electronics

MDPI

Article

Material Point Method-Based Simulation Techniques for Medical Applications

Su-Kyung Sung, Jae-Hyeong Kim and Byeong-Seok Shin *

Department of Electrical and Computer Engineering, Inha University, Incheon 22212, Republic of Korea; rebirth87@naver.com (S.-K.S.); wogudkim@gmail.com (J.-H.K.)
* Correspondence: bsshin@inha.ac.kr

Abstract: We propose a method for recognizing fragment objects to model the detailed tearing of elastic objects like human organs. Traditional methods require high-performance GPUs for real-time calculations to accurately simulate the detailed fragmentation of rapidly deforming objects or create random fragments to improve visual effects with minimal computation. The proposed method utilizes a deep neural network (DNN) to produce physically accurate results without requiring high-performance GPUs. Physically parameterized material point method (MPM) simulation data were used to learn small-scale detailed fragments. The tearing process is segmented and learned based on various training data from different spaces and external forces. The inference algorithm classifies the fragments from the training data and modifies the deformation gradient using a modifier. An experiment was conducted to compare the proposed method and the traditional MPM in the same environment. As a result, it was confirmed that visual fidelity for the tearing of elastic objects has been improved. This supports the simulation of various incision types in a virtual surgery.

Keywords: elastic object; virtual surgery; material point method; neural network

Citation: Sung, S.-K.; Kim, J.-H.; Shin, B.-S. Material Point Method-Based Simulation Techniques for Medical Applications. *Electronics* **2024**, *13*, 1340. https://doi.org/10.3390/electronics13071340

Academic Editors: Hamido Fujita, Tun-Wen Pai and Andres Hernandez-Matamoros

Received: 1 March 2024
Revised: 1 April 2024
Accepted: 1 April 2024
Published: 2 April 2024

1. Introduction

Virtual reality (VR) technology is emerging as a solution for training medical professionals and providing alternative psychological treatments. Virtual surgery, one of the VR medical technologies, is widely employed as an auxiliary tool for doctors, a learning aid for medical students, and a training or simulation tool for healthcare professionals. To deliver virtual surgery experiences that are as realistic as possible, various interdisciplinary technologies are utilized to accurately depict surgical environments and their intricacies. Creating a realistic surgical training experience necessitates providing tactile feedback, resistance, viscosity, and other physical properties of the virtual body. Numerous methods have been studied to realistically simulate the tearing or deformation processes of complex elastic objects, including virtual bodies.

To express the realistic deformation and tearing of an elastic object, the mass–spring method, the linear elasticity-based FEM, and the MPM method were used. The mass–spring system is one approach to implementing a continuous physical model using discrete points with mass and virtual springs connecting them [1]. This method can accurately and quickly represent continuous models unless precise scientific calculations are required. Mass–spring systems have been employed to simulate one-dimensional or two-dimensional structures, such as hair [2], cloth [3], and rigid bodies [4], with a limited degree of elasticity. However, the mass–spring system introduces artificial anisotropy depending on mesh selection, making it challenging to simulate soft tissue deformation properties accurately. Additionally, relating spring stability to material properties like Young's modulus is challenging. The linear elasticity-based Finite Element Method (FEM) is a simulation technique for deforming objects based on the assumption of small displacements [5]. While this method assumes linear deformation, real-world objects become nonlinear as deformations

increase. Therefore, when large deformations occur or nonlinear transformations like rotation are applied, this approach yields inaccurate results. To address this, a method was proposed to modify the precomputed stiffness matrix encompassing an object's geometric and physical specifications rather than recalculating the entire matrix [6]. This approach only updates matrices affecting the object's deformation, improving efficiency. Felippa introduced the co-rotational method to enhance deformation results based on the second issue of linear elasticity-based methods, which assumes small displacements [7]. This method separates object displacements into rigid body and strained parts, calculating internal forces using only the strained parts for a more accurate computation of nonlinear variations, such as rotation. Another method for real-time deformable model representation is using FEM to compute variations based on an explicit integration scheme [8]. The advantage of this approach is its reliance on mass matrices for deformation calculations. Mass lumping can simplify mass matrices into diagonal matrices [9]. Consequently, each degree of freedom can be independently solved by decomposing the equations of motion, allowing for an intuitive parallelization and faster equation solving [10]. The explicit integration method is suitable for real-time simulations involving complex movements, such as brain deformations, as it can artificially increase mass to simulate different materials [11]. MPM is a hybrid Lagrangian–Eulerian discretization technique for solid mechanics and a generalization of the FLIP [12] method, and it is commonly used in fluid animation [13]. MPM has found applications in various computer graphics domains, including snow [14], sand [15,16], foam [17,18], fabrics [19], and solid–fluid mixtures [20,21]. In particular, Daviet et al. [15] proposed a semi-implicit frictional boundary condition to combine MPM with rigid bodies. However, separating continuous materials in MPM remains challenging when using a single deformation gradient field. Wretborn et al. [22] animated MPM crack propagation [23] by incorporating multiple grids and assuming pre-existing cracks and pure elastic materials. In the engineering literature, material heterogeneity in MPM is sometimes achieved through explicit front tracking [24], which involves multiple ray-crossing tests per particle using an explicit mesh. Other approaches use deformation regularization or material damage [25], similarly to element removal in FEM [26], which can result in mesh-dependent volume loss and unwanted fragmentation. Gao et al. [27] developed a spatially adaptive MPM, resolving thin features by locally refining the computational grid and particles. Moutsanidis et al. [28] modeled strong heterogeneity via the MPM using a single deformation gradient field, modifying the interpolation function locally near heterogeneity. An approach tailored to higher-order MPMs was proposed to create a more accurate and smooth approximation for the MPM. Moustsanidis et al. [29] introduced the IGA-MPM method, incorporating isogeometric analysis (IGA) into MPM. IGA-MPM leverages the concept of IGA, based on Non-Uniform Rational B-Splines (NURBS), to achieve a more precise and smooth approximation for MPM. Additionally, the Barrier Finite Element Point (BFEMP) method [30] aims to combine MPM and FEM through barrier energy-based particle–network friction contact using variable timestep formulas. The MPM has the advantage of simulating tearing, deformations, and collisions in detail but incurs a high computational cost. A method has been proposed to enhance the visual reality of basic simulations using additional representations inspired by physics or heuristics. While a relatively small number of particles can provide the realistic tearing or deformation of a simple elastic object, there are limitations to simulating objects with complex shapes.

