2-27-2023

Split and Join: An Efficient Approach for Simulating Stapled Intestinal Anastomosis in Virtual Reality

Di Qi

Suvranu De

Split and Join: An Efficient Approach for Simulating Stapled Intestinal Anastomosis in Virtual Reality

Received: 15 September 2022 | Revised: 12 February 2023 | Accepted: 14 February 2023

DOI: 10.1002/cav.2151

RESEARCH ARTICLE

WILEY

Split and join: An efficient approach for simulating stapled intestinal anastomosis in virtual reality

Di Qi[1] | **Suvranu De**[2]

[1]Dale E. and Sara Ann Fowler School of Engineering, Chapman University, Orange, California, USA

[2]College of Engineering, Florida A&M University – Florida State University, Tallahassee, Florida, USA

Correspondence
Di Qi, Dale E. and Sara Ann Fowler School of Engineering, Chapman University, Orange, CA, USA.
Email: dqi@chapman.edu

Funding information
National Institutes of Health (US), Grant/Award Number: R01EB005807

Abstract

Colorectal cancer is a life-threatening disease. It is the second leading cause of cancer-related deaths in the United States. Stapled anastomosis is a rapid treatment for colorectal cancer and other intestinal diseases and has become an integral part of routine surgical practice. However, to the best of our knowledge, there is no existing work simulating intestinal anastomosis that often involves sophisticated soft tissue manipulations such as cutting and stitching. In this paper, for the first time, we propose a novel *split and join* approach to simulate a side-to-side stapled intestinal anastomosis in virtual reality. We mimic the intestine model using a new hybrid representation—a grid-linked particles model for physics simulation and a surface mesh for rendering. The proposed *split and join* operations handle the updates of both the grid-linked particles model and the surface mesh during the anastomosis procedure. The simulation results demonstrate the feasibility of the proposed approach in simulating intestine models and the side-to-side anastomosis operation.

KEYWORDS

cutting simulation, intestine simulation, stapled intestinal anastomosis, virtual reality

1 | INTRODUCTION

Colorectal cancer is a life-threatening disease. It is the second leading cause of cancer-related deaths in the United States.[1] Surgical techniques such as anastomosis used in the intestinal resection and reconnection are essential strategies for surgical treatment of colorectal cancer and other intestinal diseases.[2] In particular, stapled anastomosis has become an integral part of routine surgical practice.[3] In side-to-side stapled anastomosis, the linear stapler is inserted into two intestinal parts and aligned by the antimesenteric border. During closing and firing the linear stapler, the sidewalls of both intestinal parts are simultaneously split and then joined together to form a new continuous channel. However, due to the complexity of the intestinal anatomy, performing stapled anastomosis without leakage is a highly skillful job,[2] which requires effective training mechanisms. This work directly addresses it by developing a virtual reality-based simulator for side-to-side stapled intestinal anastomosis.

With the rapid advancement of computer power and virtual reality (VR) technologies, there is an increasingly growing interest in adopting VR simulators as an alternative surgical training approach. VR simulation systems allow the trainee to learn the surgical techniques and practice their skills on computer-simulated virtual patients in a safe virtual environment. There are many market-available computer-assisted surgical simulators[4,5] that provide training for laparoscopic

surgeries. However, most of them aim to simulate laparoscopic surgical procedures with limited tissue manipulation. To the best of our knowledge, there is no existing work that simulates intestinal anastomosis surgical procedures that involve sophisticated soft tissue manipulations such as cutting and stitching. Due to the complexity of the delicate structure of the intestine, it is challenging to plausibly simulate its physics behavior, especially when large deformation and topology changes for cutting and stitching are involved.

For the first time, we propose a novel *split and join* approach to simulate side-to-side stapled intestinal anastomosis surgical procedure in virtual reality. We simulate the intestine model using a new representation—grid-linked particles for physics simulation coupled with a surface mesh for rendering. The proposed *split and join* operations handle topological modifications and physics updates of the grid-linked particles models and the skinned surface mesh during anastomosis.

The remainder of the paper is organized as follows. The state-of-the-art that is related to this paper is reviewed in Section 2. Intestine simulation with grid-linked particles is introduced in Section 3. Section 4 presents the *split and join* operations for simulating side-to-side intestinal anastomosis, by updating both grid-linked particles models and skinned meshes of two intestine models. Finally, we demonstrate experimental results of the proposed *split and join* operations for side-to-side anastomosis implemented in a game engine in Section 5, followed by discussion and concluding remarks in Section 6.

2 | RELATED WORK

Simulating accurate deformation characteristics of soft tissue is essential for a realistic virtual simulation of surgical procedures. However, developing real-time computation methods that faithfully mimic the physical behavior of virtual organs and allow real-time visual and haptic interactions, and interactive geometric and topological modifications of soft tissues as a result of surgical operations is a challenging task. This section reviews the simulation methods specifically for intestine anatomy and their surgical operations in virtual surgical simulation. For a more comprehensive review of soft tissue simulation in virtual surgery, we refer the reader to a recent survey.[6]

Finite element methods (FEM)[7,8] and mass-spring systems (MSS)[9,10] are the most commonly employed approaches for soft tissue simulation. In the FEM,[11] the soft tissue model is discretized using a mesh and the governing partial differential equations are solved on mesh. Although FEM can achieve high simulation accuracy, it tends to be more computationally demanding. Real-time applications such as surgical simulation tend to use techniques such as MSS[9] or position-based dynamics (PBD)[12,13] to achieve a trade-off between accuracy and computational efficiency. Unlike FEM, MSS uses a network of mass nodes connected by elastic springs to mimic the physical behavior of soft tissues; deformation is generated by applying force to a node and solving algebraic equations derived from Hooke's Law and Newton's second law of motion. Akin to MSS, PBD simulates a soft body with a set of mass nodes (particles), where the interactions between the nodes are modeled as a constraint function. The deformed configuration is obtained by solving a constrained optimization problem.

