Computer Methods and Programs in Biomedicine 250 (2024) 108171

Contents lists available at ScienceDirect

Computer Methods and Programs in Biomedicine

journal homepage: www.sciencedirect.com/journal/computer-methods-and-programs-in-biomedicine

Real-time soft body dissection simulation with parallelized graph-based shape matching on GPU

Peng Yu, Zhiyuan Zhao, Ruiqi Wang, Junjun Pan *

State Key Laboratory of Virtual Reality Technology and Systems, School of Computer Science and Engineering, Beihang University, Beijing, China

ARTICLE INFO

Keywords:
K.6.1 [computer graphics]
Virtual surgery simulation

ABSTRACT

Background and Objective: Interactive soft tissue dissection has been a fundamental procedure in virtual surgery systems. Existing cutting algorithms involve complex topology changes of simulation meshes, which can increase simulation overhead and produce visual artifacts. In this paper, we proposed a novel graph-based shape-matching method that allows for real-time, flexible, progressive, and discontinuous cuts on soft tissue.

Methods: We employed shape-matching constraints within the position-based dynamics (PBD) framework, a widely adopted approach for real-time simulation applications. The soft tissue was effectively modeled using overlapping clusters, each governed by shape-matching constraints. The dissection process was bifurcated into two distinct stages. In the first stage, the surgical scalpel presses the surface of the soft tissue. The soft tissue is cut apart when the surface pressure exceeds a threshold, entering the second stage. To address the discrepancy between the visual mesh and the simulation model during cluster separation, we developed an Aggregate Finding Connected Components (AFCC) algorithm, optimized for GPU computation and integrated with a background grid. This approach also avoids ghost forces and fragmentation artifacts. To control the increase in the number of clusters, we also propose a merging strategy that can run in parallel.

Results: Our simulation outcomes demonstrated that the AFCC dissection algorithm effectively manages cluster separation and expansion with robustness. There were no ghost forces between the cutting surface and unrealistic fragments. Our simulation capability extended to supporting intricate and discontinuous cutting routes. Our dissection simulation maintained real-time performance even with over 100,000 particles constituting the soft tissue.

Conclusions: Our real-time and robust surgical dissection simulation technique enables the performance of complex cuts in various surgical scenarios, demonstrating its potential in virtual surgery applications.

1. Introduction

In recent years, as illustrated in Fig. 1, virtual reality-based surgery simulators have become popular because of their flexibility, reusability, and low surgical risks. It was confirmed that the surgical skills learned on virtual reality-based surgery simulators could be transferred to real surgeries [1,2]. Soft body dissection is one of the most fundamental minimally invasive surgeries with wide application in the fields of cholecystectomy, hepatectomy, colectomy, etc. The simulation of soft body dissection involves modeling soft bodies, rigid surgery tools, and their complex interactions, such as collision detection and topology changes after cutting. For effective operator feedback and comprehensive decision-making, the simulation algorithm should be robust, real-time, and enable visually plausible.

Over the past few decades, researchers have explored various techniques for soft-body dissections. Pan et al. [3] employed a mass-spring system to model membranes and fat tissues in intestine dissection applications. Although the mass-spring method is fast and easy to implement, adjusting the stiffness parameters for plausible visual effects and preserving soft tissue volume can be challenging. Another commonly used scheme, the Finite Element method (FEM), originally comes from continuum mechanics and is physically accurate. Hadrien et al. [4] proposed a numerical method for interactive simulations for heterogeneous soft-tissue models based on the implicit finite element method. They would asynchronously update the preconditioner to handle the deformation and topology changes. Even though simulated on GPU, it can only use coarse volume mesh to model the soft tissue. Huu Phuoc Bui et al. [5] used corotational FEM to model soft tissue in the needle

* Corresponding author.
E-mail address: pan_junjun@buaa.edu.cn (J. Pan).

https://doi.org/10.1016/j.cmpb.2024.108171
Received 11 December 2023; Received in revised form 8 April 2024; Accepted 8 April 2024
Available online 11 April 2024

Fig. 1. The medical students are training on VR laparoscopic surgery simulator [2].

insertion surgical procedure. Wang et al. [6] solved the equation of motion on GPU to accelerate the performance of cutting based on FEM. Heiden et al. [7] employed differentiable simulators to cut soft materials models by using a signed distance field and FEM. However, these methods were limited by a low-resolution grid due to the updating of a large sparse system matrix. Performance significantly decreases when cuts and topology changes occur.

The Material Point Method (MPM), popularized by Stomakhin et al. [8] for snow modeling, saw widespread adoption. It is an Eluer-Lagrangian hybrid method that uses material points to store physics parameters such as mass and density and uses an Euler grid as a background grid to solve equations of motion. MPM could handle the collision and fracture automatically. Hu et al. [9] introduced the CPIC algorithm, facilitating bidirectional coupling between rigid thin plates and the material points in cutting simulations. Wolper et al. [10] proposed a non-local continuum damage mechanics-based MPM to model soft body fractures. However, how to couple nonplanar rigid tools with MPM and generate a complex cutting path is still an open question. The MPM approach is prone to numerical issues, including stickiness and stability concerns. To learn more about the MPM method, we refer to the Siggraph course [11].

The Position-Based Dynamics (PBD) methodology, renowned for its robustness and superior performance, is extensively utilized in real-time applications, including gaming and virtual surgical simulations [12,13]. Pan et al. [14] simulated the soft tissue under the PBD framework with distance and volume constraints. This paper proposed a tetrahedron mesh subdivision and remapping method to drive surface mesh deformation. While they relied on CPUs for cutting simpler models (up to thousands of vertices), our approach leveraged GPUs to tackle complex models with tens of thousands of vertices. Pan et al. [15] demonstrated a hybrid approach combining the moving least squares mess-free strategy and meatball-based PBD methods to create physically accurate cutting. In the cutting simulation, They used the nodal points in the mesh-free approach to replace the dissected meatballs. However, the number of nodal points would increase with cutting, increasing the performance cost. Li et al. [16] addressed topology changes by updating a Cholesky factorization of the global step system matrix of projective dynamics. It can only achieve interactive frames that are unsuitable for real-time surgical simulators. Berndt et al. [17] introduced an interactive cutting framework enhanced with haptic feedback, utilizing the Nvidia Flex development library [18]. The soft tissue was discretized with multiple clusters of particles, and a rigid shape-matching constraint constrains each cluster. When cutting happens, the separation strategy is not robust enough, which would cause visual artifacts such as unrealistic fragments and ghost force after cutting. Besides, previous cutting algorithms generated cutting paths when surgical tools collide with the surface mesh of soft tissue. In the real world, cutting happens when the exerting force on the surface mesh is beyond a force threshold [19].

To alleviate the above challenges, we proposed a novel parallel aggregate finding connected components-based shape matching method, which could robustly resolve the unrealistic debris and ghost force phenomena. To address the potential issue of a continuously growing number of clusters during dissection, we introduced a parallel cluster merging algorithm. This paper utilized a distance-based cutting detection algorithm to simulate the initial deformation of the soft body surface caused by the scalpel pressing against it. This paper presented a real-time surgical cutting simulation system capable of handling complex cutting paths. The system leveraged a novel parallel graph-based shape-matching method to achieve realistic soft tissue deformation and progressive, discontinuous visual feedback during the dissection process.