Traditional high-resolution simulation methods demand significant computational resources, with the most time-consuming and challenging aspect being pressure calculation. Linear systems, despite the symmetric and positive semidefinite nature of the Laplacian matrix, typically contain numerous free parameters. As a result, standard iterative solvers must perform a large number of iterations to minimize errors, with the iteration count heavily dependent on the data size. With rapid advancements in neural network inference performance, data-driven models have emerged as a promising alternative. Researchers like Tang et al. [31] have utilized fully connected networks to regress Green's function solutions for 2D Laplacian systems, while Yang et al. [32] employed fully connected networks to

accelerate 3D smoke simulation by replacing local PCG solutions. These approaches have achieved impressive speedups in pressure calculation, surpassing traditional methods by more than tenfold without the need for multithreaded computation. Furthermore, Gao et al. [33] enhanced these methods for liquid simulations, incorporating properties like liquid level sets and velocity into their networks to account for differences between smoke and liquids. Xiao et al. [34] applied CNNs to solve the pressure Poisson equation globally, making it suitable for large-scale scenes, while Tompson et al. [35] used CNNs to replace Euler pressure projection by predicting velocity gradients, achieving faster runtimes and superior accuracy compared to Jacobi's method and performances that are comparable to PCG. Kim et al. [36] introduced a novel generative network to accelerate synthetic fluid simulation using reduced parameters. Wiewel et al. [37] divided input velocity and density matrices into latent spatial domains, leveraging a subdivided encoder and LSTM for time predictions in smoke simulations, resulting in significant speed improvements over traditional solvers. Instead of numerical acceleration, another avenue to enhance modeling efficiency and accuracy is detail enhancement. Um et al. [38] achieved substantial reductions in numerical errors in PDE solvers by training artificial neural networks and differentiable physics solvers. Chu et al. [39] employed twin CNNs to determine whether high- and low-resolution patches in the database belong to the same part of the fluid. They then synthesized high-resolution patches into low-resolution images to enhance the resolution. Xie et al. [40] pioneered the use of GANs for four-dimensional functions to train mapping relationships from low resolutions to high resolutions, improving the overall resolution. Xiao et al. [41] presented a fast CNN-based shape correction method that enables low-resolution previews while maintaining high-resolution fluid shapes. However, these methods, relying on local information as the network input, may not guarantee divergence-free conditions. To address this concern, Li et al. [42] proposed an innovative CNN-based regression model to estimate the physical parameters of Eulerian gas. The learned parameters guide high-resolution simulations by combining CNN-based velocity regression. In conclusion, recent years have seen the emergence of many state-of-the-art data-driven approaches for particle simulation, yielding convincing performances.

Our paper proposes a high-quality tearing simulation with lower computational cost compared to the traditional MPM. Using physically parameterized MPM simulation data, we trained a DNN to learn small-scale detailed fragments that tear under external forces. The trained results are then applied to low-cost GPUs experiencing similar forces, enabling the generation of physically accurate results without requiring high-performance GPUs. This enhances the realism of elastic object tearing in virtual environments. A fragment is defined as a separated region of material that exceeds its elastic limit due to external forces. Generating sufficient fragments is crucial for enhancing the realism of tearing simulations. However, due to the complexity of small-scale surface geometry and mechanics, accurate dispersion from the original object is required for those fragments. We utilize NN to represent the accurate generation and movement of fragments without requiring high computational costs. Small-scale fragment formation is learned from physically accurate simulations to generate realistic fragments even in low-resolution simulations. NN is used to learn from the parameterized simulations of highly elastic object tearing, and statistical inference is used to determine whether the result is a fragment. When the fragments are confirmed, the value of the deformation gradient, which is the main factor in determining material changes, is obtained through a loss function based on the mean and variance functions. The determination of fragment existence and the modified deformation gradient are then integrated into the traditional MPM simulation. The proposed method is tested by comparing it with the traditional MPM using a model, which is subjected to force on elastic objects, resulting in fragment generation and movement. Compared to the traditional MPM, which exhibits fragments with abnormal velocity and rotation due to strong forces that occur in a short period of time, the proposed method confirms that the speed and direction of the fragments are stable. As a result, it is determined that the proposed method successfully recognizes crucial mechanisms related to fragment generation from

precomputed data, resulting in improved fragments. The proposed method asserts that this approach provides a solid foundation for learning various real-world physical effects in virtual environments.