The intestine as a long, tube-like slender bio-tissue suggests that it needs to be modeled differently from other solid human organs such as the liver. Most of the previous intestine simulation approaches approximate its shape using a string of spheres attached to the centerline of the intestine. Chang et al.[14] proposed the beads-on-string model, where the centerline deformation of the intestine is modeled as a Cosserat Rod[15]; the spheres are used to handle collision detection and map the deformation of the centerline to the associated skinning mesh presenting detailed geometry of the intestine. Frances et al.[16] presented a layered model, which uses a spline formularized by Lagrangian dynamics to model the centerline, and the intestine's surface is visualized by an implicit surface generated using the marching cubes method.[17] However, since the spheres are rigid and not deformable, and their motions only follow the centerline deformation, beads-on-string-like methods do not support cutting or suturing operations occurring on the intestine's surface, and the intestine's cross-sections cannot be deformed either.

There is another category of approaches that distributes particles (masses) on the intestine's surface and uses either springs or constraints to connect neighboring particles. The displacement of those particles is used to deform the visual mesh when simulating soft tissue deformation. Pan et al.[18] proposed a multi-layer mass-spring system to model the rectum connected with multiple layers of soft tissues surrounding it. Dissecting the outer layer of soft tissue from the rectum is simulated by removing the spring between the mass nodes. Qi et al.[19] introduced an interactive suturing approach based on a tube-shaped soft tissue modeled by PBD for surgical suturing training. Although particle-like methods support surgical operations, the existing techniques do not support cutting and stitching at the same time, which are required by side-to-side intestinal anastomosis.

In this paper, we present grid-linked particles, a new particle-based representation to simulate the physics behavior of the intestine. For simplicity, we utilize MSS to simulate the intestine in this work, with each particle representing a mass point and each link implementing as a damped spring. We propose two novel operations—*split and join*, effectively updating both grid-linked particles and the visual meshes of the intestines when simulating stapled side-to-side anastomosis procedure in virtual reality.

3 | INTESTINE SIMULATION WITH GRID-LINKED PARTICLES

In this work, we propose a new particle-based representation, grid-linked particles, to simulate the physics behavior of the intestine in real time. Considering the intestine is a tube-like structure, we model its shape using a sequence of circular sections (layers) evenly distributed along the medial axis of the intestine with particles distributed along each layer, see Figure 1a. The visual mesh of the intestine is associated with the physics model for rendering. In the following, we introduce the representation of grid-linked particles in Section 3.1, and the generation of a grid-linked particles model is elaborated in Section 3.2. The visual mesh skinning is discussed in Section 3.3.

3.1 | Grid-linked particles representation

In a grid-linked particle representation shown in Figure 1a, particles are uniformly distributed in layers and each particle is connected to its neighboring particles with damped springs, called *links*. The data structure of the grid-link particle model is a $L \times C$ grid, where L and C denote the number of layers and the number of particles per layer, respectively.

3.1.1 | Particles

Like a regular particle system, each particle of the grid-linked particles has mass, position, and velocity and is subjected to both internal and external forces. As shown in Figure 2a, particles are sampled in layers and distributed in counterclockwise order within each layer. Each particle is expressed as $p_{i,j}$, where $i \in [0, L-1]$ is the *layer index* and $j \in [0, C-1]$ represents the *particle index* within each layer. For example, particle $p_{0,7}$ is the eighth particle from the first particle on the bottom in layer 0 in counterclockwise order.

3.1.2 | Links

We implement each link as a damped spring connecting two particles. However, links can also be implemented as constraints such as the ones used in PBD. There are two types of links depending on whether the two particles it connects with belong to the same layer or not, see Figure 2a:

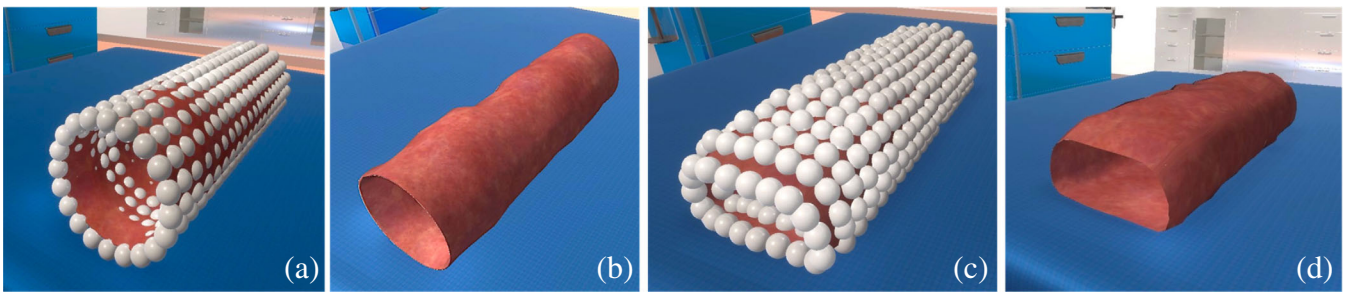


FIGURE 1 Intestine simulation with grid-linked particles: (a) an intestine model is represented by a hybrid structure—a surface mesh for visualization and a grid-linked particles model for physics; (b) the input mesh of the intestine before deformation; (c) deformed intestine with grid-linked particles; (d) visual mesh skinning.

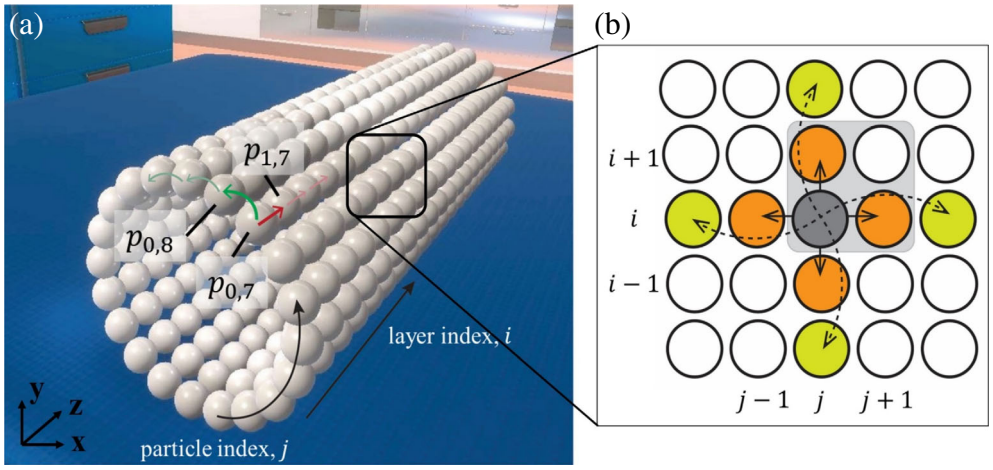


FIGURE 2 Grid-linked particles representation. (a) Particles are uniformly distributed in layers with each particle connected to its neighboring particles through two types of *links*—*in-layer links* (curved arrows in green) and *cross-layer links* (straight arrows in red); each particle $p_{i,j}$ can be access by its layer index i and particle index j along each layer. (b) For each particle, its first-ring neighborhood is shown in orange while its second-ring neighborhood is shown in green; each *particle-quad* (gray square) is formed by four particles, and each of which (except for the particles from the first and last layers) is shared by four *particle-quads*.