In summary, our contributions include:

- A novel parallel graph-based algorithm that leverages shape-matching constraints and background grids was proposed. This approach ensures consistency between the visual surface mesh and the underlying physical model during soft tissue dissection simulations.
- A parallel distance-based strain method was introduced for the real-time detection of dissection events, which also enables the simulation of a more realistic initial indentation as the scalpel presses against the soft tissue.
- To evaluate the method's ability to handle complex cutting paths, various cutting simulations were conducted. The results demonstrated the effectiveness of the proposed approach compared to existing methods.

The rest of this paper is organized as follows: Section 2 demonstrates the theory and method for building the simulation model of soft tissue and the cutting algorithms. Section 3 illustrates the simulation and experiment results of the soft tissue dissection algorithm and performance. A detailed discussion is introduced in Section 4. A summary of the main conclusions and areas requiring further research in the future closes the paper in Section 5.

2. Material and methods

2.1. Soft tissue simulation

For the sake of comprehensiveness, this paper provides a succinct overview of the soft tissue simulation model utilizing Position-Based Dynamics (PBD) with shape-matching constraints [18]. The soft body is discretized in space into n particles, denoted as $\mathbf{x} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\} \in \mathbb{R}^{3n}$. The Position-Based Dynamics (PBD) frame-work comprises three principal stages. Initially, particles are sub-jected to advection by external forces, such as gravity, independent of the internal forces imposed by PBD constraints. These resultant locations are termed predicted positions $\tilde{\mathbf{x}} = \mathbf{x}^t + h\mathbf{v}^t + h^2\mathbf{M}^{-1}\mathbf{f}_{ext}$, where the superscript t means the time at t, $\mathbf{v}$ is the velocities of articles, h is the time step size, $\mathbf{M}$ is the lumpped diagonal mass matrix and $\mathbf{f}_{ext}$ is the external force. Next, constraints are iteratively projected using a block Gauss-Seidel solver. However, to fully harness the parallel processing capabilities of GPUs, the Jacobi solver is often preferred for constraint projection, despite its requirement for a greater number of iterations to achieve convergence. Ultimately, particle velocities are updated in parallel.

The Position-Based Dynamics (PBD) framework supports various types of constraints that serve different purposes in the simulation of deformable objects. For example, the distance constraint, denoted as $c(\mathbf{x}_i, \mathbf{x}_j) = \| \mathbf{x}_i, \mathbf{x}_j \| - l = 0$ aims to maintain the rest length l between two particles $\mathbf{x}_i$, $\mathbf{x}_j$ and $i, j \in [1, n]$. Shape matching constraint, introduced by Müller et al. [20,21], is another important type of constraint used in the PBD framework. Shape matching constraint assumes that a cluster of particles undergoes only rigid body motion, which is an affine transformation consisting of rotation and translation. This requires solving a least square problem:

$$\mathbf{x}^{t+1} = \min \sum_{i=1}^{n} \| \tilde{\mathbf{x}}_i - \mathbf{t} - \mathbf{R}(\mathbf{x}_i - \mathbf{t}_0) \|^2, \tag{1}$$

where $\mathbf{t}_0$ is the center of mass of rest configuration and $\mathbf{t}$ is the center of

mass of deformed configuration. The rotation **R** matrix is computed by setting the gradient of Eq. (1) with respect to **R** equals to zero and extracting the rotation component using polar decomposition, as shown in Eq. (2),

$$\mathbf{T} = \sum_{i=1}^{n} m_i \mathbf{r}_i \bar{\mathbf{r}}_i^T = \mathbf{RS}, \tag{2}$$

where $\mathbf{r}_i = \tilde{\mathbf{x}}_i - \mathbf{t}$ and $\bar{\mathbf{r}}_i^T = \mathbf{x}_i - \mathbf{t}_0$. The stretching component **S** is discarded because it represents the non-rigid deformation of the particle cluster, which is not considered in shape matching constraints.

To achieve more controllable deformation of soft tissues, the particles comprising the soft tissue are divided into groups of clusters, with each cluster constrained by a shape-matching constraint, as illustrated in Fig. 2. The soft tissue, represented by the blue area, is discretized into nodes on the background grid, depicted as solid blue circles. The solid black circles on the grid denote free nodes without constraints and have no physical meaning. The particles are divided into multiple clusters, indicated by the yellow circles. There is overlap between clusters, and each particle is weighted and can be influenced by multiple clusters. For example, the particle **p** is controlled by three clusters $\mathbf{c}_1$, $\mathbf{c}_2$, $\mathbf{c}_3$ depicted as solid red circles. The red circle represents the center of mass of each cluster. The correction of **p** by each shape matching constraint is applied by weight $w_i, i \in \{1, 2, 3\}$.

To achieve convincing visual effects, we need to generate a surface triangle mesh for rendering. The skinning scheme presented in [18] is unsuitable for scenarios involving topology changes. Using the marching cubes method to dynamically construct a surface triangle mesh introduces visual perturbation. To address this issue, Berndt et al. [17] proposed a modified marching cube method that deforms the background grid to enclose the inner simulation particles. However, when cutting occurs, the generated visual surface mesh becomes inconsistent with the underlying physical model, resulting in visual artifacts such as ghost forces and unrealistic fragments. We incorporate their method into our paper and resolve the visual artifacts problem using a graph-based algorithm.

2.2. Cutting simulation

To simulate cuts on a soft body discretized with a group of clusters, we need a strategy to carefully and simultaneously separate clusters and skinning surface meshes. However, as illustrated in Fig. 3, after several cuts, the resulting skinning surfaces for visualization are inconsistent with the clusters used for physical simulation. Even though the soft body is visually cut into four disconnected parts, the cluster is only cut into two disconnected parts. Visually separated meshes remain connected in the physical model, producing ghost forces, and the movement of disconnected meshes affects each other. To address the issues mentioned above, we must ensure consistency between the physical model and the visual surface mesh. For instance, in Fig. 3, four disconnected clusters should be generated, with each cluster enclosed by a disconnected surface mesh. In this paper, we proposed a connected component approach to represent the clusters.

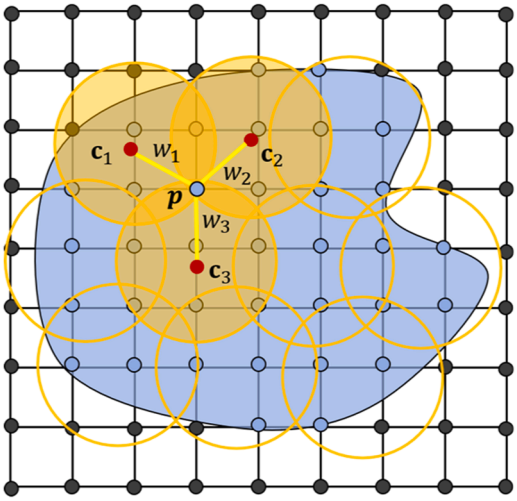


Fig. 2. Spatial discretization of a soft tissue.

