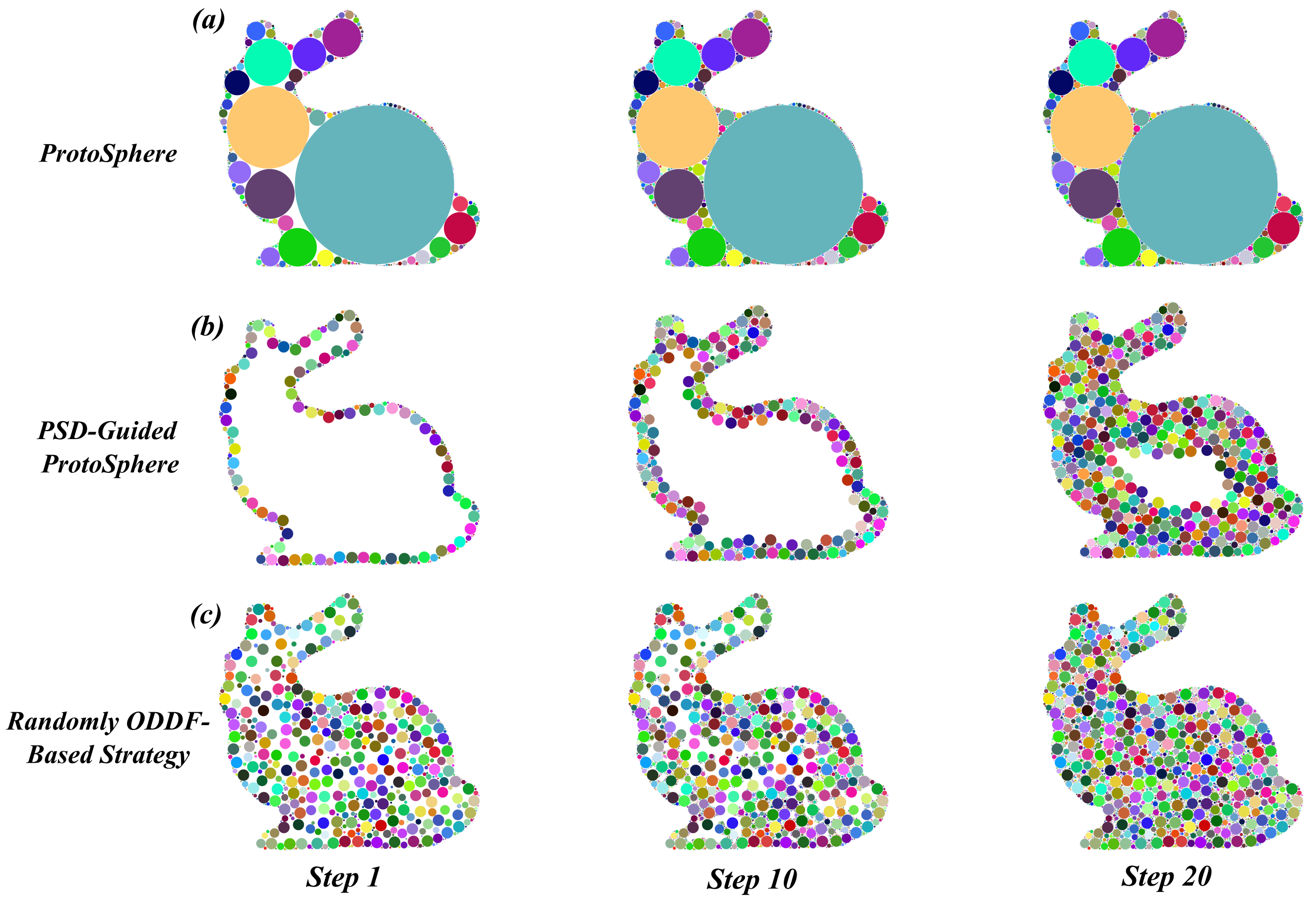

Fig. 2: Comparison of multiple particle insertion types based on the ProtoSphere algorithm, using a 2D Bunny polygon

packing algorithms, due to the flexibility and adaptability that multi-sized spheres offer in effectively modeling complex physical systems with varying particle sizes and densities. In contrast, uniform sphere packing algorithms are often limited in their applicability, as they are primarily suitable for modeling homogeneous particle systems. Furthermore, uniform sphere packing can be easily achieved by fine-tuning certain sampling algorithms, such as the fast Poisson disk sampling method [8] or SPH-based blue noise sampling [21]. However, when these sampling algorithms are applied to multi-sized sphere packing problem, their efficacy in physical simulations may be impeded by a higher overlapping rate and porosity within the sampling space [32]. Therefore, these sampling methods may not be directly applicable for certain physical simulation applications, especially in the Discrete-Element Method (DEM) [11].

In comparison to the sampling algorithms and uniform sphere packing algorithm mentioned before, Weller et al. [34] introduced a multi-sized packing algorithm called ProtoSphere. The ProtoSphere algorithm is inspired by the prototype-based approach in machine learning, and it is capable of efficiently handling arbitrarily shaped objects. Furthermore, the algorithm is highly parallelizable, which makes it a promising option for a wide range of sphere packing problems. However, the inability of the ProtoSphere method to precisely control the particle size distribution of the generated spheres may limit its applicability, in particular granular simulation tasks, such as simulating soil structures containing particles of varying sizes. This limitation may affect the ability to achieve more realistic simulation results.

To address these problems, this paper makes the following contributions:

- An extended algorithm that is based on the standard ProtoSphere algorithm. The proposed algorithm enables users to predetermine a target particle size distribution using a cumulative distribution function, thereby allowing for greater control over the particle size distribution of packed spheres.
- An randomly Offset Discrete Distance Field (ODDF) based strategy is proposed for achieving faster con-

vergence of particle size, as well as addressing the issue of boundary expansion towards the center that arises during particle generation using the extended ProtoSphere algorithm.

– A Discrete-Element based particle relaxation method is proposed to improve the packing density (see Figure 1). This method can be integrated with the extended ProtoSphere algorithm and is applicable in physically-based simulations. Compared to the SPH-based particle relaxation method, the approach offers greater stability and can be applied to a wider range of multi-sized particle distributions.

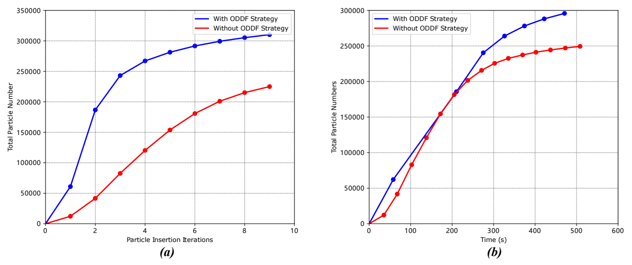

Fig. 3: Comparison of particle packing results obtained with and without the randomly offset discrete element field strategy demonstrates the efficiency of this technique in achieving higher packing densities

2 Related Work

The focus of this review section is to investigate the research and applications of the sphere packing problems in computer graphics and related fields. As is commonly understood, the proper placement of particles plays a vital role in the study of particle-based physically simulation animation. Particle-based methods typically use uniform particles, although in certain circumstances, such as those involving adaptive methods [12,1,35,36] or DEM-related frameworks [30,31], the application of multi-sized particles becomes imperative. With the advantage of being adaptable to arbitrary dimensions and objects, the ProtoSphere method was extended in our study to enable enhanced manipulation of particle size distribution. The resulting approach was explored for its suitability in physically-based simulation scenarios, specifically those involving DEM.

2.1 Sampling Based Approaches

In terms of the sampling and packing problems, although their objectives and application scenarios may differ, both algorithms are fundamentally concerned with the distribution of points in space. Thus, the points generated by the sampling method can be employed as centers for particles in a sphere packing algorithm by assigning each point a suitable radius.