The contributions of this paper are as follows:

- To accurately simulate the tearing process of a force-impacted elastic object, we utilize a hybrid approach using both particles and a grid-based method known as the MPM technique.
- We propose a DL-based fracture generation method that learns the fractures occurring during elastic object tearing. This approach allows for the effective simulation of the destruction process even at low resolutions.

Section 2 describes the fragment determination algorithm and learning process for generating accurate fragments. To evaluate the effectiveness of the proposed method, comparative experiments were conducted under conditions similar to the traditional MPM. Finally, the conclusion is summarized.

2. Materials and Methods

This section describes a data-driven approach to creating realistic fragments of elastic objects. The main idea is to infer statistics about fragment generation based on data from parameterized simulations that capture the process of fragment creation in elastic objects in nature. Our method does not require additional mathematically formulated parameters, such as a deformation gradient or curvature threshold, to represent the regions of the destroyed elastic object. It consists of two components, fragment classification and deformation gradient modification, based on the statistical model and data extracted from a series of highly detailed and precomputed simulations. Based on features composed of localized information flow, the classifier predicts whether a specific volume of elastic objects will be destroyed under force within a given duration. For the classified tearing, the modifier predicts the future deformation gradient based on the probability distribution of deformation gradient modifications. It uses NN to represent both components, and the following section describes the statistical model and its NN approach.

2.1. Elastic Object Simulation Based on Constraints Using MPM

Figure 1 describes the overall workflow of the MPM simulation with NN in the proposed method. The P2G stage involves transforming particles into grids, while the G2P stage involves transforming grids back into particles. As the simulation proceeds with the basic MPM structure, we learn the data of grid i, which is adjacent to particle p in the P2G stage. The proposed model is designed to infer whether a particle will change into a damaged state by providing particle information as feature descriptor $x \in A\{f,m\}$. A represents the area of the grid where the particle was placed, f is the deformation gradient that describes the particle's change, and m is the particle's mass. The model is trained on a given dataset composed of particle data $X = \{x_1, x_2, \cdots x_N\}$ and damage determination values $Q = \{q_1, q_2, \cdots, q_N\}$. The goal of the classifier is to infer the probability P_d that a specific vector x_i belongs to the class represented by q_i based on the trained results. Considering the probability distribution function followed by P_d, the function is approximated from the given data. The probability distribution $y_d(x_i,w_d)$ is a target function represented by the weight w_d. Weight w_d is a degree of freedom value that can be adjusted according to the data during the learning phase. P_d can be expressed in the form of y_d as follows:

$$P_d(q_i|x_i) \approx P_d(q_i|y_d(x_i, w_d)) \quad (1)$$

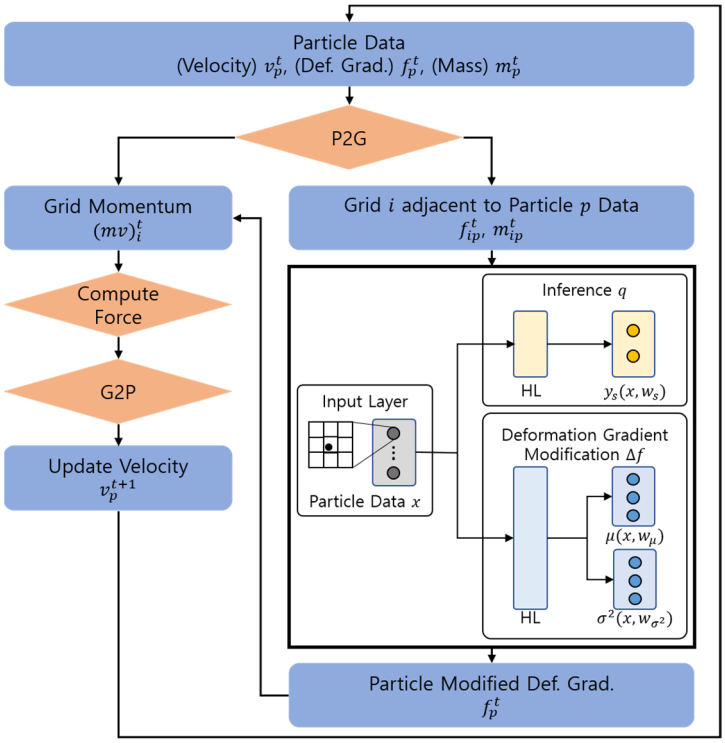

Figure 1. The overall workflow of the proposed MPM simulation with NN, which is a process of classification and modification. Each network had one hidden layer (HL), in which the input size was 1.5 times the input layer.

The probability of damage occurring for a given vector can be expressed using the maximum likelihood equation (MLE) as follows:

$$L_d(Q|X) = \prod_i^N P_d(q_i|y_d(x_i, w_d)) \tag{2}$$

To maximize this likelihood, a well-established softmax function was used for the loss of the classification network. After successfully encoding the given dataset and training the model, the flow's positions can be evaluated with new shape vectors, enabling the prediction of whether the area will be damaged within the given frame.