2.2.1. Cluster splitting

As mentioned earlier, the soft tissue is discretized as particles (blue solid circles in Fig. 2) on the background grid. In our paper, we construct an undirected graph $G(V, E)$ on the background grid, where V represents the soft tissue particles, and $E \in V \times V$ is a subset of the edges of the background grid. For any edge $e \in E$, the two endpoints $\mathbf{x}_i$, $\mathbf{x}_j$ of the edge belongs to V . Each cluster is defined as a subgraph $G_c(V_c, E_c)$ of graph G, $V_c \in V$, $E_c \in E$. By this definition, any two soft tissue particles $\mathbf{x}_i$, $\mathbf{x}_j$ in V are connected and reachable in graph G, meaning there exists a path $L = \{\mathbf{v}_0 = \mathbf{x}_i, \mathbf{v}_1, \ldots, \mathbf{v}_m = \mathbf{x}_j\}$, where $\mathbf{v}_i \in V$ and edge $(\mathbf{v}_i, \mathbf{v}_{i+1}) \in E$. This assertion also holds true for every individual cluster. It is straightforward to prove that if each cluster remains connected, consistency between the visual surface mesh and the underlying physical model can be ensured.

Upon the occurrence of cutting, particles meeting the removal criteria (indicated as yellow particles in Fig. 3) will be excluded from the particle set V and simultaneously removed from the connectivity graph G. During this process, a single cluster may potentially be divided into multiple disconnected parts, necessitating the generation of new clusters. Through observation, we discovered that the disconnected clusters are connected components of graph G. A connected component of an undirected graph G is a maximal subset S of its vertices where all vertices in S are reachable from each other. As illustrated in the right-most image of Fig. 3, four disconnected clusters could be represented as four connected components. To resolve the inconsistency between the visual surface mesh and the underlying physical model, it is necessary to ensure that each newly generated cluster forms a connected component. This can be achieved using the finding connected components algorithm.

Finding connected components (abbreviated as FCC) in an undirected graph is a well-studied problem. Traditional algorithms used to find connected components, such as depth-first search or breadth-first search, are unsuitable for GPU parallel implementation. This study draws inspiration from the work of [22] and their ECL-CC algorithm for finding connected components, proposing a highly parallel FCC algorithm tailored for our cutting scenarios. In the original FCC algorithm proposed by Jaiganesh et al. [22], each vertex v in graph G has an index to identify itself at the initial step. The index is updated to the lowest index of its neighbors until all particles' indices remain unchanged. This approach is only suitable for finding the connected components of a single cluster due to the overlapping between clusters. To find the connected components for all clusters in parallel, we proposed an aggregate finding connected components algorithm (AFCC) to compute the connected components of each cluster concurrently, as demonstrated in Fig. 5. In this 2D example, there are two overlapping clusters, with particles 2 and 3 being the overlapping connected particles. When running a parallel AFCC algorithm, we duplicate the overlapping particles to avoid interleaving between clusters. The cluster index is attached to each particle to guarantee the uniqueness of each particle. For example, particle(1, 3) belongs to cluster 1, and its particle index is 3. After applying the AFCC algorithm to the aggregate graph, the results are the connected components after cluster splitting as shown in the right image of Fig. 5.

2.2.2. Cluster expansion

Note that the AFCC algorithm can only ensure disconnected visual surface meshes are separated in the simulation structure. It does not

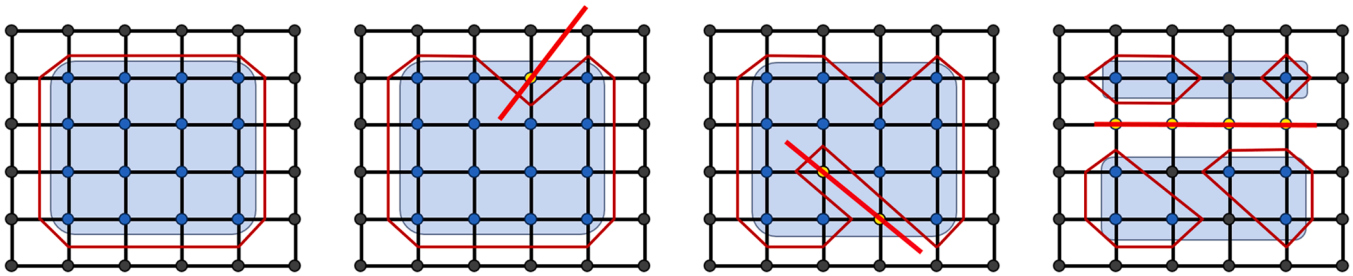


Fig. 3. A 2D example of the inconsistency of the physical and visualization models after three cuts. The first image shows a soft body discretized as solid blue particles on a background grid, and all particles belong to one cluster represented as a blue region. The surface mesh is shown as dark red lines. The black solid circles are just normal grid nodes that don't have physical meaning. The cutting tool is shown as a bright red line in the second and third images. The cut particles are colored yellow. Four separate surface meshes were generated in the last image. However, only two clusters are identified. Both clusters include two disconnected surface meshes generated by the variant of marching cubes [17].

guarantee that the separated clusters in the simulation structure will also be separated on the visual surface meshes. As illustrated in the 2D example Fig. 6, when the overlapping particles between two clusters are removed, a cluster becomes isolated and can move freely (as shown in the middle image of Fig. 6). This causes obvious visual artifacts because the surface mesh enclosing this cluster moves with it. We can tackle this problem by expanding the isolated cluster to cover more particles in its neighbor cluster as depicted in the right image of Fig. 6.

Specifically, after cutting occurs, each cluster is tested to determine if it is isolated. If an isolated cluster has neighboring clusters, we can add a portion of the particles from the neighboring clusters to the isolated cluster. The center of mass is then recomputed using the undeformed positions of these particles. The expansion of clusters can proceed in parallel without the need to consider race conditions.

2.2.3. Cluster merging

Although the inconsistency between the visual surface mesh and the underlying clusters can be resolved by applying our AFCC algorithm, the number of clusters will continue to increase as the number of cuts increases, which would slow down the simulation. Moreover, the number of particles in each cluster would decrease, making the soft tissue appear softer. To address these issues, we propose a merge algorithm to combine small clusters into a single cluster, ensuring that the average size of clusters and the number of soft tissue clusters remain within a reasonable range.

At first, we traverse each cluster and test if the number of particles of the cluster is below a threshold that is small enough to merge with its neighbor clusters. Assume the small cluster that needs to be merged is denoted as G_s. The neighbor clusters of G_s should meet the conditions that an edge e exists whose endpoints belong to the $G_s$and its neighbor cluster, respectively. This condition ensures that the resulting cluster after the merge remains a connected component. On the other hand, the resulting cluster should be a manageable size. It would also result in the divergence of the parallel algorithm. The maximum number of particles in a cluster is usually $1.3 \sim 1.5$ times the maximum number of particles in a cluster during the initial sampling procedure. One possible approach is to select the smallest neighboring cluster using a greedy strategy. However, the parallel scenario introduces the readers-writer problem because several small clusters may select the same target neighboring cluster to merge with.

To avoid the readers-writers problem when merging multiple small clusters with the same target cluster, we modify the number of particles in the target cluster using the atomic compare-and-swap (CAS) operation. The target cluster is then assigned to the small cluster that successfully acquires the lock. We can delay the merging operations and perform them in parallel after all small clusters have found their respective target clusters to merge with.

2.3. Cutting detection

The previous discussion addresses post-cutting processing methods. This section elucidates the conditions under which cutting occurs, specifically the extent of deformation on the soft body surface caused by the surgical blade that is necessary for initiating the cutting process. In traditional Finite Element Method (FEM) cutting approaches, the elements in direct contact with the surgical blade are subdivided without capturing the phenomenon of cutting resulting from the compressive deformation of the object's surface [19]. Furthermore, tetrahedral subdivision operations alter the dimensions of the system matrix, significantly impacting system performance.