Poisson Disk Sampling Blue noise sampling is a widely used technique in computer graphics due to its ability to produce a uniform distribution [38], making it applicable in a wide variety of applications [40,22,37,20]. Poisson disk sampling, one of its patterns, is widely employed in rendering, geometry processing, and physically-based simulation due to its numerous applications and versatility. In particular, the faster versions of Poisson disk sampling, enhanced by Bridson [8], exhibit greater adaptability to arbitrary dimensions and are comparatively easier to implement. Although this method has demonstrated commendable performance and broad applicability, regulating the packing density of the resulting particles can be challenging, especially in the case of multi-sized particle packing, where particles may overlap.

SPH Based Relaxation In general, the process of sampling particles for a given boundary involves dividing the plane into uniform grids in 2D (or voxelizing in 3D) and generating a particle within each uniform grid [26]. However, this method may produce an irregular distribution of particles near the boundary, which has the potential to obscure the original geometry's boundary information. To overcome this issue, Schechteret et al. [25] employed a Poisson disk relaxation method to facilitate surface and volume sampling. Subsequently, Jiang et al. [21] attempted to use a cohesion-term integrated SPH method for blue noise sampling, which yielded promising outcomes for the relaxation of boundary particles. Moreover, they demonstrated that their method can be combined with adaptive methods [1] to facilitate multi-sized particle sampling. However, these approaches are all based on the SPH-based particle sampling method, and none of them are able to avoid the issue of large overlap between particles. Another problem remains that when particles are not uniformly distributed within the SPH kernel, these algorithms may become unstable.

2.2 Multi-Sized Sphere Packing

The algorithms that relate to multi-sized sphere packing can be categorized into three principal groups, namely geometry separation-based, mesh-based, and Apollonian-based methods:

Geometric Separation Based Approach Geometric separation-based algorithms focus on the task of packing multi-sized particles by randomly inserting particles within regions with low filling rate and iteratively removing any overlapping particles. As an example, Lopes et al. [23,24] proposed a two-dimensional geometric separation method that enables the control of both porosity and particle size distribution through the use of a grid mapping approach. This method achieves high-efficiency particle insertion and removal, thereby facilitating the packing of multi-sized spheres. While geometric separation-based algorithms can control porosity and particle size distribution, their strong stochasticity and the possibility of repeated insertion and removal of particles can make it difficult to ensure their high performance and low error rate when extended to 3D space.

Mesh Based Sphere Packing The objective of these investigations is primarily to develop particle-based, non-overlapping geometries for use in DEM methods. The mesh-based methods utilized involve triangulating (2D) [4] or tetrahedral partitioning (3D) [14] in a given domain, where particles can be positioned at vertices or within the unit geometry [10,18,19]. Recently, a refined approach for multi-sized particle packing has been proposed by Zhang et al. [39], which enables efficient and precise packing of particles for arbitrary 2D geometries. This is achieved by improving upon Cui et al.'s algorithm [10] and utilizing a strategy that involves placing particles at each vertex of the triangular surface. Wang et al. [32] presented a novel method for optimizing porosity to enhance packing density using the Power diagram [2]. This approach allows for the predefinition of particle size distribution, but the outcomes attained by this method suffer from an error rate of 10-20%. Both mesh-based studies face challenges when attempting to extend their algorithms to three-dimensional spaces, due to issues with performance and algorithm instability.

Apollonian Based Method The Apollonian packing algorithm [7] necessitates the initial placement of three mutually tangent discs, with each disc touching the other two. Subsequently, the algorithm iteratively inserts additional discs into the largest available circular cavity within the remaining gap, utilizing this process to generate fractals of arbitrary dimensionality. The ProtoSphere method [34] is inspired by the prototype-guided approach in machine learning and employs an optimization process that utilizes multi-sized particles to fill geometries of arbitrary dimensions. The method yields results comparable to those produced by Apollonian sphere packing, while also being capable of accommodating arbitrary geometries that are challenging to achieve using the latter method. Subsequently, Teuber et al. [28] proposed a GPU-based adaptive grid method that significantly enhances the performance of ProtoSphere. Recently, Bonneau et al. [6] sought to incorporate the multi-sized particles generated by ProtoSphere into the DEM, and achieved this by constraining the random point locations to control the size of the packed particles within a range pre-defined by the user. However, currently available Apollonian-based methods fall short in achieving optimal particle size distribution. To address this limitation, this paper proposes an extension of the ProtoSphere method that enables precise management of the particle size distribution. Additionally, the performance of this extended approach is evaluated in the context of a physically-based particle simulation.

Algorithm 1 Parallel ProtoSphere Algorithm

```
Input: surface $\Omega$ of object $O$, required particle number $N$
Output: a group of particles with radius information
 1: $\mathcal{D}_\Omega \leftarrow$ initialize the discrete distance field
 2: repeat
 3:     $S : \{\boldsymbol{p}_1, \boldsymbol{p}_2, \cdots, \boldsymbol{p}_n\} \leftarrow$ place prototype $\boldsymbol{p}_i$ randomly
        inside grid $c_i$
 4:     for each $\boldsymbol{p}_i$ in $S$ do
 5:         repeat
 6:             $\boldsymbol{q}_c = \arg\min \{\|\boldsymbol{p}_i - \boldsymbol{q}\| : \boldsymbol{q} \in \Omega\}$
 7:             $\boldsymbol{p}_i \leftarrow \boldsymbol{p}_i + \varepsilon(t) \cdot (\boldsymbol{p}_i - \boldsymbol{q}_c)$
 8:             $r_i = \|\boldsymbol{p}_i - \boldsymbol{q}_c\|$
 9:         until $\boldsymbol{p}_i$ has converged
10:     end for
11:     sort $P$ by max radius $r_i$
12:     find $\boldsymbol{p}_k \in P$ that are not overlapped by any $\boldsymbol{p}_i$
13:     insert particles at positions $\boldsymbol{p}_k$ with radii $r_k$
14:     update discrete distance field $\mathcal{D}_\Omega$ by $\Omega \leftarrow \Omega \cup \Omega_{\boldsymbol{p}_k}$
15: until number of inserted particles $> N$
```

3 ProtoSphere

Weller et al. [34] introduced the ProtoSphere algorithm, which is centered around the determination of the sphere radius by measuring the shortest distance between a point and the surface. To be specific, let Ω represent the surface of an arbitrary object O. The point $\boldsymbol{q}_c$ on surface Ω that is closest to point $\boldsymbol{p}$ can be defined as follows:

$$\boldsymbol{q}_c = \arg\min \{\|\boldsymbol{p} - \boldsymbol{q}\| : \boldsymbol{q} \in \Omega\} \tag{1}$$

Here, point $\boldsymbol{p}$ can represent any position located within the interior of object O. The generated particle is centered at point $\boldsymbol{p}$ and has a radius $\|\boldsymbol{p} - \boldsymbol{q}_c\|$.