Based on the inferred data about the damage status, the change in the deformation gradient of both the original elastic object and the fragments was predicted. Δf is the deformation gradient of a damaged elastic object's particle. This value is modified based on the particle's deformation gradient learned from the simulation data. Similarly to the classifier, the proposed method infers the deformation gradient modification set $\Delta F = \{\Delta f_1, \Delta f_2, \cdots, \Delta f_N\}$ based on the feature X. From the training data's statistics, it was found that it is reasonable to assume that the deformation gradient modifications follow a normal distribution relative to the average flow direction. Therefore, the modifier is modeled as a modification function $M(\Delta f_i \mid x_i)$, which follows a normal distribution with mean μ and variance σ^2.

$$M(\Delta f_i|x_i) = \frac{e^{-(\Delta f_i - \mu_i)^2/2\sigma_i^2}}{\sqrt{2\pi\sigma^2}} \tag{3}$$

Additionally, the loss function equation for the corresponding deformation gradient can be expressed as follows:

$$L(\Delta F|X) = \frac{\sum_i^N \sum_j^d (\Delta f_{i,j} - \mu_i)^2/\sigma_i^2 + ln\sigma_{i,j}^2}{2} \tag{4}$$

Equation (4) is based on the mean variance estimation (MVE) formula [43,44]. The MVE formula estimates the mean and variance, assuming that the error follows a normal distribution around the mean, instead of directly estimating the target's mean. $\mu(x_i, w_\mu)$ and $\sigma^2(x_i, {w_\sigma}^2)$ are the target functions approximated by each weight w_μ and ${w_\sigma}^2$, respectively. The proposed mean and variance functions approximate the two sets of weights by estimating them to minimize the loss function L for the given data, $\{X, \Delta F\}$.

The proposed NN model learns about two individual components: classifier and modifier. While sharing the input data x, the two components learn based on separate double layers. The output size of the first layer was 1.5 times the input vector, and the result was connected to the output layer. All outputs of each layer were activated using a hyperbolic tangent function. To train the NN, a large dataset with target outputs is needed. The model consists of classification results q, which indicate whether fragments are formed, and a deformation gradient Δf, which predicts the trajectory of the fragments. The number, position, mass, and deformation gradient of particles forming the elastic object as initial conditions were used to generate training data in the tearing simulation. Then, each condition's range was selected to ensure sufficient data generation variability.

2.2. Fragment Detection Algorithm

To train the neural network model illustrated in Figure 1, it is imperative to establish a dataset comprising training and validation data. This dataset should encompass various environmental conditions, such as free fall or compression, where deformations lead to the formation of fragments. To effectively segment these fragments, an algorithm is required. The criterion for determining the fragments is the continuous number of particles included in the grid. The breadth-first search (BFS) method is utilized for searching nearby grids based on a specific grid [45]. While there are advantages to searching for fragments in the grid in detail, the time complexity can increase up to $O(N^2)$ for a grid of size $N \times N$. Performing this algorithm in a GPU environment can reduce the operation per core d to $O(N^2/d)$. However, since this method fundamentally uses a queue data structure, if the size of the queue that each thread processes cannot be adjusted, load imbalance may occur when using it in a GPU. The method proposed in Algorithm 1 utilizes storing the minimum value in a hash table to enable efficient searching in a GPU environment without causing a load [46].

Algorithm 1: Fragment detection algorithm

```
1: procedure searchFragmentParticles():
2:       Init grid G
3:       for each particle P_i:
4:             Hash Table T ← (P_i, G index)
5:             SetGroupNumber(T)
6:           if number of G includes the min fragment =< P_i < max fragment
7:              Fragment = P_i
8:     end for
9: end procedure
```

With particle P_i and its corresponding up, down, left, and right coordinates, the minimum non-zero value of the group was stored in the grid. The grid coordinates were shifted one by one, and when a continuous non-zero value was discovered, the value of the continuous group was changed to the minimum value. This was repeated for all the grids. When searching for particle P_i, if a group larger than the specified minimum area and smaller than the maximum is found, it is considered a fragment.

3. Experimental Results

3.1. Fragment Movement during Elastic Object Tearing

In this paper, we utilized i9 CPU and RTX 3090 GPU hardware and configured the experimental environment with 25,000 particles and a grid size of 128 × 128 using the

Python-based Taichi library. The proposed NN-based fragment generation model easily integrates into the traditional MPM simulation pipeline. Following the momentum update step, classification was performed for all particles within the up, down, left, and right range, including the reference grid. If positive results were generated, the deformation gradient modification network was evaluated to calculate the mean and variance of the components. Subsequently, random numbers were generated from the parameterized normal distribution and used to update the deformation gradient of the new fragments. Fragments with the applied deformation gradient do not participate in the physical calculations of the MPM simulation until they collide with another elastic object. As a result, our method reduces the occurrence of unexpected velocities or rotations of fragments caused by sudden forces. The experimental results for the mentioned content are presented in Figure 2.