- *In-layer links*: connect to the particles of the same layer index, such as the link between particles $p_{0,7}$ and $p_{0,8}$ (curved arrow in green). Note that *in-layer links* can also be formed by particles of two grid-linked particles models but with the same layer index, see $\left(p^B_{1,9}, p^A_{1,4}\right)$ in Figure 4c.
- *Cross-layer links*: connect to the particles of different layers but with the same particle index j, such as the link between particles $p_{0,7}$ and $p_{1,7}$ (straight arrow in red).

To prevent the link between two particles from being repeatedly constructed, every link between two particles is associated with only one particle, specifically the one with a smaller i or j index. For example, the *in-layer* link between particle $p_{i,j}$ and $p_{i,j+1}$, denoted as $\left(p_{i,j}, p_{i,j+1}\right)$, belongs to $p_{i,j}$, while $p_{i,j+1}$ holds the link connecting to the next particle $p_{i,j+2}$. A similar strategy is applied to the *cross-layer* links. Therefore, the direction of each *in-layer* link is consistent with the particle distribution within each layer, that is, counterclockwise, which is denoted by *particle-layer* orientation.

3.1.3 | Particle neighborhood

As shown in Figure 2b, in the grid-linked particles model, for each particle, $p_{i,j}$, its stretching resistance term is modeled with the links (both *in-layer* and *cross-layer*) connecting to its *first-ring neighborhood*—neighboring particles that are directly adjacent to the current particle—denoted by $\mathcal{N}^{\,1}_{i,j} = \left\{p_{i,j-1}, p_{i,j+1}, p_{i-1,j}, p_{i+1,j}\right\}$ (particles in orange). The bending resistance term is modeled with the links connecting to its *outer-ring neighborhoods*—neighboring sets outside the *first-ring neighborhood*, $\left\{\mathcal{N}^{\,k}_{i,j} : 2 \le k \le K\right\}$ (particles in green), where K is the maximum number of outer-ring neighborhoods utilized to model the bending resistance. For example, when $K = 2$, only one *outer-ring neighborhood*, that is, the second-ring neighborhood, is employed to model bending resistance, $\left\{\mathcal{N}^{\,k=2}_{i,j}\right\} = \left\{\left\{p_{i,j-2}, p_{i,j+2}, p_{i-2,j}, p_{i+2,j}\right\}\right\}$.

3.1.4 | Particle-quads

As illustrated in Figure 2b, the grid-linked particles model is a grid-like structure made up of quads, namely *particle-quads* (see the quad in gray). For the *particle-quads* from the same model, each *particle-quad* is formed by four adjacent particles $Q_{i,j} = \left(p_{i,j}, p_{i,j+1}, p_{i+1,j+1}, p_{i+1,j}\right)$. Note that a *particle-quad* can also be formed by the particles from two grid-linked particles models (see $\left(p^A_{1,4}, p^B_{1,9}, p^B_{2,9}, p^A_{2,4}\right)$ in Figure 4c), which will be entailed in Section 4.1. The orientation of the four

particles is also in counterclockwise order, which forms a normal vector pointing to the outside of the surface based on the right-hand rule. In the grid-linked particles model, each particle (except the ones in the first and last layer) is shared by four particle-quads. *Particle-quads* provide surface information for the grid-linked particles model. Hence, maintaining good particle connectivity plays a vital role in surface mesh skinning. We will discuss this in detail in Sections 4 and 5, respectively.

3.2 | Grid-linked particles construction

As mentioned in the previous section, a grid-linked particles model can be considered a $L \times C$ grid wrapped around the intestine surface. Its initial shape is modeled by L number of circular sections (layers) evenly distributed along the medial axis of the model, with C number of particles uniformly sampled along each circle.

3.2.1 | Particle generation

Aiming to evenly place a set of circular sections along the model's medial axis and uniformly sample particles along the circle, we first align the model with the z-axis and then calculate the axis-aligned bounding box (AABB) of the model in the system's coordinate frame (see Figure 2a). The AABB of the intestine model is used to estimate the layer positions and the radius of the circular sections. There are L points, $\{o_i : 0 \leq i < L\}$, sampled along the model's medial axis. For each layer of index i, the point o_i presents the circle's origin in that layer, whose radius is $r_i = max\left(\frac{W}{2}, \frac{H}{2}\right)$, where W and H are the weight and height of the bounding box along x and y axes, respectively. Along each circle of layer i, we then sample C particles starting from particle $p_{i,0}$ at the bottom of the circle in counterclockwise order.

3.2.2 | Link construction

As mentioned in grid-particle representation, to prevent the link between two particles from being repeatedly constructed, every link between two particles is associated with only one particle with a smaller i or j index. Additionally, to ensure the grid-linked particles model has both stretching and bending resistance, for each particle, we choose both first-ring and second-ring neighborhoods to construct in-layer and cross-layer links. However, there can be more than one outer-ring of neighborhoods used when modeling the bending resistance of the grid-linked particles models, depending on the desired physical behavior of the soft body.

3.2.3 | *Particle-quads* recording

The surface of the grid-linked particles model is presented by *particle-quads*, each formed by four particles stored in counterclockwise order in our data structure. For every particle, we record all the particle-quads sharing that particle to keep track of particle connectivity, which will be updated and maintained during the *split and join* operations.

3.3 | Visual mesh skinning

Once the grid-linked particles model is constructed around the intestine's surface mesh, mesh skinning can be achieved in two steps: (1) mapping each mesh vertex to the closest *particle-quad*, and (2) updating the vertex position by interpolating the particle positions of the *particle-quad*.