To approximate Apollonian-like sphere packing, the ProtoSphere algorithm employs a prototype-guided strategy that considers point $\boldsymbol{p}$ as a prototype and seeks to

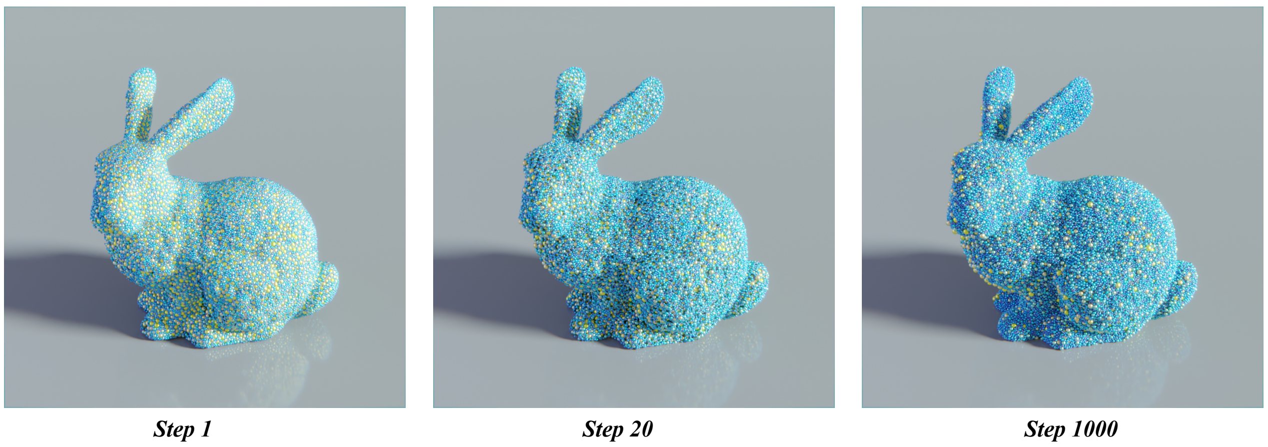

Fig. 4: Visualization of the particle movement process during coupling with the Discrete Element Relaxation

maximize its distance from the surface Ω (thus obtaining a particle with the largest possible radius within object O). The described process can be achieved using the following equation:

$$\boldsymbol{p} \leftarrow \boldsymbol{p} + \varepsilon(t) \cdot (\boldsymbol{p} - \boldsymbol{q}_c) \tag{2}$$

where the cooling function $\varepsilon(t) \in [0,1]$ is employed to facilitate large movements during initial iterations and finer adjustments in subsequent steps. By employing this approach, the standard version of the ProtoSphere algorithm can be implemented through the following three steps: 1. Stochastically generate a point $\boldsymbol{p}$ that lies within the interior of object O. 2. Update the position of point $\boldsymbol{p}$ using Equation(2) iteratively until the convergence criterion is met. 3. Insert a particle with radius $\|\boldsymbol{p} - \boldsymbol{q}_c\|$ at point $\boldsymbol{p}$ and return to the first step. It should be noted that the addition of each new particle necessitates the availability of surface information Ω_p to facilitate the updating of surface $\Omega = \Omega \cup \Omega_p$.

Importantly, the requirement for surface information with each new particle may lead to computational performance issues. Moreover, the standard ProtoSphere algorithm is confined to local optimization for each prototype and does not achieve global optimization. To overcome these limitations, Weller et al. [34] employs a gridding strategy that partitions the interior space of object O. With this approach, a prototype can be positioned within each grid, enabling global optimization by allowing them to move independently. In addition, they enhance the computational efficiency of determining the nearest boundary point by pre-computing the discrete distance field. The parallel version of the ProtoSphere algorithm is presented in Algorithm 1 with a detailed description. Figure 2(a) showcases a two-dimensional result obtained after computation with the parallelized algorithm.

4 Extended ProtoSphere (XProtoSphere)

ProtoSphere presents a superior option in sphere packing algorithms, as it possesses the capability to address arbitrarily shaped and multi-dimensional particle packing challenges. Additionally, its algorithmic implementation is straightforward and lends itself to parallelization, further increasing its competitiveness. Nevertheless, the fractal characteristics of the results generated by ProtoSphere pose challenges to its direct utilization in DEM-related granular simulations. Bonneau et al. [6] made an attempt to constrain the size of the particles generated by ProtoSphere by setting a range limit. However, they acknowledged explicitly that their method does not provide a means to regulate the particle size distribution of the outcomes. Therefore, our extended approach endeavors to enable control over the particle size distribution and address associated challenges.

4.1 PSD-Guided ProtoSphere

Modeling granular materials requires the determination of the relative proportion of particles of different sizes present in the material, which can be characterized by a piecewise constant distribution function $f(r)$ according to the following expression:

$$f(r) = \begin{cases} P_0 & \text{if } r_0 \le r < r_1 \\ P_1 & \text{if } r_1 \le r < x_2 \\ \vdots & \vdots \\ P_{n-1} & \text{if } r_{n-1} \le r < r_n \end{cases} \tag{3}$$

where the probability density function (PDF) of each particle size interval $[r_0, r_1), [r_1, r_2), ..., [r_{n-1}, r_n)$ is a constant value $P_0, P_1, ..., P_{n-1}$.

To achieve a controlled particle size distribution, it is essential to ensure that the radii of the particles generated by ProtoSphere align as closely as possible with the PDF $f(r)$. Assuming that a set of target radii that conform to the PDF $f(r)$ can be obtained beforehand, the objective is then turn to ensure that the radii of all the particles generated by ProtoSphere converge to their respective target radii. Regarding the pre-calculation of the target radii, the cumulative distribution function (CDF) $F(r)$ of the PDF $f(r)$ can be computed by $F(r) = \int_{-\infty}^{r} f(t)dt$ [13].

The fundamental concept behind our approach is to assign a target radius r_i' to each prototype $\boldsymbol{p}$ during the gridding process (in Algorithm 1 line 3), and then to iteratively adjust the radius of each prototype r_i until it converges to its target radius r_i'. We can modify Equation(2) as follows:

$$\boldsymbol{p} \leftarrow \boldsymbol{p} + \varepsilon(t) \cdot (r' - r)\frac{\boldsymbol{p} - \boldsymbol{q}_c}{\|\boldsymbol{p} - \boldsymbol{q}_c\|} \tag{4}$$

This modified formula indicates that if the target radius r' exceeds the current radius $r = \|\boldsymbol{p} - \boldsymbol{q}_c\|$, point $\boldsymbol{p}$ will move away from $\boldsymbol{q}_c$, otherwise it will move towards $\boldsymbol{q}_c$ if the current radius is greater than the target radius. For the cooling function $\varepsilon(t)$, we found that a time-based decay function performed better in our experiments. The function is shown as follows:

$$\varepsilon(t) = \frac{\varepsilon(t-1)}{1 + kt} \tag{5}$$

where k is a parameter that regulates the decay rate, and we set its value to 0.01 for all experiments.

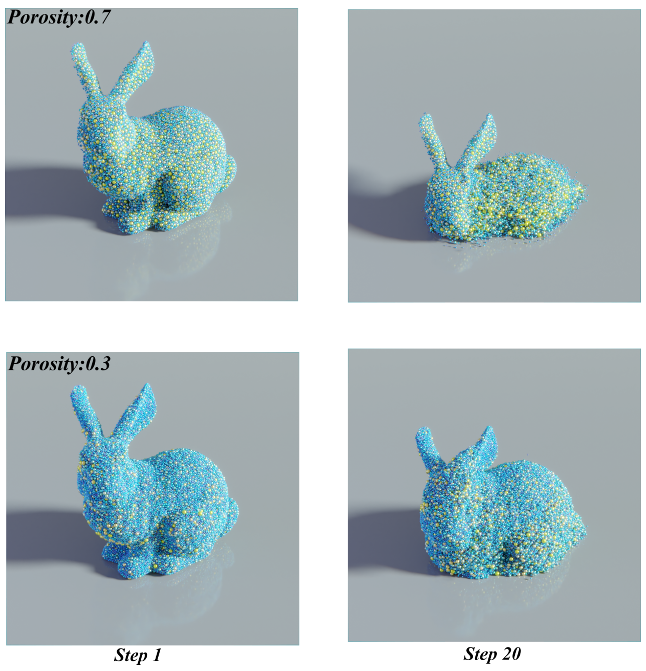

Fig. 5: Comparison experiment on the effects of multi-sized particle packing on capillary forces at two porosities

4.2 Randomly ODDF-Based Strategy

In Section 4.1, we introduce a modification to the prototype's motion, whereby its radius continuously converges towards the predetermined radius. Nevertheless, in contrast to the standard ProtoSphere algorithm, our approach yields relatively diminutive particle sizes in the initial stages of particle insertion. This results in the initial particles being inevitably placed at the boundary of the object, and subsequent inserted particles gradually expand from the boundary towards the center, which is an undesirable outcome. Specifically, this method of particle insertion gives rise to two main issues. Firstly, inserting particles in this way requires more steps, as each newly inserted particle can be seen as a new boundary of the object. Secondly, it results in a more homogeneous distribution of the inserted particles, as the particle size distribution of the newly inserted particles should be similar for each layer. Figure 2(b) illustrates the process of particle insertion using the 2D PSD-Guided ProtoSphere method, where it is evident that the process starts from the boundary and gradually progresses towards the center.