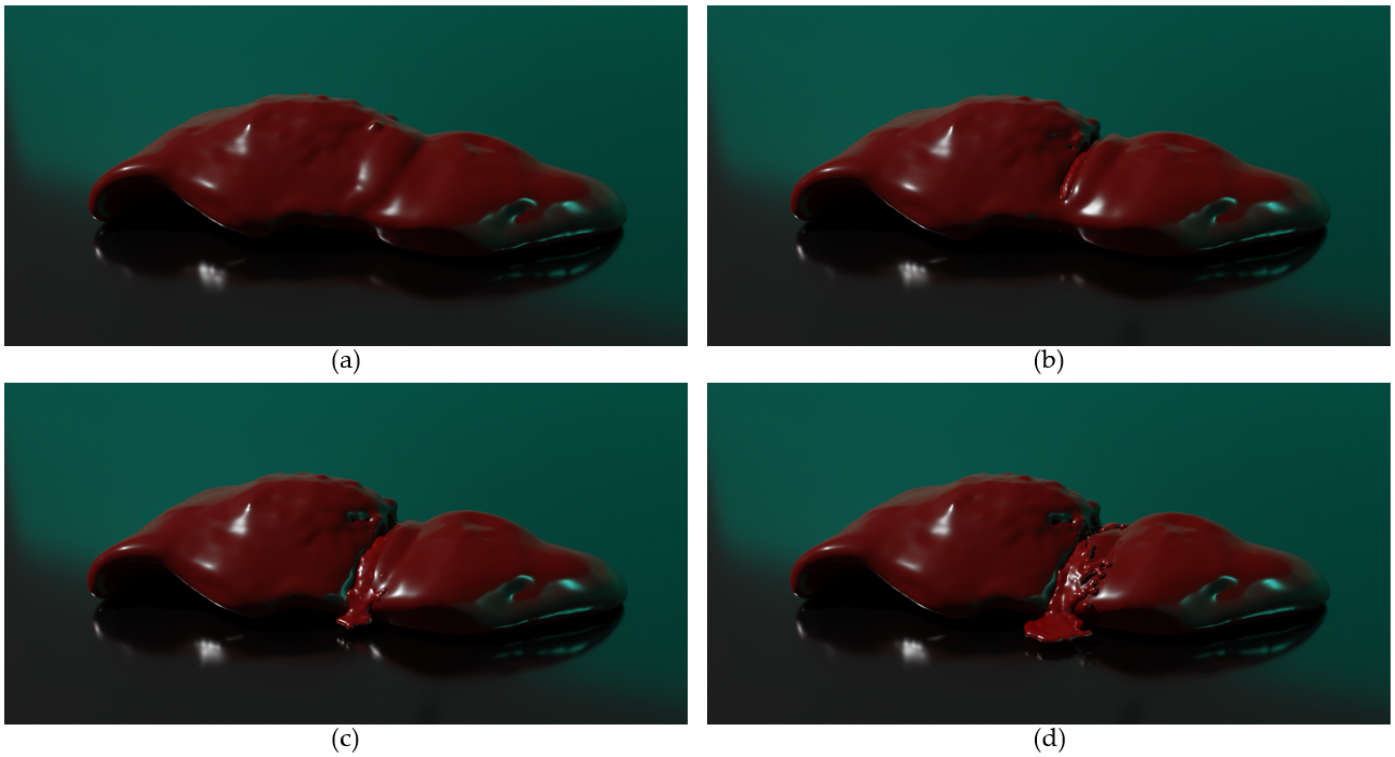

Figure 2. Cutting a 3D organ model with the proposed method. (**a**) State without any external force applied. (**b**) 30% of cutting progress from a specific position. (**c**) 60% of cutting progress. (**d**) 100% of cutting progress.

Figure 2 shows the results of cutting a 3D organ model. We have observed the fragments formed when elastic objects with varying degrees of elasticity, ranging from rubber to gel, are torn. We hypothesized that elastic objects with internal fluids, such as organs, could also be learned as fragments. To prove this, we created 3D elastic objects with similar elasticity to organs and compared the results with the traditional MPM method. (a) shows the initial state where no force is applied to the object, maintaining its shape. (b) shows the process where cutting begins as force is applied to the object, resembling the use of a scalpel. The internal fluid within the elastic object is expelled through the incision. In contrast to organs like the heart, the relatively low internal pressure and the learned outcome result in minimal ejection. Complete severance is achieved as the incision progresses from (c) to (d). While the internal pressure remains relatively low, a significant amount of fluid flows out as the incision widens, causing the exterior of the severed elastic object to contract. In the case of the traditional MPM, the force applied to the particles along the x, y, and z axes was excessively high at the moment of fragmentation, resulting in particles bouncing in all directions despite their mass. By contrast, the proposed method

suppressed excessive movements based on the learned data and produced the expected results. When comparing the degree of fragment bouncing to real-life visual media, it can be observed that the proposed method yields results similar to the traditional MPM, confirming its superiority. However, the limitation lies in the lack of testing with a variety of body models, and the failure to make an accurate comparison with real-life objects is also a limitation.

3.2. Algorithm Execution Time Comparison

Table 1 is a comparison table between the BFS and hash table algorithms, which are used to determine phenomena occurring in specific areas. As the size of particles and grids increases, the BFS-based search algorithm for fragment detection takes a long time to search the queue due to duplicate nodes, resulting in the outcome shown in Table 1. MPM simulates the deformation of elastic objects based on the movement of individual particles. The level of detail in animations increases as the size of particles decreases and the number of particles increases. Since the neural networks learning the particle changes also scale with detail, the proposed method is highly useful in content such as movies or games where many elastic objects appear on the screen.

Table 1. Performance comparison of the BFS method and the proposed method.

Algorithm	Execution Time (FPS)	Training Data Generation Time (FPS)
BFS	794	8.10
Hash Table	1695	17.30

3.3. Simulation Time

Figure 3 is a graph that verifies whether the simulation of the proposed idea causes any frame difference compared to the traditional MPM. Since the occurrence of fragments is measured using the data learned based on NN, no additional calculations are required in the final simulation, and it approaches a stable frame rate of 60 fps. Moreover, since parallel operations are performed in the GPU environment, there is little change in frames between cases with 9000 particles and 25,000 particles. When simulating the learned results, there can sometimes be a significant difference in FPS compared to conventional simulations due to the learning method or process. Our proposed method has conducted FPS comparisons in extreme environments ranging from relatively few particles to many particles, demonstrating superiority over the traditional MPM with differences within 10%.