3.3.1 | Vertices mapping

To map the visual mesh to the surface of the grid-linked particles model, each mesh vertex is mapped to a *particle-quad* satisfying two conditions: (a) containing the vertex's projection and (b) having the shortest distance to the vertex. For every

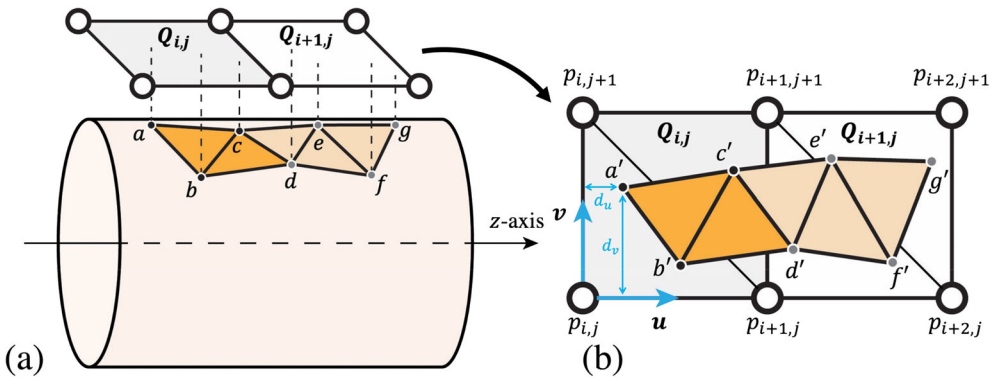


FIGURE 3 Visual mesh mapping and skinning. (a) Vertex mapping: each mesh vertex, for example, a, is mapped to a particle-quad (e.g., $Q_{i,j}$) that contains the vertex projection (e.g., a') and has the shortest distance to the particle. (b) Mesh skinning: after vertex mapping, each vertex is then represented by its u–v coordinates with respect to the particles of the particle-quad, and its position can be calculated by barycentric interpolation based on which triangle of the particle-quad the vertex is mapped to.

vertex, instead of mapping the vertex to every particle-quads to check proximity, the searching can be narrowed down to those particle-quads whose z-coordinates contain the vertex's z-coordinate. Among those candidate particle-quads, we then identify the particle-quad that has the shortest distance to the vertex. As shown in Figure 3, vertex a is projected to particle-quad $Q_{i,j}$ at a'.

For each particle-quad, for example, $Q_{i,j} = \left(p_{i,j}, p_{i+1,j}, p_{i+1,j+1}, p_{i,j+1}\right)$ in Figure 3b, we construct a local u–v coordinate system whose origin is at particle $p_{i,j}$, and the u-axis is along the layer direction while the v-axis is consistent with the direction of *particle-layer* orientation. Each vertex projected to a *particle-quad* can be represented by its u–v coordinates: $\left(u = \frac{d_u}{\|p_{i+1,j} - p_{i,j}\|}, v = \frac{d_v}{\|p_{i,j+1} - p_{i,j}\|}\right)$, where d_u and d_v are the distances from the vertex to the edges (links) of the *particle-quad*, and $u, v \in (0, 1)$ represent the local coordinates of the particles within the *particle-quad*.

3.3.2 | Mesh skinning

With each mesh vertex mapped to a particle-quad and represented by its local u–v coordinates, mesh skinning and deformation can be achieved by interpolating vertex positions based on the particles. Specifically, since each particle-quad $Q_{i,j}$ is made of two triangles, that is, $\Delta\left(p_{i,j}, p_{i+1,j}, p_{i,j+1}\right)$ and $\Delta\left(p_{i+1,j+1}, p_{i,j+1}, p_{i+1,j}\right)$, see Figure 3b, depending on which triangle a mesh vertex w is mapped to, vertex w's position can be calculated by performing barycentric interpolation of the triangle vertices based on the vertex's u–v coordinates based on Algorithm 1.

Algorithm 1. Mesh skinning

```
if w is inside Δ(p_{i,j}, p_{i+1,j}, p_{i,j+1}) then
    w = p_{i,j} + u(p_{i+1,j} − p_{i,j}) + v(p_{i,j+1} − p_{i,j})
else if w is inside Δ(p_{i+1,j+1}, p_{i,j+1}, p_{i+1,j}) then
    w = p_{i+1,j+1} + (1 − u)(p_{i,j+1} − p_{i+1,j+1}) + (1 − v)(p_{i+1,j} − p_{i+1,j+1})
```

4 | SPLIT AND JOIN OPERATIONS FOR INTESTINAL ANASTOMOSIS

In side-to-side stapled anastomosis, a linear stapler is used to split the sidewalls of the two intestine parts and then join them along their sidewall openings to create a continuous channel as shown in Figure 7. Based on this procedure, we derive our algorithm, *split* and *join*, performed on both the grid-linked particles models and the skinned surface meshes of the two intestine parts during the simulation. In the following, we introduce the *split and join* operations for both representations, respectively.

4.1 | Split and join grid-linked particles

4.1.1 | A Split

The split operation on the grid-linked particles involves two steps: (1) removing *in-layer links* of the particles affected by the cut and (2) splitting the *particle-quads* containing those *in-layer links* being cut. When performing anastomosis using a linear stapler, the two stapler parts must be inserted into the two intestine models with the same depth. As shown in Figure 4a, a cutting plane (plane in orange) formed by the two stapler parts cuts the same number of layers of the two grid-linked particle models. Moreover, the particle indices of the particle pair whose connection is split by the cutting plane should be the same for all the cutting layers of each model, denoted as $(j_c, j_c + 1)$.

A. Removing in-layer links being split

For every cutting layer, $i_c \in [0, L_c - 1]$, where L_c is the total number of particle layers being split, we remove the *in-layer link* between the particle pair $p_{i_c j_c}$ and $p_{i_c j_c+1}$. If there are more in-layer links associated to $p_{i_c j_c}$, then all of them will be removed. For simplicity, we consider the cutting plane intersects with each in-layer link at the middle point of the link for all the cutting layers of both models. As Figure 4b shows, the in-layer links of model A, that is, $\left(p^A_{1,3}, p^A_{1,4}\right)$ and $\left(p^A_{2,3}, p^A_{2,4}\right)$, and those of model B, that is, $\left(p^B_{1,9}, p^B_{1,10}\right)$ and $\left(p^B_{2,9}, p^B_{2,10}\right)$, are split by the cutting plane and thus removed from the data structure as shown in Figure 4c.