We draw inspiration from [23] and use the principal stretching strain of $\mathbf{S}$ in Eq. (2) to determine if the cut occurs. However, the principal stretching strain of cutting (left image in Fig. 7) is the same as uniform stretching (right image in Fig. 7), which would result in many false positive cutting detections during the simulation. To overcome this ambiguity in cutting detection, we propose a distance-based cutting detection method. For each particle $\mathbf{x}_i$ of soft tissue, we compute the maximal neighbor distance di between $\mathbf{x}_i$ and its neighbors. If the maximal neighbor distance d_i exceeds the cutting threshold ε, the particle is removed from the soft tissue and marked as a normal grid node of the background grid. As shown in the first left image of Fig. 4, the left side of the cuboid stretches under gravity but does not create cut effects. The scalpel presses the surface of the cuboid surface until the particle's maximal neighbor distance surpasses the cutting threshold. As the surgical blade descends, the entire object is incrementally cut and separated into two distinct pieces.

3. Results

All experiments in this chapter were conducted in the following hardware environment: Intel i7 10700F CPU, Nvidia RTX 2060 GPU, and 16GB memory. The deformation and dissection simulations were all implemented using C++ and CUDA and rendered with OpenGL. In the following sections, we present the simulation results and performance data.

3.1. Soft tissue dissection

To demonstrate the complete process of the scalpel before and after cutting and the squeezing effect on the object surface when cutting occurs, we performed a cutting operation on a regular block-shaped flexible body from above, as shown in Fig. 4. The second image from the left shows the scalpel exerting forces on the soft body surface, creating a dent where the scalpel's blade makes contact. As the scalpel moved downward, the object was gradually cut and separated into two parts.

We also applied our method to simulate liver resection and nephrectomy. As shown in Fig. 11, our method can handle the cutting of

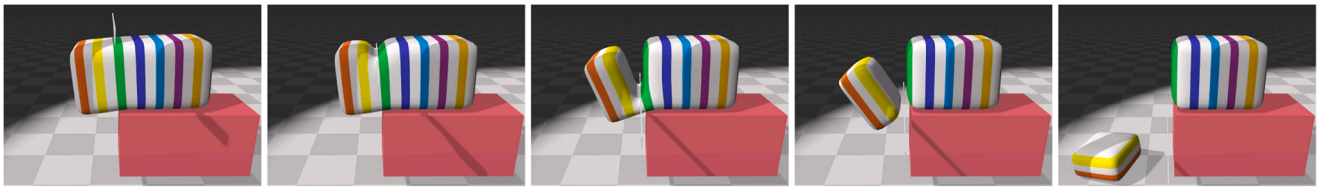


Fig. 4. A cuboid soft body rests on a red platform. Moving from the left image to the right, a scalpel descends from above, cutting the soft body into two separate parts.

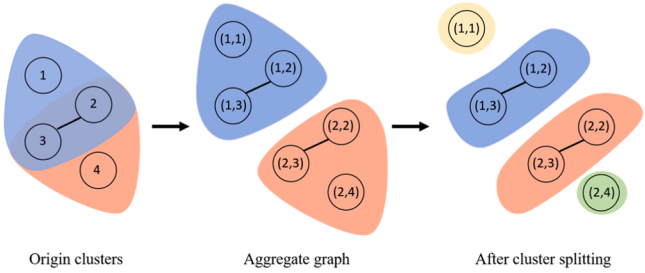


Fig. 5. Aggregrate graph of clusters.

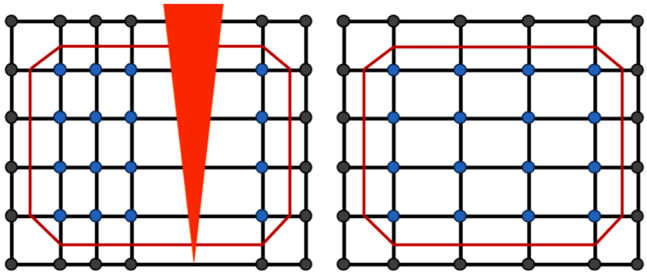


Fig. 7. Cutting and stretch have the same principal strain.

the liver, which has a complex surface model. To demonstrate the superiority of the cutting algorithm, we performed complex cuts on the kidney. The cutting path in Fig. 11 was a curve. In Fig. 12, a part of the kidney was removed by dividing it into two parts using cuts from top to bottom and from left to right. No ghost forces attract the cut part, even when it was visually separated, allowing the cut part to move freely.

To demonstrate the necessity of expanding clusters after dissection, we compared the visual effects of dissection with expansion enabled and disabled, as shown in Fig. 6. During the dissection procedure with a continuous irregular cut path, some isolated clusters moved freely and appeared as debris when expansion was disabled, which is not the expected effect, as demonstrated in Fig. 13a. With expansion enabled, there are no isolated clusters, as evident in Fig. 13b.

3.2. Comparison with CPIC

We conducted experiments to compare our method with the CPIC method [9]. The CPIC method uses the CDF(Colored Distance Fields) method to resolve the collision between the soft and rigid bodies. A background grid is used to solve the equation and motion and detect collisions. The physical meaning of points in CPIC is Gauss quadrature nodes for integration. The sampling density is denser than the shape-matching method to obtain resealable deformation effects. We measured the time overhead per frame for the simulation phase of these two methods during the cutting. The cutting results are shown in Fig. 14. The CPIC scene includes 80 × 80 × 80 background grid and 5000 particles. The timestep is one millisecond and four substeps. Our method only needs 1000 particles with 65 clusters, and the time step is 16 milliseconds and four substeps. The performance of our method is relatively close to that of the MPM method, as shown in Fig. 9. Let's focus on the horizontal axis of the figure, specifically within the range of frames 110 to 125. In this range, a clear observation can be made: as the cutting operation progresses, the CPIC method demonstrates higher per-frame processing times compared to our approach. We found that cutting the soft body using the CPIC method requires slow and controlled scalpel movement. Otherwise, the particles still stick together after cutting because of the numerical viscosity effects of the MPM method. In the third image from the left in Fig. 14, the CPIP method suffers from the sticking problem where ghost force pulls the right part towards the left. Apart from that, it can only support simple cutting planes and cutting paths. In contrast, our cutting method is more flexible and supports more complex cutting paths.

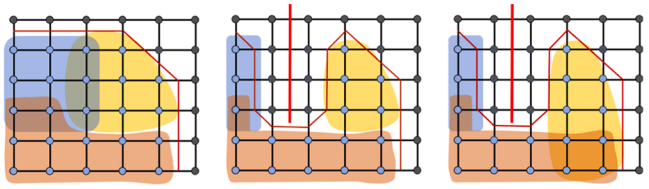


Fig. 6. Isolated cluster after cutting and cluster expansion strategy: In the left image, three overlapping clusters are colored blue, yellow, and orange. The red line represents the surface mesh. In the middle image, the overlapping particles between the yellow and blue clusters are removed by the scalpel, which is depicted as the red line. Although the surface mesh is correct, the yellow cluster can move freely, causing visual artifacts. In the right image, we expand the yellow cluster to include particles from the orange cluster, limiting the movement of the yellow cluster.