This issue arises due to the standard ProtoSphere algorithm's determination of the radius for each particle to be inserted, which is based on the distance between the current particle $\boldsymbol{p}$ and the nearest point $\boldsymbol{q}_c$ on the boundary. More specifically, the standard ProtoSphere algorithm neglects the consideration of particle radius sizes, leading to the insertion of several large-radius particles during the initial stages of particle insertion, forming the internal skeleton of the geometry. While our modified ProtoSphere method (Equation(4)) is designed to regulate the size of the particle radius, and in the majority of cases, it is unnecessary to generate particles with such a large radius. Consequently, the process of gradually inserting particles layer by layer from the boundary towards the center, as illustrated in Fig-

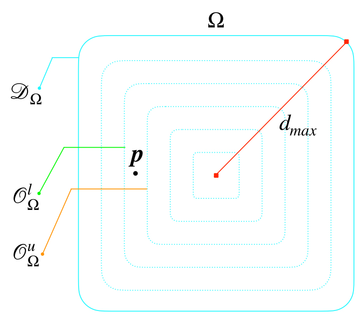

Fig. 6: Illustration of the randomly offset discrete distance field-based strategy for particle packing

ure 2(b), is observed. To address the issue of undesired particle insertion, our proposed solution involves assigning a unique discrete distance field to each particle. This allows for the recalculation of both the displacement direction $\boldsymbol{p}-\boldsymbol{q}_c$ and the particle radius r. We achieve this by introducing a random variable $\rho \in [0.04, 0.2]$ to each prototype. Then we divide the discrete distance field $\mathcal{D}_\Omega$ into multiple subfields based on the maximum distance $d_{\max}$ in the field. As a result, we obtain multiple distance fields with varying offset levels, as shown by the dashed lines in Figure 6. Based on the position of current prototype $\boldsymbol{p}$ in the distance field, we locate the two offset distance fields, $\mathcal{O}^u_\Omega$ and $\mathcal{O}^l_\Omega$ that are closest to it. At last, the updated distance field $\mathcal{D}'_\Omega$ can be computed by utilizing the pre-computed distance field $\mathcal{D}_\Omega$, the offset distance fields $\mathcal{O}^u_\Omega$ and $\mathcal{O}^l_\Omega$, as follows:

$$\begin{aligned}
\mathcal{D}'_\Omega(\boldsymbol{p}) &= \min(\mathcal{D}_\Omega(\boldsymbol{p}), \min(\mathcal{O}^u_\Omega(\boldsymbol{p}), \mathcal{O}^l_\Omega(\boldsymbol{p}))) \\
\mathcal{O}^u_\Omega(\boldsymbol{p}) &= \left\lceil \frac{\mathcal{D}_\Omega(\boldsymbol{p})}{\rho d_{\max}} \right\rceil \rho d_{\max} - \mathcal{D}_{\Omega(\boldsymbol{p})} \\
\mathcal{O}^l_\Omega(\boldsymbol{p}) &= \mathcal{D}_\Omega(\boldsymbol{p}) - \left\lfloor \frac{\mathcal{D}_\Omega(\boldsymbol{p})}{\rho d_{\max}} \right\rfloor \rho d_{\max}
\end{aligned} \tag{6}$$

As illustrated in Figure 2(c), our proposed strategy can insert particles of the target size at unpredictable locations in the space during the initial insertion phase. This is in contrast to Figure 2(b), where particle insertion is constrained to the boundaries only. Moreover, we evaluated the performance of the XProtoSphere algorithm with and without the randomly ODDF strategy, and obtained promising results. As shown in Figure 3(a), the algorithm with the ODDF strategy is capable of inserting more particles at each iteration, resulting in a higher total number of inserted particles compared to the non-ODDF strategy. To further evaluate the algorithm's efficiency, we measured the computation time for the ODDF and non-ODDF strategies, as depicted in Figure 3(b). The results demonstrate that the ODDF strategy consistently outperforms the non-ODDF strategy in generating the same number of particles. This suggests that although the ODDF strategy requires more computational resources, it is more efficient for particle insertion.

4.3 Coupling with Discrete Element Relaxation

In the standard ProtoSphere algorithm, the porous regions within the space can be continuously explored to identify and fill them with particles of appropriate sizes. However, when there is a limit on the particle size that we need to insert, it becomes challenging to maximize the packing density of the entire region and minimize its porosity. It is essential to acknowledge that the inability to further reduce the spatial porosity is mainly attributed to the suboptimal distribution of the already inserted particles, which limits our capability of further particle insertion. If we induce particle movements, then more space can be created during the motion, thereby allowing for the insertion of additional particles. The ProtoSphere algorithm detects particle overlaps during the insertion of new particles and subsequently removes them. However, our proposed approach involves relaxing the strict constraints on particle overlap during the

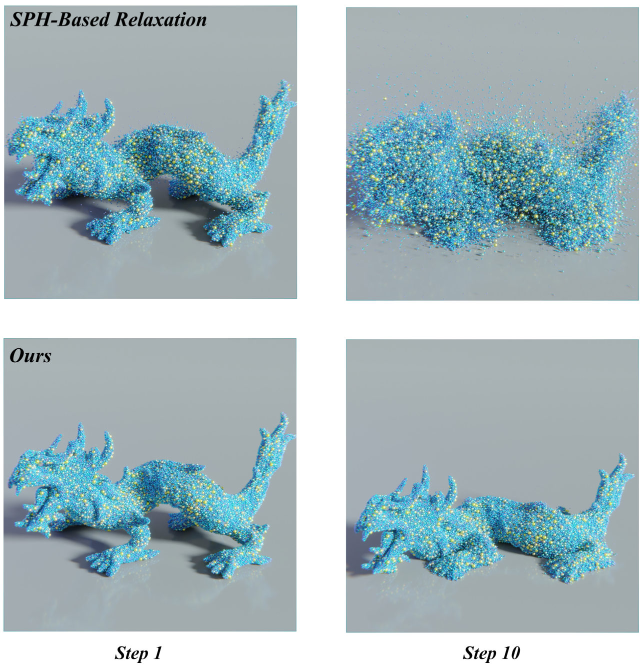

Fig. 7: Comparison between multi-sized particles packing using the SPH-based relaxation and our method when applied to DEM

Algorithm 2 XProtoSphere Coupling with Discrete Element Relaxation

```
Input: surface $\Omega$ of object $O$, a probability density function
    $f(r)$, maximum relaxation steps $N$
Output: a group of particles with radius information
 1: insertion number $\ell \leftarrow 1$
 2: repeat
 3:     Using XProtoSphere($\Omega$,$f(r)$) to insert particles once
 4:     for relaxation steps $i = 0$;$i < N$;$i = i + 1$ do
 5:         elastic force $\boldsymbol{F}^n_k$ between the particles can be computed by employing Equation(7)
 6:         compute the acceleration $\boldsymbol{a}$ of particles and update it using the energy decay mechanism outlined in Equation(8)
 7:         use Equation(9) corrects the position and velocity of particles near the boundary
 8:         advance the particle
 9:     end for
10: until $\ell >$ maximum insertion number
```