B. Handling *particle-quads* being split

In addition to splitting *in-layer links* for each cutting layer, the *particle-quad* containing those *in-layer links* should also be split. As shown in Figure 4b, there are two types of *particle-quads* depending on the number of edges being split: (1) *fully-split particle-quad* with two split edges (*in-layer links*), for example, $\left(p^A_{1,3}, p^A_{2,3}, p^A_{2,4}, p^A_{1,4}\right)$; and (2) *partially-split particle-quad* with only one split edge (*in-layer link*), for example, $\left(p^A_{2,3}, p^A_{3,3}, p^A_{3,4}, p^A_{2,4}\right)$. *Fully-split particle-quads* are directly removed from the grid-linked particles model (see Figure 4c). For the *particle-quads* that are partially split, we still keep them because the inside mesh is not fully split. The connectivity of the remaining mesh that is not affected by the split should be preserved. Special handling for *partially-split particle-quads* will be discussed in detail in the join operation.

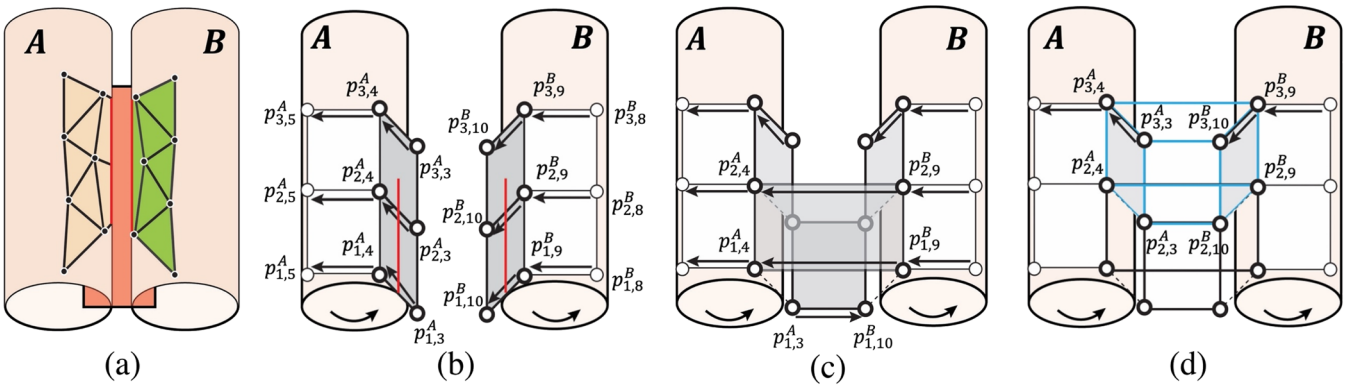


FIGURE 4 Split and join grid-linked particles. (a) The cutting plane (plane in orange) formed by the two linear stapler parts inserted into the two intestine models A and B cuts both models with the same depth (number of layers). (b) The split operation removes the *in-layer links* (e.g., $\left(p^A_{1,3}, p^A_{1,4}\right)$) and fully splits the particle-quads (e.g., $\left(p^A_{1,3}, p^A_{2,3}, p^A_{2,4}, p^A_{1,4}\right)$) that intersect with the cutting plane. The directions of *in-layer links* that are consistent with the *particle-layer* orientation are shown with curved black arrows. (c) The join operation creates new *in-layer links* (e.g., $\left(p^B_{1,9}, p^A_{1,4}\right)$) and *particle-quads* (e.g., $\left(p^A_{1,4}, p^B_{1,9}, p^B_{2,9}, p^A_{2,4}\right)$) connecting particles across the two models to form a single loop with the same *particle-layer orientation* along each layer being cut. (d) A *particle-cube* (in blue) is constructed to join the meshes being split at the split end within the *partially-split particle-quads* (i.e., $\left(p^A_{2,4}, p^A_{2,3}, p^A_{3,4}, p^A_{2,4}\right)$ and $\left(p^B_{3,9}, p^B_{3,10}, p^B_{2,10}, p^B_{2,9}\right)$) of the two models.

4.1.2 | A Join

The join operation is performed on the two grid-linked particles models in order to join them together along the cut openings created by the split operation. The join operation is simultaneously conducted on two grid-linked particles models together, consisting of two steps: (1) building new *in-layer links* across the two models layer by layer, and (2) creating new *particle-quads* across the two models on the surface for every two adjacent layers.

A. Building new in-layer links

To join the cutting layers of the two models into one layer, we construct new *in-layer links* to connect the particles of one model to those of the other model. The particles that are used to form those new *in-layer links* are the ones whose original *in-layer links* are split and removed by the split operation. Moreover, the direction of each new *in-layer link* should be consistent with the *particle-layer orientation*, that is, counterclockwise. For example in Figure 4b,c the split operation removes both in-layer link $\left(p^A_{1,3}, p^A_{1,4}\right)$ of model A and the in-layer link $\left(p^B_{1,9}, p^B_{1,10}\right)$ of model B. To join the two models, we create new *in-layer links* $(p^A_{1,3}, p^B_{1,10})$ and $\left(p^B_{1,9}, p^A_{1,4}\right)$, connecting all the particles of layer 1 in a single loop.

B. Creating new particle-quads across two models

For every two adjacent layers being cut, we construct two new *particle-quads* to connect the two grid-linked particles models on the surface. For instance, since the first wo layers are split in Figure 4b, the two *particle-quads*, $\left(p^A_{1,4}, p^B_{1,9}, p^B_{2,9}, p^A_{2,4}\right)$ and $\left(p^A_{1,3}, p^B_{1,10}, p^B_{1,9}, p^A_{2,3}\right)$, are built to join the two models on top and bottom. The newly created *particle-quads* replace the *particle-quads* that have been fully split and removed by the split operation. The mesh vertices corresponding to the particle-quads that are being removed will be mapped to the new particle-quads in order to join the two intestine meshes together. This will be discussed in detail in the next section.