3.3. Performance

The entire dissection procedure involves several stages, as shown in Figs. 4, 8, 11, we evaluated the time cost of each stage in several scenes (Table 1). All experiments were conducted and measured with the same configuration ten times to calculate the average time. The particle and cluster columns in Table 1 show the number of particles and clusters in each scene. The constraints solver stage, which uses the Flex library to solve shape-matching constraints, is the most costly stage. The cutting detection stage includes the time for removing particles as both of them cost little time. The time per frame is the total time a frame takes, which is smaller than the sum of all stages because some stages are parallelized using multiple threads. These times do not include rendering. The performance data in Table 1 demonstrated that our method can reach near real-time in complex model detection scenarios.

To assess the impact of particle and cluster quantities corresponding to soft body sampling on performance, we conducted a comparative analysis of the performance variations in the continuous cutting paths of kidney resection under different sampling rates, considering various particle and cluster number scales. The cutting paths and physical parameter settings were consistently maintained throughout the experiments. As depicted in Table 2, no linear relationship is observed between time expenditure and the scales of particles or clusters. When both particle and cluster quantities are doubled (second and fourth rows), the time expenditure increases by only half, indicating the

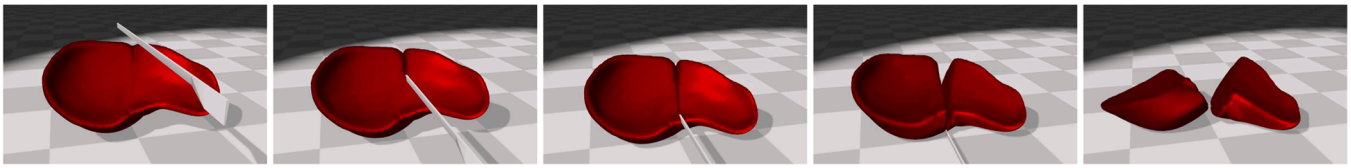


Fig. 8. The liver is placed on the ground. From left to right, the scalpel moves from above to below, and the liver is cut into two separate parts.

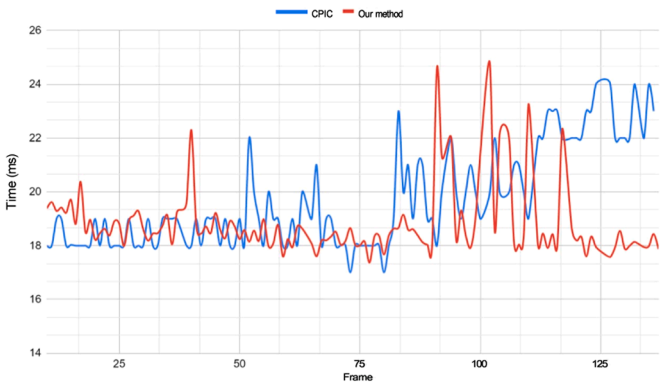


Fig. 9. Performance comparison between MPM and our method.

commendable scalability of the proposed cutting method in this study.

The number of clusters during the cutting process directly corresponds to the number of constraints, as illustrated in Table 1. Constraint solving represents the most time-consuming stage within the entire simulation workflow. Therefore, we examined the variation in the number of clusters in the context of continuous cutting paths for kidney resection, as shown in Fig. 10. The cutting process commences at the 35th frame, leading to an increase in the number of clusters. As the cutting progresses, the cluster count exhibits oscillations, with a noticeable decrease in certain frames, such as frame 65. This reduction is attributed to the particle count within the clusters reaching a certain threshold, prompting the fusion of multiple clusters. Although an overall increasing trend in cluster count is observed throughout the cutting, our algorithm effectively restrains the growth of cluster quantity. By the conclusion of the cutting process at frame 95, the cluster count has only increased by 1%.

In addition, we compared the performance of our method of kidney dissection experiment with those of Pan et al. [14], Wu et al. [24], Bert et al. [12], and CPIC [9]. As shown in Table 3, only the cutting method proposed by Bert et al. slightly outperforms ours in terms of performance. However, the main difference compared to our research method is that Bert et al.'s work does not elaborate on the cluster separation process in detail. From the actual effect, their soft body cutting seems to directly delete clusters without dividing them. Thus, the cutting effect in their paper is more like a "burning" process, where the soft body loses a large amount of volume. In contrast, the method proposed in this paper supports more refined and regular cutting operations, which can reduce the disappearance of volume. Although our performance metrics do not surpass their method, their cutting effect may produce ghost forces and fragment artifacts, which is precisely what this paper strives to improve.

4. Discussion

This paper proposed a general soft tissue dissection framework based on the shape-matching simulation method. Two novel algorithms were introduced to address topological inconsistencies between reconstructed surface meshes and clusters in the physical model due to complex cutting paths during the soft body cutting simulation process. Firstly, the

Table 2
The temporal expenditure associated with varying quantities of particles and clusters in the context of Kidney resection scenes involving continuous cutting paths in Fig. 12.

Num. of particles	Num. of clusters	Time per frame (ms)
37,704	301	42.05
48,385	383	46.99
63,649	506	48.34
101,096	795	59.75

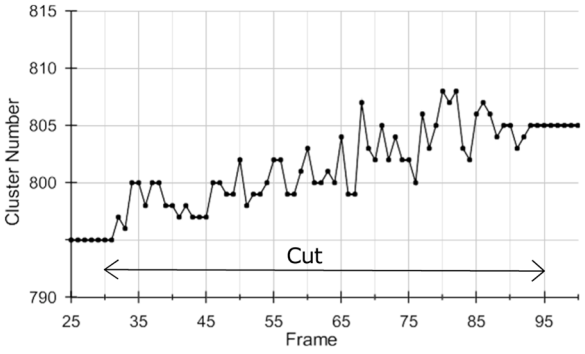


Fig. 10. The figure shows how the cluster number varies during a cut. The cut starts at frame 30 and ends at frame 95. In the Kidney scene, the cluster number fluctuates up and down due to splitting clusters and merging clusters. After one cut, the cluster number increases by only 1%.

Table 1
The time cost of each stage per frame in several scenes.

Time (ms) \ Scene / Stage	Cuboid		Liver		Kidney	
	Particles	Clusters	Particles	Clusters	Particles	Clusters
	2000	124	70318	543	101096	795
Constraints solver	16.99		26.34		31.79	
Cutting detection	0.03		0.33		0.41	
Marching cubes	5.97		8.67		8.03	
Cluster splitting	3.98		6.75		7.16	
cluster merging	1.98		2.47		2.9	
Cluster expansion	3.37		6.35		10.77	
Time per frame	26.92		50.19		59.75	

Table 3
Comparison between our simulation and previous works.

Time (ms) \ Methods / No. of particls	Pan's methods [14]	Wu's methods [24]	Berndt's methods [17]	CPIC [9]	Our methods
12,800	22.1	70.2	13.6	22.4	20.2
37,704	56.3	310.6	38.9	45.6	42.1
48,385	64.4	601.1	46.0	52.0	47.0
101,096	105.7	2038.6	56.2	70.2	59.8

AFCC algorithm was introduced to compute the connectivity subgraph corresponding to each cluster, thereby reconstructing new shape-matching constraints based on these subgraphs. This ensures the absence of ghost forces between visually separated meshes. The four experiments are shown in Figs. 4, 8, 11, 12 demonstrated that under various types of cutting paths, the absence of ghost forces between objects is ensured after the cutting process.