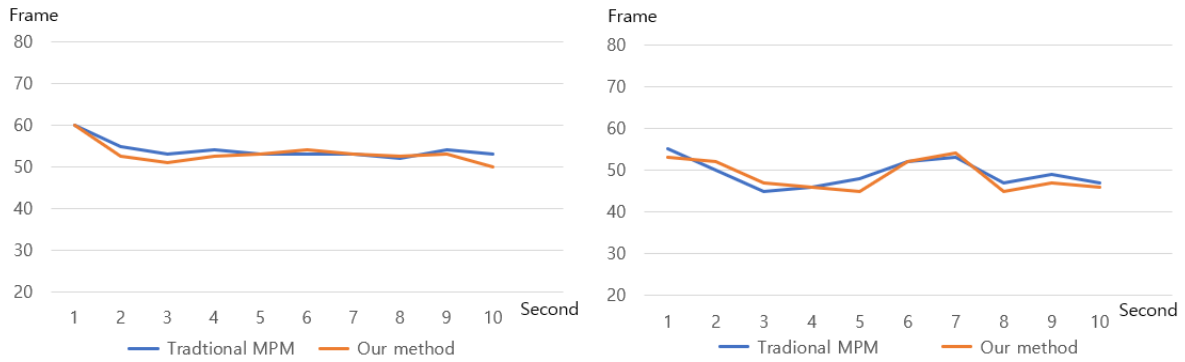

Figure 3. Comparison of simulation FPS between MPM and the proposed method. Comparison with 9000 particles (**left**). Comparison with 25,000 particles (**right**).

4. Conclusions

The method proposed in this paper allows for a more realistic representation of the tearing of elastic objects through fragment recognition. Typically, high-performance GPUs are required to achieve this. However, the proposed method utilizes neural networks (NNs)

to remove movements that cannot be observed in real physical phenomena, eliminating the need for high-performance GPUs. The proposed method combines MPM simulations with neural networks to achieve the realistic tearing and fragmentation of elastic objects like human organs. A key advantage is removing the need for high-performance GPUs by utilizing efficient algorithms and neural network models. The experimental results clearly demonstrate the effectiveness of the proposed approach. As evident from Figure 3, when increasing the number of particles from 9000 to 25,000, the frame rate remained stable at around 60 and 50 fps for both the traditional MPM and the proposed method. This highlights the proposed method's ability to handle complex, high-resolution scenarios involving a large number of particles without introducing any significant performance overhead compared to the traditional MPM simulation. The seamless integration of neural networks for fragment recognition and modification does not adversely impact the real-time performance, making the proposed technique well suited for virtual surgery and other applications demanding high fidelity and responsiveness. Moreover, the hash table-based fragment detection algorithm proved to be 2.14 times faster than the BFS method at generating the required training data for the neural networks. This efficient data generation enabled the learning of fragment behaviors across a wide range of materials and conditions. Furthermore, when an elastic object is destroyed by sudden force, the traditional MPM expels fragments with strong force. In contrast, the proposed method allows for the control of unnatural particle scattering and bouncing effects through learning, enabling stable and physically plausible fragment motions. While our work presents a promising approach for accurately simulating the tearing process of elastic objects, several limitations and challenges should be addressed. One notable disadvantage of the proposed method is the limited discussion of its potential limitations and areas for improvement. Although the comparative experiments with the traditional MPM demonstrate the superiority of the proposed method in terms of stable fragment speed and direction, further evaluations using larger datasets or real-world scenarios would enhance the credibility and generalizability of the findings. Additional research is also needed regarding the complexity of organ structures. While organs' elasticity and shape details can be simulated similarly to previous studies through tensile experiments and adjusting the number of particles, detailed investigations are required for organs with complex internal structures. For example, in the case of the heart, there are atria and ventricles, as well as valves between them. Additionally, pumping for blood supply must also be considered when creating the model.

Author Contributions: Conceptualization, S.-K.S.; methodology, S.-K.S.; validation, J.-H.K.; formal analysis, S.-K.S.; investigation, S.-K.S. and J.-H.K.; resources, S.-K.S.; data curation, J.-H.K.; writing—original draft preparation, S.-K.S.; writing—review and editing, B.-S.S.; visualization, J.-H.K.; supervision, B.-S.S.; project administration, S.-K.S. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by the INHA University grant.

Data Availability Statement: The raw data supporting the conclusions of this article will be made available by the authors upon request.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Gibson, S.F.F.; Mirtich, B. *A Survey of Deformable Modeling in Computer Graphics*; Mitsubishi Electric Research Laboratories: Cambridge, MA, USA, 1997.
2. Selle, A.; Lentine, M.; Fedkiw, R. A Mass Spring Model for Hair Simulation. In *ACM SIGGRAPH 2008 Papers*; Association for Computing Machinery: New York, NY, USA, 2008; pp. 1–11.
3. Choi, K.-J.; Ko, H.-S. Stable but Responsive Cloth. In *ACM SIGGRAPH 2005 Courses*; Association for Computing Machinery: New York, NY, USA, 2005; pp. 604–611.
4. Nealen, A.; Müller, M.; Keiser, R.; Boxerman, E.; Carlson, M. Physically Based Deformable Models in Computer Graphics. *Comput. Graph. Forum* **2006**, *25*, 809–836. [CrossRef]