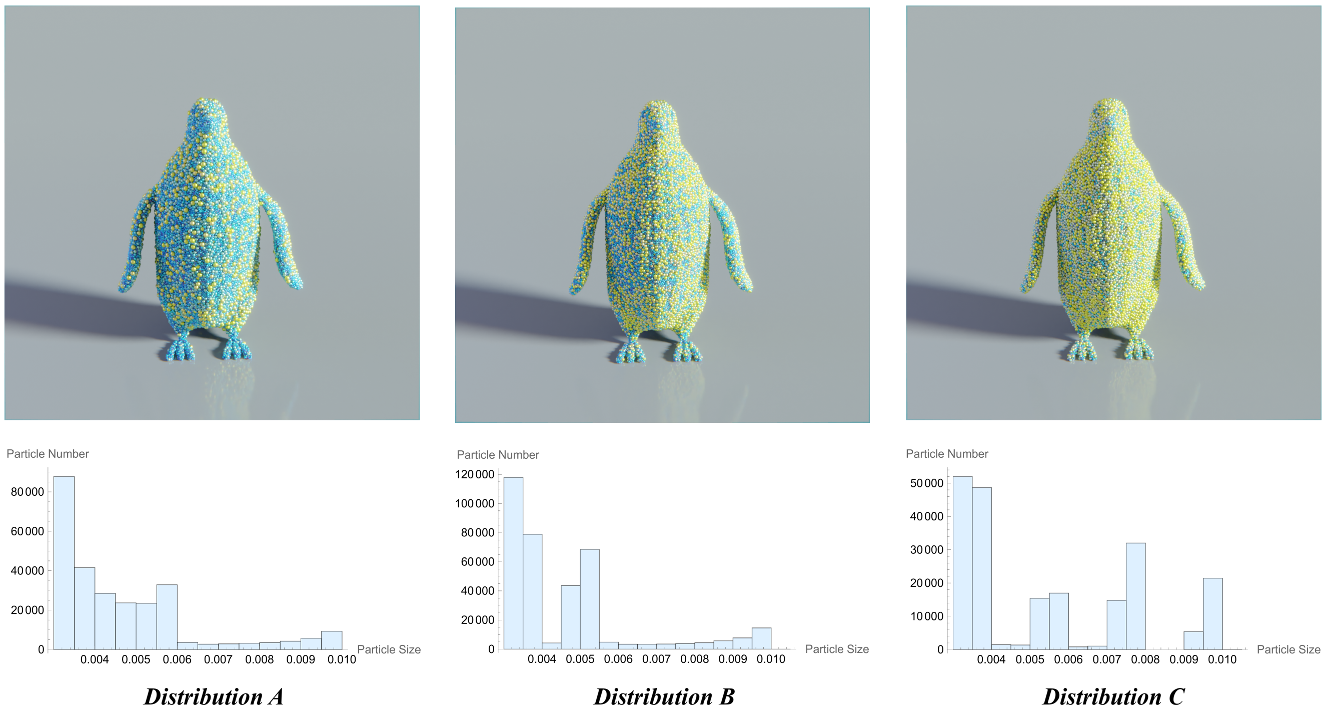

Fig. 8: Comparison experiment using XProtoSphere for multi-sized particle packing under different pre-defined particle size distributions

insertion phase. Subsequently, we utilize the DEM to iteratively reduce the overlap between particles.

The DEM is a widely used Lagrangian-based simulation model for granular media, commonly employed in the field of soil mechanics [11]. In the DEM system, each particle is treated as a rigid body with a defined position and radius. When the distance between two particles, denoted as $d = \|\boldsymbol{x}_{k_1} - \boldsymbol{x}_{k_2}\|$, is less than the sum of their radii (i.e., $r_{k_1} + r_{k_2}$), they are considered to be in contact or collide with each other. At this point, normal and tangential forces, denoted as $\boldsymbol{F}_k^n$ and $\boldsymbol{F}_k^t$, respectively, are exerted on the particles to simulate their interactions [16]. Our proposed approach focuses on utilizing only the elastic force $\boldsymbol{F}_k^n$ in the normal direction to separate overlapping particles, define as:

$$\begin{aligned} \boldsymbol{F}_k^n &= K_N(r_{k_1} + r_{k_2} - d)\hat{\boldsymbol{x}}_{k_1k_2} \\ K_N &= \frac{2Y_1R_1Y_2R_2}{Y_1R_1 + Y_2R_2} \\ \hat{\boldsymbol{x}}_{k_1k_2} &= \frac{\boldsymbol{x}_{k_1} - \boldsymbol{x}_{k_2}}{d} \end{aligned} \tag{7}$$

where K_N represents the normal stiffness coefficient, which is related to Young's modulus Y and particle radius R.

To prevent continuous fluctuations caused by particles getting trapped in narrow gaps, we employ an energy decay mechanism [27] as follow:

$$\boldsymbol{a} \leftarrow \boldsymbol{a}\left(1 - \lambda \mathrm{sgn}\left(\boldsymbol{a}\left(\boldsymbol{v} + \frac{\boldsymbol{a}\Delta t}{2}\right)\right)\right) \tag{8}$$

where the parameter λ represents the damping coefficient, $\boldsymbol{a}$ denotes acceleration, $\boldsymbol{v}$ represents velocity, and Δt is the time step. Additionally, the sgn function is utilized with possible values of -1, 0, or 1.

It is important to note that when a particle begins to move due to elastic forces, it must be constrained to remain within the object. In particular, if the particle makes contact with the boundary, a repulsive force must be applied to prevent it from crossing the boundary. In our approach, if a particle $\boldsymbol{p}$ exceeds the boundary Ω, we do not apply a repulsive force. Rather, we modify its position and adjust its velocity to ensure that it remains inside the object O. Specifically, we use position modification to move the particle back inside the boundary, and velocity adjustment to match the post-repulsive velocity, as shown below:

$$\begin{aligned} &\boldsymbol{x} \leftarrow \boldsymbol{x} - \mathcal{D}_\Omega(\boldsymbol{x})\nabla\hat{\mathcal{D}}_\Omega(\boldsymbol{x}) \\ &\boldsymbol{v} \leftarrow e(\boldsymbol{v} - 2(\boldsymbol{v} \cdot \hat{\mathcal{D}}_\Omega(\boldsymbol{x}))\hat{\mathcal{D}}_\Omega(\boldsymbol{x})) \end{aligned} \tag{9}$$

The equation involves the repulsive coefficient represented by the parameter e, and the normalized gradient of the discrete distance field $\mathcal{D}_\Omega$ at position $\boldsymbol{x}$, denoted by $\hat{\mathcal{D}}_\Omega(\boldsymbol{x})$. Note that the formula presented above is executed only when the distance $\mathcal{D}_\Omega(\boldsymbol{x})$ is less than or equal to 0. The complete particle motion was recorded during the application of the Discrete Element Relaxation, as shown in Figure 4. The bunny-shaped multi-sized spheres in Step 1 were generated using the

XProtoSphere, with a minor modification to the insertion process. Specifically, the insertion condition in line 18 of Algorithm 3 was altered to permit the insertion of spheres with an overlapping rate of ϵ and ξ or less, rather than those without any overlapping spheres. This is due to the fact that in the initial stages of the process, if the particles do not overlap, the Discrete Element Relaxation is unable to generate the necessary forces to move the particle swarm and adjust particle positions to optimize space filling. It is important to note that when coupled with the discrete element relaxation method, a small overlap between particles may still occur. Detailed experimental results on the spatial porosity and overlapping rate obtained by coupling XProtoSphere with discrete element relaxation are presented in Table 1.

4.4 Implementation Details

Algorithm 3 outlines the specific implementation details for XProtoSphere, highlighting the differences from the standard ProtoSphere algorithm in red. Compared to ProtoSphere, XProtoSphere necessitates a pre-defined probability density function $f(r)$ to set the target particle size distribution as input data. Regarding the randomly ODDF-strategy discussed in Section 4.2, it is worth noting that it is only necessary to apply this strategy during the initial particle insertion process (line 7). This is due to the fact that particles can already be inserted at non-boundary locations of the object O in the first operation, and the resulting inserted particles will subsequently be utilized as new boundary information in subsequent computations. During the particle insertion process, we apply a stricter constraint as indicated in line 18 of our algorithm.