Special handling is needed when joining the two models in the last cutting layer, $(L_c - 1)$. As explained in Section 4.1(A), each model's *partially-split particle-quad* containing the last cutting layer is still kept after the split operation in order to preserve the surface connectivity of the uncut mesh. Moreover, since the cutting within a *partially-split particle-quad* ends at a point, the mesh vertices must present this information within that *particle-quad*. Our goal is to join the cut surface meshes of both models at split endpoint while still retaining each model's surface connectivity of the unsplit parts. As demonstrated in Figure 4d instead of creating two new *particle-quads* from top and bottom, we create one *particle-cube* (cube in blue) formed by the eight particles of the two *partially-split particle-quads* of the two models, that is, $\left(p^A_{2,4}, p^A_{2,3}, p^A_{3,4}, p^A_{2,4}\right)$ and $\left(p^B_{3,9}, p^B_{3,10}, p^B_{2,10}, p^B_{2,9}\right)$. The *particle-cube* can be used to join the portions of the surface meshes of both models being cut to a point by considering all eight particle positions. The *partially-split particle-quads* on the two sides of the particle-cube are still kept for the mesh vertices that are not affected by the split for both models. We will elaborate the handling of partial-split case on the skinning meshes in the next section.

After performing the join operation on the grid-linked particles models, each particle (except the first and last layers) is still shared by four particle-quads. Maintaining valid particle connectivity in the grid-linked particles models is essential to joining the surface meshes and making sure the cut meshes of both models are seamlessly stitched together and form a continuous channel.

4.2 | Split and join skinned surface meshes

As mentioned in Section 3.3, the surface mesh is bound to the grid-linked particles of an intestinal model by mapping each mesh vertex to exactly one *particle-quad* closest to the vertex's original position. Once mapped, the displacement of the skinned mesh's vertices is entirely determined by its u–v coordinates with respected to the *particle-quad*. Such one-to-one correspondence also simplifies the split and join operations performed on the meshes. Specifically, the meshes can be updated by mapping vertices to different portions of the existing *particle-quads* (split) or the newly created *particle-quads* or a *particle-cube* when connecting the meshes of the two intestine parts (join).

This section introduces the split and join operations for the skinned surface mesh. We also discuss the handling of the mesh vertices mapped to *partially-split particle-quads* to join the meshes at split end, with the help of a *particle-cube*.

4.2.1 | A Split

Figure 5a–d give an example of splitting the skinned mesh of model A in Figure 4a. For each *particle-quad* being fully split, a splitting line can be generated by connecting the middle points of the two splitting *in-layer links* contained by the *particle-quad*. Within each *particle-quad* being fully cut, we first identify a sequence of vertices, called *splitting vertices*, based on three conditions: (1) closest to the splitting line segment, (2) connected to the previous splitting vertex by an edge, and (3) not forming a triangle with the previous two *splitting vertices*. We then identify all the triangles sharing those *splitting vertices*. Each *splitting vertex* is then projected to the splitting line and split into two identical vertices (Figure 5c,d). To split the triangles along the splitting line, we first need to label all the triangle vertices (including splitting vertices) to be *above* or *below* the splitting line by comparing the v-coordinates of those vertices with the splitting line's v-coordinate (i.e., 0.5). The triangle vertices with larger v-coordinates are tagged *above*; otherwise, they are labeled *below*. Since each cut triangle should contain no more than two splitting vertices, we label the triangle based on its non-splitting vertex(s)'s *above*/*below* label. For example, the triangle with vertices a_2, a_3, a_4 in Figure 5b is labeled *below* based on the only non-splitting vertex a_3. At last, for the triangle labeled *above*, all of its vertices remain the same and for those labeled *below*, all of its splitting vertices are replaced with their duplicates (see triangle with vertices a_2', a_3, a_4' in Figure 5d). Therefore, the mesh mapped to the *fully-split particle-quads* can be separated along the splitting line completely.

The split operation performed on the mesh mapped to a *partially-split particle-quad* is implemented differently from the fully-split case. As shown in Figure 6a, in a *partially-split particle-quad* the splitting line ends at a point at which the previous two mesh parts separated by the splitting line join again, whereas the mesh part beyond that point remains intact. To model this connectivity, we first need to identify which vertices of the *partially-split particle-quad* should be considered *splitting vertices* to model the split endpoint. To identify such vertices, we first calculate the projection of each