5. Bro-Nielsen, M.; Cotin, S. Real-Time Volumetric Deformable Models for Surgery Simulation Using Finite Elements and Condensation. *Comput. Graph. Forum* **1996**, *15*, 57–66. [CrossRef]
6. Lee, B.; Popescu, D.C.; Joshi, B.; Ourselin, S. Efficient Topology Modification and Deformation for Finite Element Models Using Condensation. *Stud. Health Technol. Inf.* **2006**, *119*, 299–304.
7. Felippa, C. A Systematic Approach to the Element-Independent Corotational Dynamics of Finite Elements; University of Colorado, CO, USA, 2000.
8. Taylor, Z.A.; Comas, O.; Cheng, M.; Passenger, J.; Hawkes, D.J.; Atkinson, D.; Ourselin, S. Modelling Anisotropic Viscoelasticity for Real-Time Soft Tissue Simulation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2008: 11th International Conference, New York, NY, USA, 6–10 September 2008*; Metaxas, D., Axel, L., Fichtinger, G., Székely, G., Eds.; Springer: Berlin/Heidelberg, Germany, 2008; pp. 703–710.
9. Asareh, I.; Song, J.-H.; Mullen, R.L.; Qian, Y. A general mass lumping scheme for the variants of the extended finite element method. *Int. J. Numer. Methods Eng.* **2020**, *121*, 2262–2284. [CrossRef]
10. Zhang, W.; Zhong, Z.; Peng, C.; Yuan, W.; Wu, W. GPU-Accelerated Smoothed Particle Finite Element Method for Large Deformation Analysis in Geomechanics. *Comput. Geotech.* **2021**, *129*, 103856. [CrossRef]
11. Joldes, G.R.; Wittek, A.; Couton, M.; Warfield, S.K.; Miller, K. Real-Time Prediction of Brain Shift Using Nonlinear Finite Element Algorithms. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2009: 12th International Conference, London, UK, 20–24 September 2009*; Yang, G.-Z., Hawkes, D., Rueckert, D., Noble, A., Taylor, C., Eds.; Springer: Berlin/Heidelberg, Germany, 2009; pp. 300–307.
12. Brackbill, J.U.; Ruppel, H.M. FLIP: A Method for Adaptively Zoned, Particle-in-Cell Calculations of Fluid Flows in Two Dimensions. *J. Comput. Phys.* **1986**, *65*, 314–343. [CrossRef]
13. Zhu, Y.; Bridson, R. Animating Sand as a Fluid. *ACM Trans. Graph.* **2005**, *24*, 965–972. [CrossRef]
14. Stomakhin, A.; Schroeder, C.; Chai, L.; Teran, J.; Selle, A. A Material Point Method for Snow Simulation. *ACM Trans. Graph.* **2013**, *32*, 1–10. [CrossRef]
15. Daviet, G.; Bertails-Descoubes, F. A Semi-Implicit Material Point Method for the Continuum Simulation of Granular Materials. *ACM Trans. Graph.* **2016**, *35*, 1–13. [CrossRef]
16. Klar, G.; Gast, T.; Pradhana, A.; Fu, C.; Schroeder, C.; Jiang, C.; Teran, J. Drucker-Prager Elastoplasticity for Sand Animation. *ACM Trans. Graph.* **2016**, *35*, 1–12. [CrossRef]
17. Ram, D.; Gast, T.; Jiang, C.; Schroeder, C.; Stomakhin, A.; Teran, J.; Kavehpour, P. A Material Point Method for Viscoelastic Fluids, Foams and Sponges. In Proceedings of the 14th ACM SIGGRAPH/Eurographics Symposium on Computer Animation, Los Angeles, CA, USA, 7–9 August 2015; Association for Computing Machinery: New York, NY, USA, 2015; pp. 157–163.
18. Yue, Y.; Smith, B.; Batty, C.; Zheng, C.; Grinspun, E. Continuum Foam: A Material Point Method for Shear-Dependent Flows. *ACM Trans. Graph.* **2015**, *34*, 1–20. [CrossRef]
19. Jiang, C.; Gast, T.; Teran, J. Anisotropic Elastoplasticity for Cloth, Knit and Hair Frictional Contact. *ACM Trans. Graph.* **2017**, *36*, 1–14. [CrossRef]
20. Stomakhin, A.; Schroeder, C.; Jiang, C.; Chai, L.; Teran, J.; Selle, A. Augmented MPM for Phase-Change and Varied Materials. *ACM Trans. Graph.* **2014**, *33*, 1–11. [CrossRef]
21. Tampubolon, A.P.; Gast, T.; Klár, G.; Fu, C.; Teran, J.; Jiang, C.; Museth, K. Multi-Species Simulation of Porous Sand and Water Mixtures. *ACM Trans. Graph.* **2017**, *36*, 1–11. [CrossRef]
22. Wretborn, J.; Armiento, R.; Museth, K. Animation of Crack Propagation by Means of an Extended Multi-Body Solver for the Material Point Method. *Comput. Graph.* **2017**, *69*, 131–139. [CrossRef]
23. Cherepanov, G.P. Crack propagation in a continuum. *Prikl. Mat. Mekh. (USSR)* **1967**, *31*, 476–488.
24. Terashima, H.; Tryggvason, G. A front-tracking/ghost-fluid method for fluid interfaces in compressible flows. *J. Comput. Phys.* **2009**, *228*, 4012–4037. [CrossRef]
25. Wolper, J.; Fang, Y.; Li, M.; Lu, J.; Gao, M.; Jiang, C. CD-MPM: Continuum damage material point methods for dynamic fracture animation. *ACM Trans. Graph.* **2019**, *38*, 1–15. [CrossRef]
26. Xiong, P.-Y.; Almarashi, A.; Dhahad, H.A.; Alawee, W.H.; Issakhov, A.; Chu, Y.-M. Nanoparticles for Phase Change Process of Water Utilizing FEM. *J. Mol. Liq.* **2021**, *334*, 1–11. [CrossRef]
27. Gao, M.; Tampubolon, A.P.; Jiang, C.; Sifakis, E. An Adaptive Generalized Interpolation Material Point Method for Simulating Elastoplastic Materials. *ACM Trans. Graph.* **2017**, *36*, 1–12. [CrossRef]
28. Moutsanidis, G.; Kamensky, D.; Chen, J.S.; Bazilevs, Y. Hyperbolic Phase Field Modeling of Brittle Fracture: Part II—Immersed IGA–RKPM Coupling for Air-Blast–Structure Interaction. *J. Mech. Phys. Solids* **2018**, *121*, 114–132. [CrossRef]
29. Moutsanidis, G.; Long, C.C.; Bazilevs, Y. IGA-MPM: The Isogeometric Material Point Method. *Comput. Methods Appl. Mech. Eng.* **2020**, *372*, 113346. [CrossRef]
30. Li, X.; Fang, Y.; Li, M.; Jiang, C. BFEMP: Interpenetration-Free MPM–FEM Coupling with Barrier Contact. *Comput. Methods Appl. Mech. Eng.* **2022**, *390*, 114350. [CrossRef]
31. Tang, J.; Azevedo, V.C.; Cordonnier, G.; Solenthaler, B. Neural Green's Function for Laplacian Systems. *Comput. Graph.* **2022**, *107*, 186–196. [CrossRef]
32. Yang, C.; Yang, X.; Xiao, X. Data-Driven Projection Method in Fluid Simulation. *Comput. Animat. Virtual Worlds* **2016**, *27*, 415–424. [CrossRef]