In order to minimize the discrepancy between the target particle size and the actual size following insertion, we also require that $|\frac{r_k}{r'_k}-1|$ not exceed the threshold $\epsilon = 0.01$. Typically, an acceptable particle overlapping rate (ξ which is described in Section 4.3) of 0.4 to 0.6 is employed, as this allows for minimal particle overlapping following implementation of the Discrete Element Relaxation. Algorithm 2 details the process of coupling XProtoSphere with Discrete Element Relaxation to further enhance the packing density. Upon the insertion of each particle by means of XProtoSphere, the iteration process is performed $N = 1000$ times using Discrete Element Relaxation.

Algorithm 3 XProtoSphere Algorithm

```
Input: surface $\Omega$ of object $O$, a probability density function
    $f(r)$
Output: a group of particles with radius information
 1: insertion number $\ell \leftarrow 1$
 2: $\mathcal{D}_\Omega \leftarrow$ initialize the discrete distance field
 3: repeat
 4:     $S : \{\boldsymbol{p}_1, \boldsymbol{p}_2, \cdots, \boldsymbol{p}_n\} \leftarrow$ place prototype $\boldsymbol{p}_i$ randomly
        inside grid $c_i$
 5:     for each prototype $\boldsymbol{p}_i$, a target radius $r'_i$ is generated
        independently based on $f(r)$
 6:     if $\ell == 1$ then
 7:         use Equation(6) to compute $\mathcal{D}'_\Omega$ as discrete distance
            field
 8:     else
 8:         update discrete distance field $\mathcal{D}_\Omega$ by $\Omega \leftarrow \Omega \cup \Omega_{\boldsymbol{p}_k}$
 9:     end if
10:     for each $\boldsymbol{p}_i$ in $S$ do
11:         repeat
12:             $\boldsymbol{q}_c = \arg\min\{\|\boldsymbol{p}_i - \boldsymbol{q}\| : \boldsymbol{q} \in \Omega\}$
13:             $\boldsymbol{p}_i \leftarrow \boldsymbol{p}_i + \varepsilon(t) \cdot (r'_i - r_i)\frac{\boldsymbol{p}_i - \boldsymbol{q}_c}{\|\boldsymbol{p}_i - \boldsymbol{q}_c\|}$
14:             $r_i = \|\boldsymbol{p}_i - \boldsymbol{q}_c\|$
15:         until $\boldsymbol{p}_i$ has converged
16:     end for
17:     sort $P$ by max radius $r_i$
18:     find $\boldsymbol{p}_k \in P$ that are not overlapped by any $\boldsymbol{p}_i$ and
        $|\frac{r_k}{r'_k} - 1| < \epsilon$
19:     insert particles at positions $\boldsymbol{p}_k$ with radii $r_k$
20:     $\ell = \ell + 1$
21: until $\ell >$ maximum insertion number
```

5 Experimental Results

This section presents the results obtained by our algorithm for sphere packing of various geometries, along with its application in physically-based simulation algorithms. We also evaluated the performance of the XProtoSphere when applied to different geometries, as shown in Table 1. A, B, C meaning the particle size distributions shown in Figure 8.

The computational processes defined by lines 6 to 17 in Algorithm 3 and lines 4 to 8 in Algorithm 2 are executed on an NVIDIA GeForce RTX3090 24GB GPU by using CUDA programming framework. The pre-calculation of the discrete distance field and the computation of the particle insertion step are executed on a AMD Ryzen 9 3900X CPU.

Application to Capillary Model To accurately simulate sandy materials with clay-like behavior, achieving a higher packing density of multi-sized particles is crucial to enhance the realism of the simulation and distinguish the loose nature of the sand. In this study, we employed multi-sized particles with high porosity (low packing density) and low porosity (high packing density) to model sand with clay properties, as depicted in Figure 5. By applying the capillary force model proposed by Wang et al. [31], Our findings suggest that low-porosity particles exhibit a greater propensity for clay-like behavior compared to high-porosity particles. This finding shows that the proposed method is useful for reproducing sediment structures.

Table 1: Comparing XProtoSphere performance across different geometries

Model	Number			Time(s)			Overlap($\times 10^{-6}$)			Porosity(%)			BHD($\times 10^{-3}$)			JSD($\times 10^{-3}$)			KLD($\times 10^{-2}$)		
	A	B	C	A	B	C	A	B	C	A	B	C	A	B	C	A	B	C	A	B	C
Armadillo	243k	323k	174k	328	431	231	7.87	15.1	13.4	30	33.2	37.8	2.4	4.3	24	2.3	4.28	23.6	0.8	1.71	9.6
Beast	292k	401k	217k	410.1	592.56	315.53	8.63	23.39	12.18	29	32.8	35.7	3.8	5.5	24.13	3.7	5.5	23.7	1.27	2.2	9.6
Bunny	282k	387k	206k	384.53	571.01	278.3	8.519	10.7	12.4	31	35.2	38	3.5	6.12	27.9	3.43	6.09	27.44	1.17	2.4	11.1
Cheburashka	229k	324k	184k	256.249	367.13	203.9	8.1	11.6	11.3	28.7	34.3	34	6.34	10	26.6	6	9.9	26.1	2.06	4.04	10.6
Cow	303k	418k	226k	389.06	577.68	285.25	8.707	15.5	11.8	29.9	34.4	36.1	4.13	6.7	26.2	3.99	6.7	25.8	1.36	2.7	10.4
Dolphin	293k	415k	230k	363.56	555.11	280.4	7.888	32.1	10.5	28.8	33.8	34.7	5.66	7.5	26.3	5.44	7.4	25.9	1.84	3	10.5
Dragon	202k	274k	156k	221.17	286.7	180.47	11.44	15.5	33.1	28.9	35	33.5	5.9	15.6	27.1	5.7	15.4	26.6	1.952	6.3	10.8
Homer	221k	309k	174k	246.33	351.2	189	7.267	11.5	10.8	28.2	32.9	34.1	5.49	7.1	24.5	5.3	7	24.1	1.8	2.8	9.7
Horse	270k	371k	206k	353.29	508	273.3	8.508	11.9	12.7	30.5	34.6	36.2	4.4	7.1	26.1	4.3	7.1	25.7	1.46	2.8	10.4
Lucy	224k	306k	166k	400.41	349.95	200	10.25	11.8	13.5	29.3	34.5	35.2	4.67	10.2	26.3	4.5	10.2	25.8	1.53	4.1	10.4
Nefertiti	326k	443k	239k	502.9	740	373.75	16.2	92.9	12.6	31.3	33.7	38	2.94	4.1	24.9	2.8	4.1	24.5	0.99	1.6	10
Penguin	264k	364k	211k	299.99	417	232.5	8.027	12	10	28.9	34.5	33.4	6.2	11.6	28.6	5.9	11.5	28	2.02	4.7	11.5
Spot	292k	416k	232k	370.166	547.9	284.98	8.003	12.4	10	29.3	34.8	34.9	5.95	9	27.8	5.7	9.1	27.3	1.94	3.7	11.1
Suzanne	311k	448k	262k	404.163	606	337	7.97	13.7	10.9	33.6	39	37.8	6.8	11.3	33.5	6.5	11.3	32.8	2.23	4.6	13.4
Turtle	236k	347k	223k	267.365	386.6	260.46	60.77	12.2	152	25.57	33.6	29.6	13.6	18.4	30.8	12.9	18.1	30	4.39	7.3	12.9
Average	**266k**	**370k**	**207k**	**346.485**	**485.856**	**261.722**	**12.543**	**20.152**	**22.478**	**29.531**	**34.42**	**35.266**	**5.452**	**8.968**	**26.982**	**5.23**	**8.911**	**26.489**	**1.787**	**3.596**	**10.8**