Secondly, the paper introduced a cluster expansion algorithm to mitigate the emergence of isolated clusters and unrealistic cutting fragments during the cutting process. This algorithm enlarges the particle count in isolated clusters by incorporating particles from neighboring clusters, thereby constraining the movement of isolated clusters. As shown in Fig. 6, there are no freely moving clusters after applying cluster expansion.

In response to the continuously increasing number of clusters during the cutting process, a cluster merging strategy was proposed to reduce the overall cluster count by merging smaller clusters within a cluster and neighboring clusters. To address data read-write conflicts on the GPU during merging, a CAS strategy was employed to modify the total particle count of the merged clusters. This shift of the most time-consuming stage of merging outside atomic operations enhances the efficiency of parallel cluster merging. As shown in Fig. 10, the cluster number increases within an acceptable range during soft tissue dissection simulation, thanks to the proposed merging strategy.

Upon careful examination of the time cost at various stages listed in Table 2, it is noteworthy that the combined time cost of the cutting detection, cutting response, and skinning phases are comparable to the shape-matching constraint-solving stage. Most of the time cost in the shape-matching constraint-solving stage stems from the synchronization overhead between the GPU and CPU. This portion of the cost remains significant, irrespective of the complexity of the problem, even in scenarios involving simple boxes. However, it is pertinent to observe that this overhead does not significantly increase even in complex scenes because it is fixed. Furthermore, the time spent on cutting detection is extremely short, amounting to less than one millisecond, even in intricate scenes.

Table 2 reveals that cluster splitting and cluster expansion become performance bottlenecks. Even AFCC algorithms optimized for GPUs which have a time complexity of $O(E)$ and a theoretical 1.x speedup when the number of processors is less than$O(V)$, there is still a high lower bound on the algorithm's execution time. The performance bottleneck associated with expanding clusters may be attributed to insufficient parallelism. The current implementation of the cluster expansion algorithm relies on CUDA dynamic parallelism. Although in theory, this implementation can achieve intra-cluster and inter-cluster parallelism for maximum efficiency, practical execution may fall short of this theoretical limit due to the influence of GPU scheduling strategies on dynamic parallelism.

Observing the time per frame is consistently smaller than the sum of the time durations for each stage. This is because certain stages run in parallel. The most significant parallelization occurs in the skinning (marching cubes) and cluster update stages (comprising the sum of cluster splitting, cluster merging, and cluster expansion). The prerequisite for this parallelization is the independence of these two parts, meaning they do not contend for the same resources, including hardware and data. Concerning hardware resources, although both parts predominantly rely on GPU computing, they require synchronization with the CPU. Additionally, some steps in both parts cannot achieve maximum GPU utilization. Therefore, while there is competition for GPU resources, this competition is not persistent throughout the execution. Regarding data resources, they share only the connectivity information in the background grid, and both parts perform read-only operations without any write operations. Consequently, there is no contention for data resources between these two parts. In summary, the multithreaded parallelization of these two parts contributes to improved algorithm execution efficiency. Indeed, the data in Table 1 also suggests that this parallel optimization typically results in a performance improvement ranging from 5% to 15%.

Regarding the performance bottleneck of the cluster update stage, the parallel algorithm proposed by Jaiganesh et al. [22] was employed, achieving a higher level of parallelism through the aggregation graph method. However, the implementation in this study does not fully leverage the hardware features of the GPU. Techniques like read-write merging was not applied, resulting in an I/O bottleneck in memory reads and writes. Despite a sufficiently high kernel utilization, the register utilization is low, possibly due to overly fine-grained kernel partitioning. Moreover, the synchronization frequency between the CPU and GPU remains high.

5. Conclusion

This paper focuses on the real-time simulation of soft body cutting in virtual surgery, exploring how to achieve soft body cutting using the shape-matching method for soft body modeling. Localized and

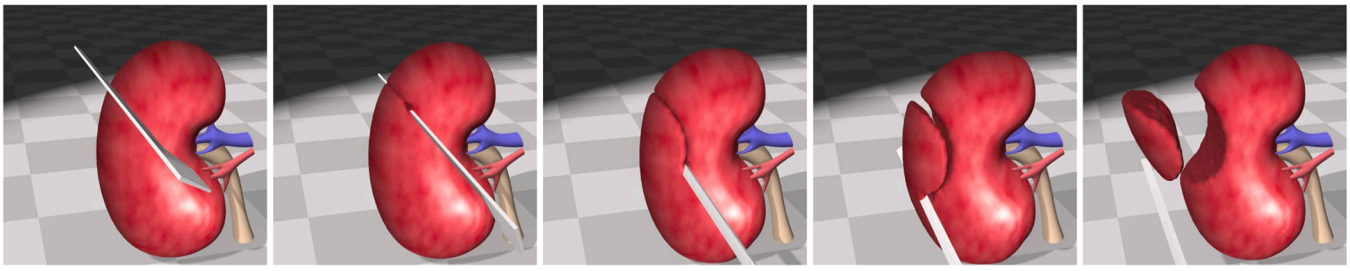


Fig. 11. The kidney is cut along an arc path using a scalpel.

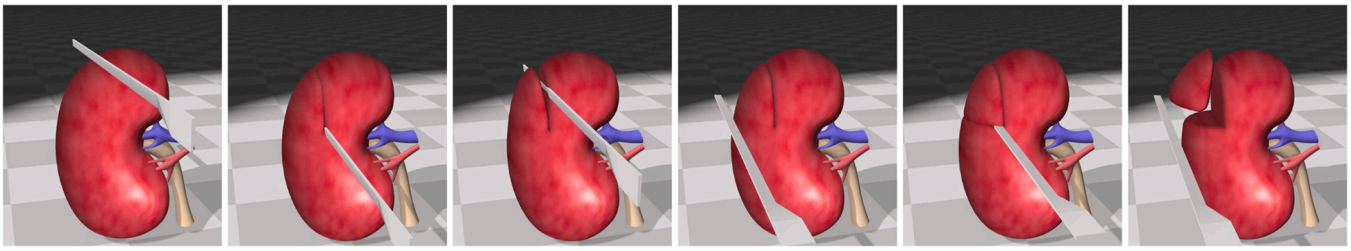
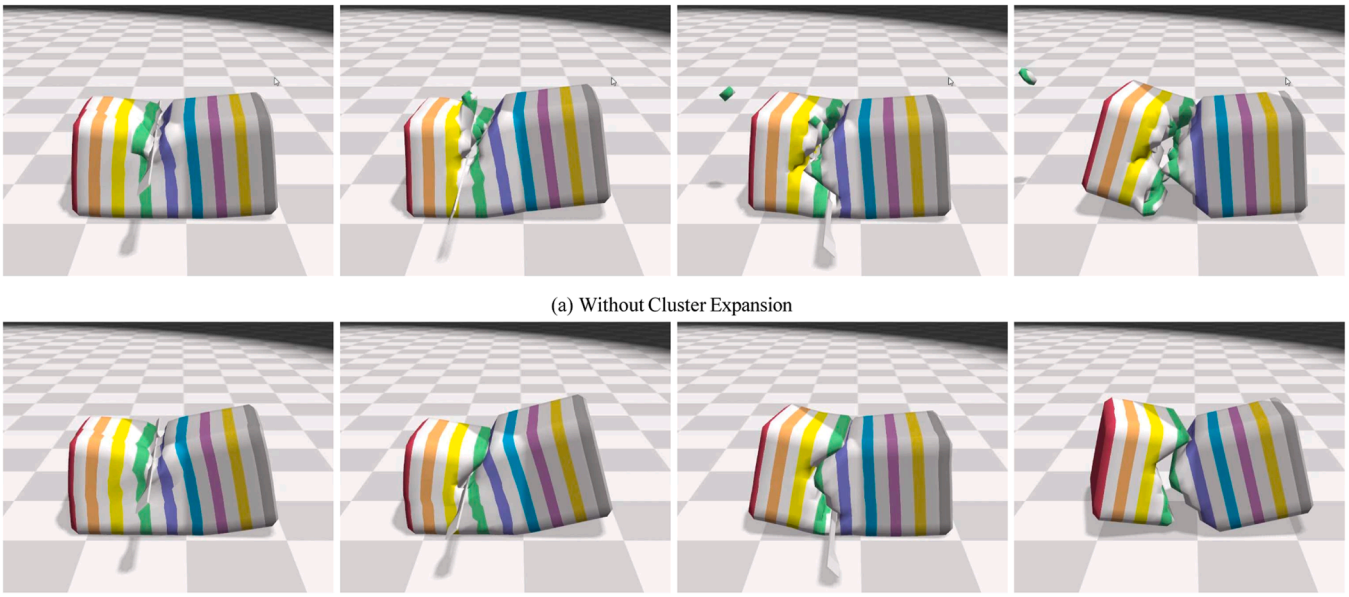