33. Gao, Y.; Zhang, Q.; Li, S.; Hao, A.; Qin, H. Accelerating liquid simulation with an improved data-driven method. *Comput. Graph. Forum* **2020**, *39*, 180–191. [CrossRef]
34. Xiao, X.; Zhou, Y.; Wang, H.; Yang, X. A novel CNN-based Poisson solver for fluid simulation. *IEEE Trans. Vis. Comput. Graph.* **2020**, *26*, 1454–1465. [CrossRef] [PubMed]
35. Tompson, J.; Schlachter, K.; Sprechmann, P.; Perlin, K. Accelerating Eulerian Fluid Simulation with Convolutional Networks. In Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, 6–11 August 2017; pp. 3424–3433.
36. Kim, B.; Azevedo, V.C.; Thuerey, N.; Kim, T.; Gross, M.; Solenthaler, B. Deep fluids: A generative network for parameterized fluid simulations. *Comput. Graph. Forum* **2019**, *38*, 59–70. [CrossRef]
37. Wiewel, S.; Kim, B.; Azevedo, V.C.; Solenthaler, B.; Thuerey, N. Latent space subdivision: Stable and controllable time predictions for fluid flow. *Comput. Graph. Forum* **2020**, *39*, 15–25. [CrossRef]
38. Um, K.; Brand, R.; Fei, Y.R.; Holl, P.; Thuerey, N. Solver-in-the-Loop: Learning from Differentiable Physics to Interact with Iterative PDE-Solvers. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 6111–6122.
39. Chu, M.; Thuerey, N. Data-Driven Synthesis of Smoke Flows with CNN-Based Feature Descriptors. *ACM Trans. Graph.* **2017**, *36*, 1–14. [CrossRef]
40. Xie, Y.; Franz, E.; Chu, M.; Thuerey, N. TempoGAN: A temporally coherent, volumetric GAN for super-resolution fluid flow. *ACM Trans. Graph.* **2018**, *37*, 1–15. [CrossRef]
41. Xiao, X.; Wang, H.; Yang, X. A CNN-based flow correction method for fast preview. *Comput. Graph. Forum* **2019**, *38*, 431–440. [CrossRef]
42. Li, C.; Qiu, S.; Wang, C.; Qin, H. Learning physical parameters and detail enhancement for gaseous scene design based on data guidance. *IEEE Trans. Vis. Comput. Graph.* **2020**, *27*, 3867–3880. [CrossRef] [PubMed]
43. Gasteiger, J.; Giri, S.; Margraf, J.T.; Günnemann, S. Fast and Uncertainty-Aware Directional Message Passing for Non-Equilibrium Molecules. *arXiv* **2020**, arXiv:2011.14115.
44. Khosravi, A.; Nahavandi, S.; Creighton, D.; Atiya, A.F. Comprehensive Review of Neural Network-Based Prediction Intervals and New Advances. *IEEE Trans. Neural Netw.* **2011**, *22*, 1341–1356. [CrossRef] [PubMed]
45. Hao, L.; Wang, Y.; Bai, Y.; Zhou, Q. Energy Management Strategy on a Parallel Mild Hybrid Electric Vehicle Based on Breadth First Search Algorithm. *Energy Convers. Manag.* **2021**, *243*, 114408. [CrossRef]
46. Lessley, B.; Childs, H. Data-Parallel Hashing Techniques for GPU Architectures. *IEEE Trans. Parallel Distrib. Syst.* **2020**, *31*, 237–250. [CrossRef]