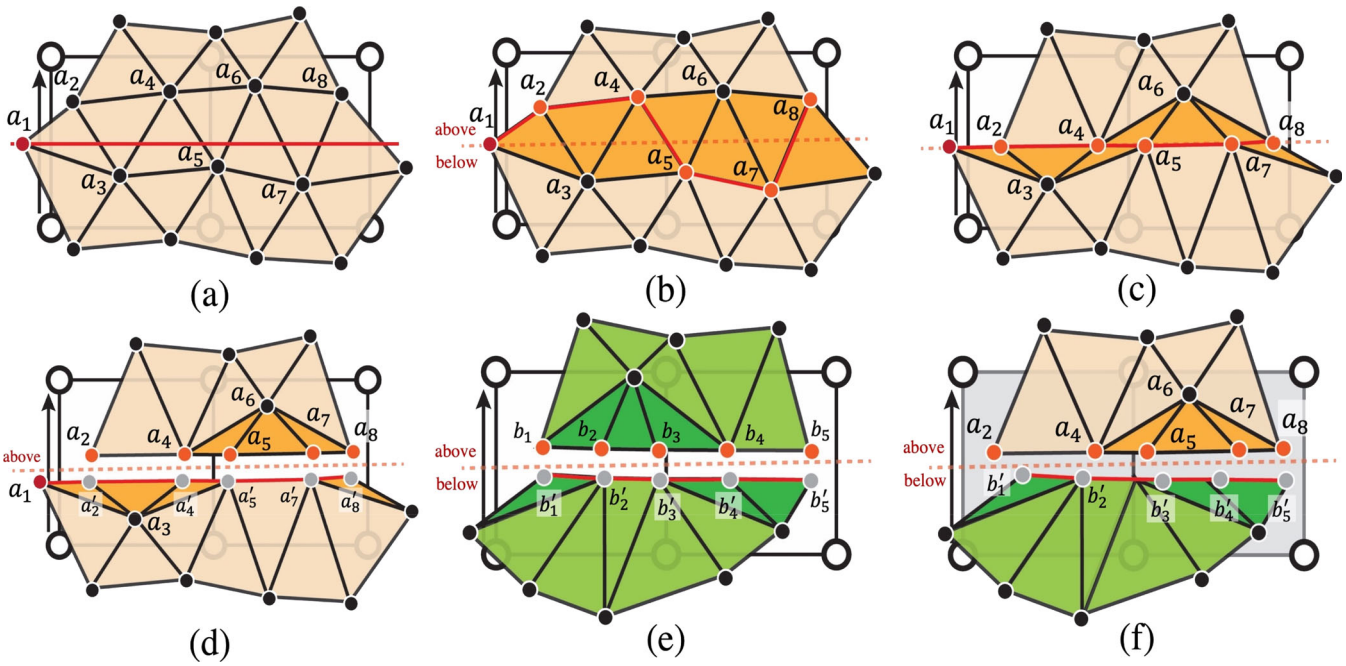


FIGURE 5 Split and join skinned meshes of intestine models A and B in Figure 4a. (a) Splitting model A's mesh with the splitting line (horizontal line in red) passing through the intersections between the cutting plane in Figure 4a and the particle-quads' vertical edges (in-layer links) at v-coordinate = 0.5. The direction of the in-layer links being split indicates (vertical arrow in black) "*above*" and "*below*" of a vertex with respect to the splitting line. (b) Identifying a sequence of connected *splitting vertices* (points in orange) that will be used to split the mesh against to the splitting line. (c) Projecting all the splitting vertices onto the splitting line. (d) Duplicating splitting vertices and assigning them to the triangles (triangles highlighted) with labels "*above*" or "*below*" sharing those splitting vertices, so that the mesh can be completely separated against the splitting line. (e) Mesh splitting result of model B by following the same procedure of (a–d). (f) Mapping the "above" triangles of model A and "below" triangles of model B to the new particle-quad on top (see Figure 4c); since the splitting vertices of A (point in orange) and the duplicates of splitting vertices of B (points in gray) are all at v-coordinate = 0.5, the meshes from A and B can be joined seamlessly when interpolating based on the same four particles of each new particle-quad (in gray).

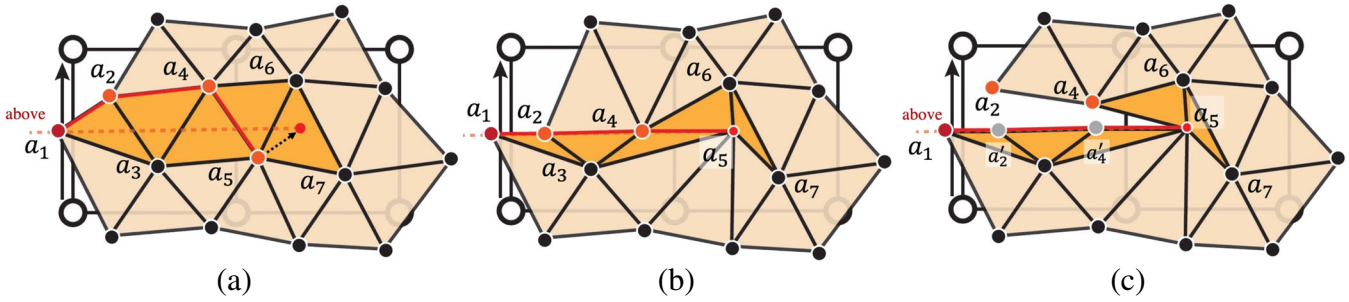


FIGURE 6 Splitting skinned mesh within a *partially-split particle-quad* (the particle-quad on the right). (a) The splitting vertices within a *partially-split particle-quad* can be identified in a similar fashion as the ones within a *fully-split particle-quad*. The vertices (e.g., a_7) whose projections are beyond the splitting line will not be considered as *splitting vertices* and will be left intact. In this example, the *splitting vertices* within the *partially-split particle quad* is a_5, and it is connected with the splitting vertices from the previous particle-quad (i.e., a_4). (b) For a *fully-split particle quad* (left) the splitting vertices are projected onto the splitting line. However, for a *partially-split particle quad*, the splitting vertices are moved to the endpoint of the splitting line to represent the split end (red dot in [a]). (c) The two mesh parts that are separated by the split operation are joined at the endpoint of the splitting line (i.e., a_5).

vertex mapped to the splitting line within the *partially-split particle-quad*. For those vertices whose projections are inside the splitting line segment, we follow the same three conditions mentioned in the full-split case in Section 4.2(A) to further determine *splitting vertices*. Once all the *splitting vertices* are identified, they are moved to the endpoint of the splitting line to present the cut-end (see a_5 in Figure 6b). At last, the two mesh parts that are separated by the split operation within *fully-split particle-quads* are joined at the endpoint of the splitting line (i.e., a_5) as illustrated in Figure 6c. The rest of the mesh within the *partially-split particle-quad* beyond the splitting line remain the same.

4.2.2 | A Join

For every *particle-quad* being fully split, the triangles associated with those particle-quads are labeled *above* or *below* according to the splitting line. As the model *A* in Figure 5f shows, all the vertices of the triangles labeled *above* are mapped to the new *particle-quad* on top as shown in Figure 4c; whereas, all the vertices of the triangles labeled *below* are mapped to the new particle-quad at the bottom. The same mapping procedure is applied to model *B* in Figure 5e but with the opposite manner, that is, the vertices of the triangles labeled *above* are mapped to the new particle-quad at the bottom, while those of the triangles labeled *below* are mapped to the new particle-quad on top. Within each new particle-quad, we interpolate the positions of all the splitting vertices and their duplicates from both models based on the same particles according to their *u*–*v* coordinates. Since the *v*-coordinates of all the *splitting vertices* and their duplicates are the same for both models, the two mesh of the two models can be joined seamlessly.

As mentioned in Section 4.1(A), when splitting the mesh within a *partially-split particle-quad* of the intestine model, the two mesh parts that are separated along the splitting line are joined at the same endpoint of the splitting line within a *partially-split particle-quad*. In order to join the meshes of the two models at the same point between the two models, we construct a *particle-cube* using all the particles from the two *partially-split particle-quads* of the two models, see the blue cube in Figure 4d. The position of the endpoint within the particle-cube can be calculated by averaging the two splitting endpoints within the *partially-split particle-quads* of the two models. We then map all the *splitting vertices* within the *partially-split particle-quads* identified in the split operation to the endpoint of the particle-cube aiming to join the meshes in the end. Therefore, two meshes are joined together along their cut openings and eventually merged to a single point where the splitting ends; beyond that point, the two meshes separate from each other while remaining the same.