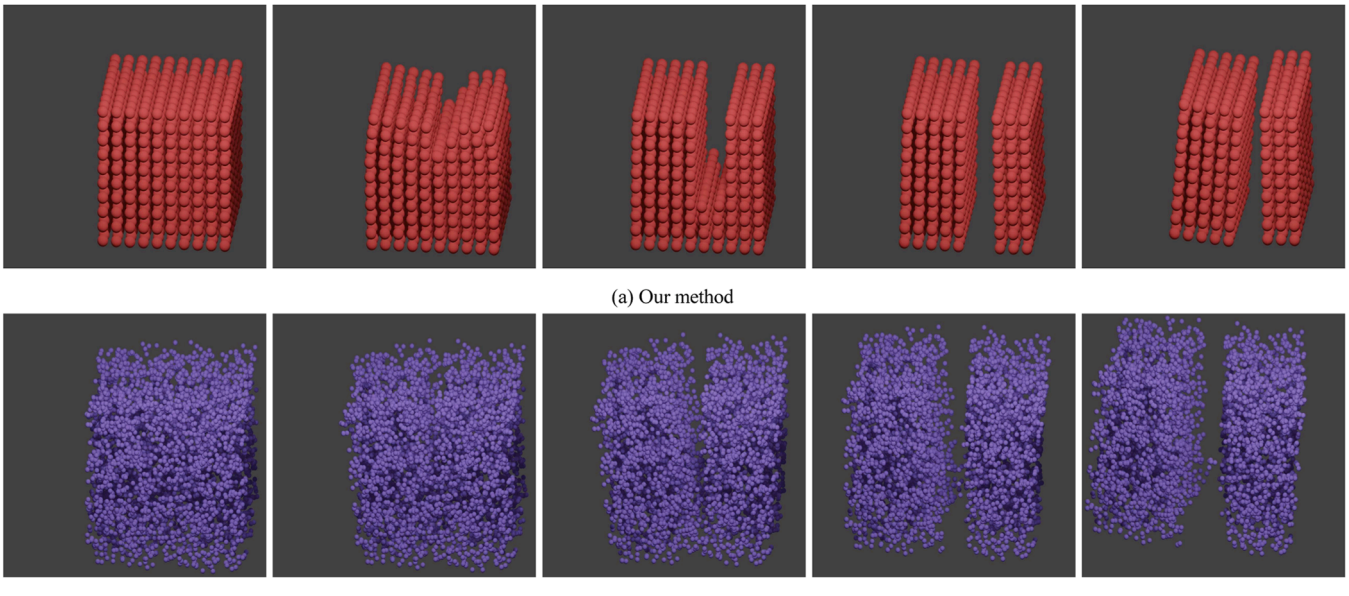
Fig. 12. Discontinuous resection of the kidney. The three left images show that the kidney is cut vertically, which does not cause the object to separate. The three right images show that after a subsequent horizontal cut, a part of the kidney is cut off.



(a) Without Cluster Expansion

(b) With Cluster Expansion

Fig. 13. The *Isolated Cluster* problem can become very serious without the cluster expansion procedure. As shown in Fig. 13a, lots of fragments appear during cutting, which are isolated clusters.



(a) Our method

(b) CPIC [7]

Fig. 14. The comparison experiment between our method and CPIC. The experiment is only performed on the simulation without rendering for a clear comparison. The spheres in the figures are the particles in both methods.

discontinuous cuts can cause mismatch issues between the surface mesh and the low-level physics model, resulting in ghost forces and fragment artifacts. To address these problems, a graph representation of clusters was introduced, and the connectivity of a background grid in the skinning stage was utilized to define the connectivity within clusters. Building on this foundation, a high-level parallel implementation using the AFCC algorithm was provided. Additionally, the paper proposed the cluster merging method to address the issue of excessive growth in the number of clusters caused by the cluster splitting method. The challenge of parallel cluster merging was solved by using CAS locks to establish critical sections. To address the issue of isolated clusters caused by randomness in the cluster sampling process, the paper introduced the cluster expansion method, which involved merging neighboring particles. In practical applications, the proposed method results in only a 1% increase in the number of clusters before and after a complete cutting process. Experiments validate that this method effectively eliminates isolated clusters, ensuring smooth cutting surfaces.

The experiments evaluated the performance and effectiveness of the proposed method in scenarios involving liver and kidney resection surgeries. The results demonstrated that the method achieves real-time performance, reaching 30 frames per second even in complex scenes. The proposed method produces smooth cutting surfaces, eliminates ghost forces and fragments, precisely matches cutting paths, and generates realistic soft deformations during cutting. The method surpasses the material point method in long-term simulation performance, faster internal force propagation, and immunity to ghost force and fragments artifacts, making it more suitable for real-time interactive simulation scenarios such as virtual surgery, where fine cutting is essential.

In the future, the soft tissue dissection simulation will be tested and improved by medical residents. Optimizing the implementation to better utilize GPU hardware features is recommended to further enhance performance. This includes applying read-write merging techniques to alleviate the memory read-write bottleneck, adjusting kernel partitioning to improve register utilization, and reducing synchronization frequency between the CPU and GPU. These optimization measures are expected to significantly reduce the computational overhead in the cluster division and cluster expansion stages, thereby achieving better real-time performance in complex scenarios.

Declaration

During the preparation of this work the author(s) used Grammarly, ChatGPT4, Claude-3-Opus, and Kimi in order to improve readability and language. After using this tool/service, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the content of the publication.

CRediT authorship contribution statement

Peng Yu: Writing – original draft, Visualization, Validation, Software. **Zhiyuan Zhao:** Visualization, Validation, Software, Resources, Methodology. **Ruiqi Wang:** Visualization, Software. **Junjun Pan:** Writing – review & editing, Project administration, Funding acquisition.