Comparison with SPH-Based Relaxation To enhance packing density, a comparison experiment between the SPH-based relaxation [21] and our method was conducted using the DEM. Results from Figure 7 revealed that the SPH-based relaxation exhibited considerable instability during the early stages of simulation, which is likely due to high particle overlap resulting from the algorithm's application to irregularly distributed multi-sized particles. In contrast, our proposed method demonstrates significantly more stable simulations, and increases packing density with minimal particle overlap, as shown in Table 1. Furthermore, we performed a quantitative analysis to assess the stability of our method by comparing the total energy, as shown in Figure 9. The blue line in the figure depicts the total energy variation of the DEM simulation with non-overlapping particles in the initial stage. Its variation trend is highly similar to that of our proposed method (green line) during the DEM simulation. However, the SPH Relaxation-based method produces abnormally high total energy values during the early stages of the simulation. The unstable simulation results, as shown in Figure 7, provide further evidence of this issue. As outlined in Section 4.3, our proposed method for further enhancing the particle packing density cannot completely eliminate particle overlap, leading to a slightly higher total energy in the DEM simulation compared to the case without particle overlap. Nevertheless, the average overlap rate between particles packed by our method is significantly below 1%, whereas that of the SPH-based method is approximately 30%. This discrepancy is also why our method can produce stable DEM simulations, while the SPH-based method cannot.

Comparison Experiments on Various Predefined Particle Size Distributions To illustrate the versatility of our method, we conducted experiments using three distinct representative particle size distributions, as shown in Figure 8. We present visualizations of the resulting multi-sized particle packing and corresponding particle distributions. For the quantitative analysis results of these three sets of experiments, please refer to the data row labeled 'Penguin' in Table 1.

Application to Discrete Element Method Herein, we present the results of multi-sized particle packing experiments on 15 commonly used 3D models to demonstrate the applicability and versatility of our algorithm in handling objects of arbitrary geometry. The obtained packed particles are shown to be stable when applied to DEM-related algorithms for sand-like simulation (see Figure 10). In Table 1, we provide a comprehensive record of the results of our multi-sized particle packing experiments for the 15 commonly used 3D models, including the total number of particles, the computational time, the average overlapping rate, and the porosity. Moreover, we employed three metrics, Bhattacharyya Distance (BHD), Jensen-Shannon Divergence (JSD), and Kullback-Leibler Divergence (KLD), respectively, to quantitatively evaluate the difference between the particle size distribution obtained by our algorithm and the pre-defined particle size distribution. Specifically, these three indicators can serve as measures to assess the similarity between two distributions. A value closer to 0 indicates a higher level of similarity between the two distributions, and a value of 0 suggests that the distributions are identical. As shown by the results in Table 1 from the experiments conducted using three different predefined distributions, the indicators used to evaluate the similarity of the distributions range from approximately 0.1 to 0.001. These results highlight the effectiveness of our algorithm in generating particle distributions that closely resemble the pre-defined distributions.

6 Conclusion

This study presents the XProtoSphere method, a multi-sized particle packing algorithm that can conform to a pre-defined distribution by extending the ProtoSphere approach. Specifically, the radius of each prototype is gradually adjusted to match its target radius. Additionally, we propose a randomly ODDF-based strategy to

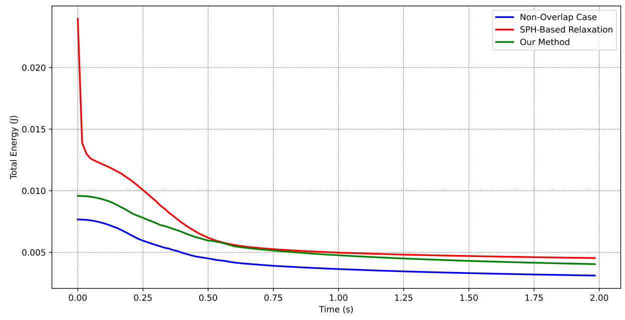

Fig. 9: Quantitative assessment of the stability of particle relaxation methods based on SPH and DEM through comparative total energy analyses

enhance the initial particle insertion phase and enable the packing of a larger number of particles. Finally, we integrate the XProtoSphere with the Discrete Element Relaxation method to further increase packing density and minimize particle overlap, which are essential for stable granular material simulations.

The major limitation of our algorithm that we need to remove overlapping particles and insert newly generated particles without overlap when all prototypes converge. It presents two challenges: firstly, it is difficult to parallelize this process on the GPU, as it may consume significant computational resources. Secondly, the removal of overlapping particles can result in a large discrepancy between the final packed particle distribution and the pre-defined particle distribution. To address these issues, we would like to propose a fully GPU-based XProtoSphere algorithm in the further research, which can control particle size distribution more effectively by avoiding the need for particle deletion. In addition to computational performance-related issues, while our proposed method allows for the regulation of the particle radius distribution, it does not have the capability to regulate the spatial distribution of particles with varying sizes. One potential solution involves utilizing the medial axis-based local geometric feature function introduced in Adams et al. [1] to regulate the spatial distribution of particles. However, this approach presents a challenge in simultaneously regulating the distribution of particle sizes. Therefore, in future work, we would like to propose a method that can effectively control both the particle size distribution and the spatial distribution of multi-sized particles. By doing so, we can extend the applicability of the XProtoSphere method to a wider range of scenarios.

Conflict of Interest Statement

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

Data Availability Statement

Data will be made available on reasonable request.

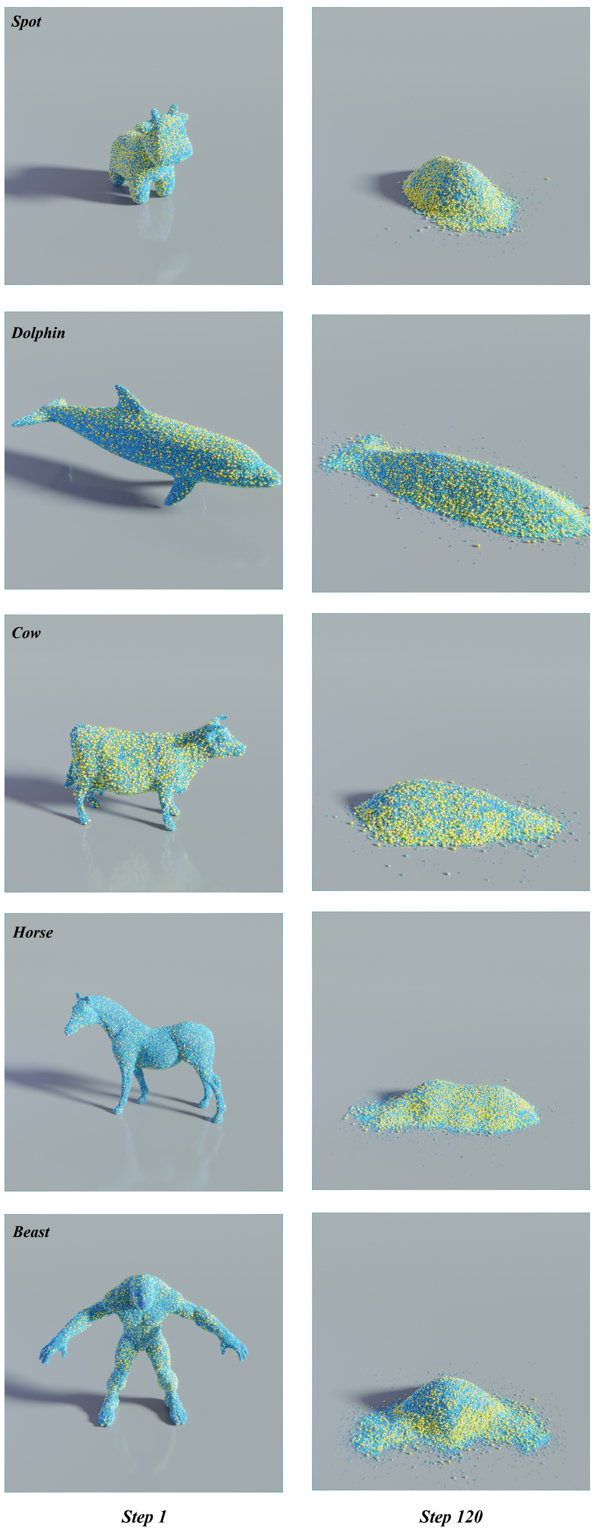

Fig. 10: Additional results generated by the XProtoSphere and their effects after importation into the sand simulator