Declaration of competing interest

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

Acknowledgments

This work was supported in part by the National Key R&D Program of China (no. 2023YFC3604505), the Postdoctoral Fellowship Program of CPSF under grant number GZC20233375, Natural Science Foundation of China (no. U20A20195, 62272017, 62172437), Beijing Natural Science Foundation (L232065), CAS Interdisciplinary Project (JCTD-2020-11).

Supplementary materials

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.cmpb.2024.108171.

References

[1] M. Higgins, C. Madan, R. Patel, Development and decay of procedural skills in surgery: a systematic review of the effectiveness of simulation-based medical education interventions, Surgeon 19 (4) (2021) e67–e77, https://doi.org/10.1016/j.surge.2020.07.013.

[2] P. Yu, J. Pan, Z. Wang, Y. Shen, J. Li, A. Hao, H. Wang, Quantitative influence and performance analysis of virtual reality laparoscopic surgical training system, BMC Med. Educ. 22 (1) (2022) 1–10, https://doi.org/10.1186/s12909-022-03150-y.

[3] J.J. Pan, J. Chang, X. Yang, J.J. Zhang, T. Qureshi, R. Howell, T. Hickish, Graphic and haptic simulation system for virtual laparoscopic rectum surgery, Int. J. Med. Robot. Comput. Assist. Surg. 7 (3) (2011) 304–317, https://doi.org/10.1002/rcs.399.

[4] H. Courtecuisse, J. Allard, P. Kerfriden, S.P. Bordas, S. Cotin, C. Duriez, Real-time simulation of contact and cutting of heterogeneous soft-tissues, Med. Image Anal. 18 (2) (2014) 394–410, https://doi.org/10.1016/j.media.2013.11.001.

[5] H.P. Bui, S. Tomar, S.P. Bordas, Corotational cut finite element method for real-time surgical simulation: application to needle insertion simulation, Comput. Method. Appl. Mech. Eng. 345 (2019) 183–211, https://doi.org/10.1016/j.cma.2018.10.023.

[6] J. Wang, S. Jia, G. Wang, Z. Pan, X. Yu, An improved CPU–GPU parallel framework for real-time interactive cutting simulation of de- formable objects, Comput. Graph. 114 (2023) 59–72, https://doi.org/10.1016/j.cag.2023.05.013.

[7] E. Heiden, M. Macklin, Y.S. Narang, D. Fox, A. Garg, F. Ramos, DiSECt: a differentiable simulation engine for autonomous robotic cutting, in: *Proceedings of Robotics: Science and Systems*. Virtual, 2021, https://doi.org/10.15607/RSS.2021.XVII.067.

[8] A. Stomakhin, C. Schroeder, L. Chai, J. Teran, and A. Selle. A material point method for snow simulation. 32(4), jul 2013. doi:10.1145/2461912.2461948.

[9] Y. Hu, Y. Fang, Z. Ge, Z. Qu, Y. Zhu, A. Pradhana, C. Jiang, A moving least squares material point method with displacement discontinuity and two-way rigid body coupling, ACM Trans. Graph. 37 (4) (2018), https://doi.org/10.1145/3197517.3201293.

[10] J. Wolper, Y. Chen, M. Li, Y. Fang, Z. Qu, J. Lu, M. Cheng, C. Jiang, Anisompm: animating anisotropic damage mechanics, ACM Trans. Graph. 39 (4) (2020), https://doi.org/10.1145/3386569.3392428.

[11] C. Jiang, C. Schroeder, J. Teran, A. Stomakhin, A. Selle, The material point method for simulating continuum materials, in: *ACM SIGGRAPH 2016 Courses*, SIGGRAPH '16. Association for Computing Machinery, New York, NY, USA, 2016, https://doi.org/10.1145/2897826. 2927348.

[12] J. Bender, M. Müller, M. Macklin, A survey on position based dynamics, 2017, in: *Proceedings of the European Association for Computer Graphics: Tutorials*, EG '17, Goslar, DEU, Eurographics Association, 2017, https://doi.org/10.2312/egt.20171034.

[13] F. Liu, Z. Li, Y. Han, J. Lu, F. Richter, M.C. Yip, Real-to-sim registration of deformable soft tissue with position-based dynamics for surgical robot autonomy, in: 2021 IEEE International Conference on Robotics and Automation (ICRA), IEEE Press, 2021, pp. 12328–12334, https://doi.org/10.1109/ICRA48506.2021.9561177.

[14] J. Pan, J. Bai, X. Zhao, A. Hao, H. Qin, Real-time haptic manipulation and cutting of hybrid soft tissue models by extended position-based dynamics, Comput. Animat. Virt. World. 26 (3–4) (2015) 321–335, https://doi.org/10.1002/cav.1655.

[15] J. Pan, S. Yan, H. Qin, A. Hao, Real-time dissection of organs via hybrid coupling of geometric metaballs and physics-centric mesh-free method, Vis. Comput. 34 (1) (2018) 105–116, https://doi.org/10.1007/s00371-016-1317-x.

[16] J. Li, T. Liu, L. Kavan, B. Chen, Interactive cutting and tearing in projective dynamics with progressive cholesky updates, ACM Trans. Graph. 40 (6) (2021), https://doi.org/10.1145/3478513.3480505.

[17] I. Berndt, R. Torchelsen, A. Maciel, Efficient surgical cutting with position-based dynamics, IEEE Comput. Graph. Appl. 37 (3) (2017) 24–31, https://doi.org/10.1109/MCG.2017.45.

[18] M. Macklin, M. Müller, N. Chentanez, T.-Y. Kim, Unified particle physics for real-time applications, ACM Transact. Graph. (TOG) 33 (4) (2014) 104, https://doi.org/10.1145/2601097.2601152.

[19] M. Kamarianakis, A. Protopsaltis, D. Angelis, M. Tamiolakis, and G. Papagiannakis. Progressive tearing and cutting of soft-bodies in high-performance virtual reality. *arXiv preprint arXiv:2209.08531*, 2022.

[20] M. Müller, B. Heidelberger, M. Teschner, M. Gross, Mesh- less deformations based on shape matching, ACM Trans. Graph. 24 (3) (2005) 471–478, https://doi.org/10.1145/1073204.1073216.

[21] M. Müler, M. Macklin, N. Chentanez, S. Jeschke, Physically based shape matching, Comput. Graph. Forum 41 (8) (2022) 1–7, https://doi.org/10.1111/cgf.14618.

[22] J. Jaiganesh, M. Burtscher, A high-performance connected components implementation for gpus, in: *Proceedings of the 27th International Symposium on High-Performance Parallel and Distributed Computing*, HPDC '18, York, NY, USA,

Association for Computing Machinery, New, 2018, pp. 92–104, https://doi.org/10.1145/3208040.3208041.

[23] X. Li, Y. Cao, M. Li, Y. Yang, C. Schroeder, C. Jiang, Plasticitynet: learning to simulate metal, sand, and snow for optimization time integration, in: S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, A. Oh (Eds.), Advances in Neural Information Processing Systems, Curran Associates, Inc., 2022, pp. 27783–27796, vol. 35.

[24] J. Wu, R. Westermann, C. Dick, A survey of physically based simulation of cuts in deformable bodies, Comput. Graph. Forum 34 (6) (2015) 161–187, https://doi.org/10.1111/cgf.12528.