References

1. Adams, B., Pauly, M., Keiser, R., Guibas, L.J.: Adaptively sampled particle fluids. ACM Transactions on Graphics (TOG) **26**(3), 48–es (2007)
2. Aurenhammer, F.: Power diagrams: properties, algorithms and applications. SIAM Journal on Computing **16**(1), 78–96 (1987)
3. Baran, I., Popović, J.: Automatic rigging and animation of 3d characters. ACM Transactions on Graphics (TOG) **26**(3), 72–es (2007)
4. Barber, C.B., Dobkin, D.P., Huhdanpaa, H.: The quickhull algorithm for convex hulls. ACM Transactions on Mathematical Software (TOMS) **22**(4), 469–483 (1996)
5. Bell, N., Yu, Y., Mucha, P.J.: Particle-based simulation of granular materials. In: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation, pp. 77–86 (2005)
6. Bonneau, F., Scholtes, L., Rambure, H.: An algorithm for generating mechanically sound sphere packings in geological models. Computational Particle Mechanics **8**(2), 201–214 (2021)
7. Borkovec, M., De Paris, W., Peikert, R.: The fractal dimension of the apollonian sphere packing. Fractals **2**(04), 521–526 (1994)
8. Bridson, R.: Fast poisson disk sampling in arbitrary dimensions. In: ACM SIGGRAPH 2007 Sketches, pp. 22–es (2007)
9. Conway, J.H., Sloane, N.J.A.: Sphere packings, lattices and groups, vol. 290. Springer Science & Business Media (2013)
10. Cui, L., O'Sullivan, C.: Analysis of a triangulation based approach for specimen generation for discrete element simulations. Granular Matter **5**, 135–145 (2003)
11. Cundall, P.A., Strack, O.D.: A discrete numerical model for granular assemblies. geotechnique **29**(1), 47–65 (1979)
12. Desbrun, M., Cani, M.P.: Space-time adaptive simulation of highly deformable substances. Ph.D. thesis, INRIA (1999)
13. Devroye, L.: Sample-based non-uniform random variate generation. In: Proceedings of the 18th conference on Winter simulation, pp. 260–265 (1986)
14. George, P.L., Borouchaki, H.: Delaunay triangulation and meshing : application to finite elements. Hermès (1998)
15. Hales, T.C.: A proof of the kepler conjecture. Annals of mathematics pp. 1065–1185 (2005)
16. Hentz, S., Donzé, F.V., Daudeville, L.: Discrete element modelling of concrete submitted to dynamic loading at high strain rates. Computers & structures **82**(29-30), 2509–2524 (2004)
17. Hifi, M., M'hallah, R.: A literature review on circle and sphere packing problems: models and methodologies. Advances in Operations Research **2009** (2009)
18. Jerier, J.F., Imbault, D., Donze, F.V., Doremus, P.: A geometric algorithm based on tetrahedral meshes to generate a dense polydisperse sphere packing. Granular Matter **11**, 43–52 (2009)
19. Jerier, J.F., Richefeu, V., Imbault, D., Donzé, F.V.: Packing spherical discrete elements for large scale simulations. Computer Methods in Applied Mechanics and Engineering **199**(25-28), 1668–1676 (2010)
20. Jiang, M., Southern, R., Zhang, J.J.: Energy-based dissolution simulation using sph sampling. Computer Animation and Virtual Worlds **29**(2), e1798:1–20 (2018)
21. Jiang, M., Zhou, Y., Wang, R., Southern, R., Zhang, J.J.: Blue noise sampling using an sph-based method. ACM Transactions on Graphics (TOG) **34**(6), 211:1–11 (2015)
22. Kita, N., Miyata, K.: Multi-class anisotropic blue noise sampling for discrete element pattern generation. The Visual Computer **32**, 1035–1044 (2016)
23. Lopes, L.G., Cintra, D.T., Lira, W.W.: A geometric separation method for non-uniform disk packing with prescribed filling ratio and size distribution. Computational Particle Mechanics **8**, 169–182 (2021)
24. Lopes, L.G., Cintra, D.T., Lira, W.W.: A particle packing parallel geometric method using gpu. Computational Particle Mechanics **8**, 931–942 (2021)
25. Schechter, H., Bridson, R.: Ghost sph for animating water. ACM Transactions on Graphics (TOG) **31**(4), 61:1–8 (2012)
26. Schwarz, M., Seidel, H.P.: Fast parallel surface and solid voxelization on gpus. ACM Transactions on Graphics (TOG) **29**(6), 179:1–10 (2010)
27. Šmilauer, V., Chareyre, B.: Yade dem formulation. http://yade-dem.org/doc/formulation.html
28. Teuber, J., Weller, R., Zachmann, G., Guthe, S.: Fast sphere packings with adaptive grids on the gpu. In: GI AR/VRWorkshop, 12 pages (2013)
29. Torquato, S., Haslach Jr, H.: Random heterogeneous materials: microstructure and macroscopic properties. Appl. Mech. Rev. **55**(4), B62–B63 (2002)
30. Tsuzuki, S., Aoki, T.: Large-scale granular simulations using dynamic load balance on a gpu supercomputer. In: Poster at the 26th IEEE/ACM International Conference on High Performance Computing, Networking, Storage and Analysis (2014)
31. Wang, X., Fujisawa, M., Mikawa, M.: Visual simulation of soil-structure destruction with seepage flows. Proceedings of the ACM on Computer Graphics and Interactive Techniques **4**(3), 41:1–18 (2021)
32. Wang, X., Fujisawa, M., Mikawa, M.: Multi-sized particle sampling method based on porosity optimization in 2d space. IIEEJ Transactions on Image Electronics and Visual Computing **10**(2), 150–161 (2022)
33. Weller, R., Zachmann, G.: A unified approach for physically-based simulations and haptic rendering. In: Proceedings of the 2009 ACM SIGGRAPH Symposium on Video Games, pp. 151–159 (2009)
34. Weller, R., Zachmann, G.: Protosphere: A gpu-assisted prototype guided sphere packing algorithm for arbitrary objects. In: ACM SIGGRAPH ASIA 2010 Sketches, pp. 8:1–2 (2010)
35. Winchenbach, R., Hochstetter, H., Kolb, A.: Infinite continuous adaptivity for incompressible sph. ACM Transactions on Graphics (TOG) **36**(4), 102:1–10 (2017)
36. Winchenbach, R., Kolb, A.: Optimized refinement for spatially adaptive sph. ACM Transactions on Graphics (TOG) **40**(1), 8:1–15 (2021)
37. Wong, K.M., Wong, T.T.: Blue noise sampling using an n-body simulation-based method. The Visual Computer **33**, 823–832 (2017)
38. Yan, D.M., Guo, J.W., Wang, B., Zhang, X.P., Wonka, P.: A survey of blue-noise sampling and its applications. Journal of Computer Science and Technology **30**(3), 439–452 (2015)
39. Zhang, K., Liu, F., Zhao, G., Xia, K.: Fast and efficient particle packing algorithms based on triangular mesh. Powder Technology **366**, 448–459 (2020)
40. Zheng, X., Si, J., Dai, S.: Blue noise sampling with a pbf-based method. In: Proceedings of the 33rd Computer Graphics International, pp. 77–80 (2016)