5 | EXPERIMENTAL RESULTS

The proposed split and join algorithm for the side-to-side stapled anastomosis surgical procedure has been implemented in Unity[20] on a standard desktop computer with an Intel Core i7-4930K, 3.4 GHz CPU and 16.0 RAM. Our implementation

also allows the user to interact with the intestine models through the virtual surgical tools (linear stapler and forceps) controlled by Geomagic Touch haptic devices.[21]

In this section, we demonstrate experimental results of the proposed method from three aspects: (1) intestine simulation, (2) anastomosis simulation based on the split and join operations, and (3) tool-model interactions. Specifically, three quantitative experiments were designed to measure the first two aspects.

5.1 | Intestine simulation

The intestine model is represented by a hybrid structure – a surface mesh for visualization and a grid-linked particles model for physics. Figure 7a–c demonstrates the experimental results of intestine simulation by using the proposed grid-linked particles models. For each input intestine mesh with approximately 6K triangles, we construct a grid-linked particle model based on a 20×20 particle grid—20 layers of particles with 20 particles evenly distributed along each layer (circle). Each particle is connected to its neighboring particles through two *in-layer links* and two *cross-layer links*, respectively. In this work, each grid-linked particle model was implemented as a mass-spring system (MSS) based on Unity. Each particle is implemented as a rigid sphere game object. Each of the *in-layer* and *cross-layer link* is modeled with a configurable joint in Unity, which can be considered as a damped spring with six-degree of freedom. During the simulation, once the input mesh is bound to the grid-linked particle model, the skinned mesh starts deforming as the associated particles start moving under gravity. Our experiment's average rendering frame rate is around 58 frames per second (FPS), demonstrating its ability to render in real time (i.e., >30 FPS).

5.1.1 | Experiment 1: Intestine simulation with increasing particle resolution

To further evaluate the scaling of the rendering frame rates with respect to the total number of particles being used for simulating the intestine models, we gradually increase the number of layers and the number of particles per layer for each

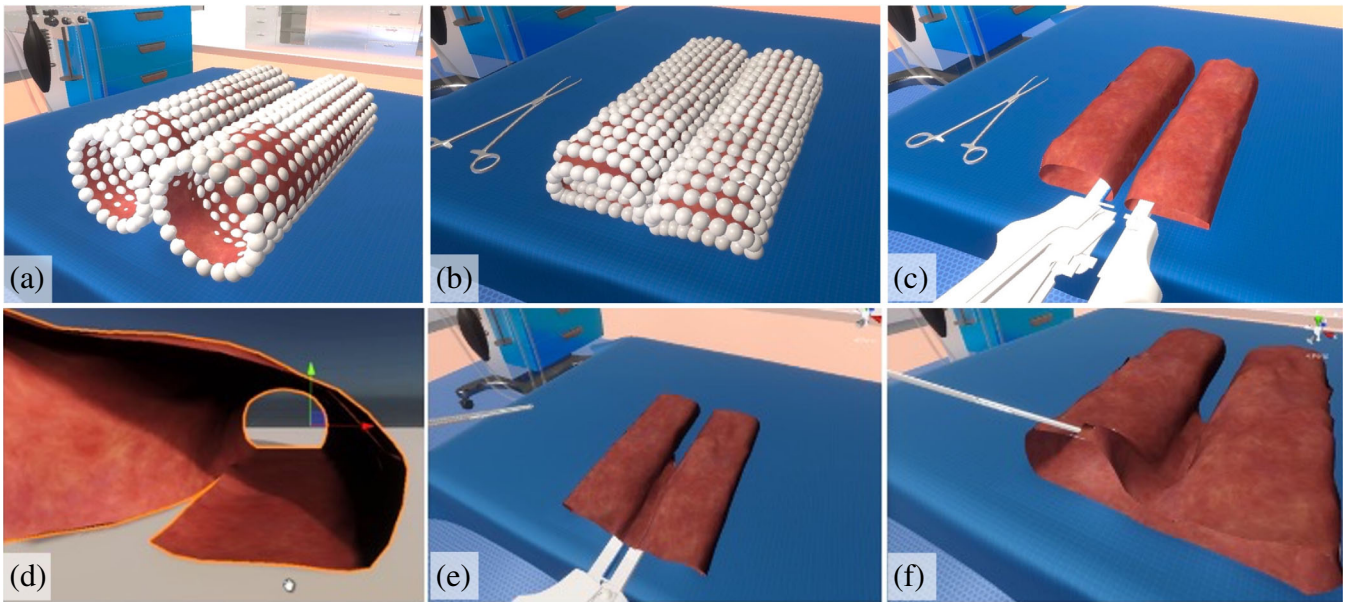


FIGURE 7 Experiment results of the *split and join* algorithm for the side-to-side stapled anastomosis surgical procedure, implemented in Unity running on a desktop computer. (a) The input intestine models whose surface meshes are encapsulated in grid-linked particles models for physics simulation. (b) The intestine meshes are deformed according to their grid-linked particles models during the simulation. (c) The two parts of a linear stapler are inserted inside the intestine models placed side-by-side before performing the split operation. (d) The result of splitting one of the intestine models on the side. (e) The result of joining two intestine models together based on the actual linear stapler operational length. (f) Grasping the intestine models where they are joined using a surgical tool controlled by a haptic device held by the user.

TABLE 1 Average rendering frame rate (FPS) with increasing particles being used to simulate the intestine models.

#Layers	#Particles/layer	#Particles (2 models)	Average FPS
11	10	220	71
20	10	400	68
20	15	600	62
20	**20**	**800**	**58**
25	20	1000	56
25	24	1200	52
26	27	1404	49
28	29	1624	46
30	30	1800	43
32	32	2048	32
35	32	2240	29
35	35	2450	25

Note: The number of layers and the number of particles per layer are the same for both models. The values in bold (i.e., 20 layers and 20 particles per layer for each model) are the configuration used in our simulation.

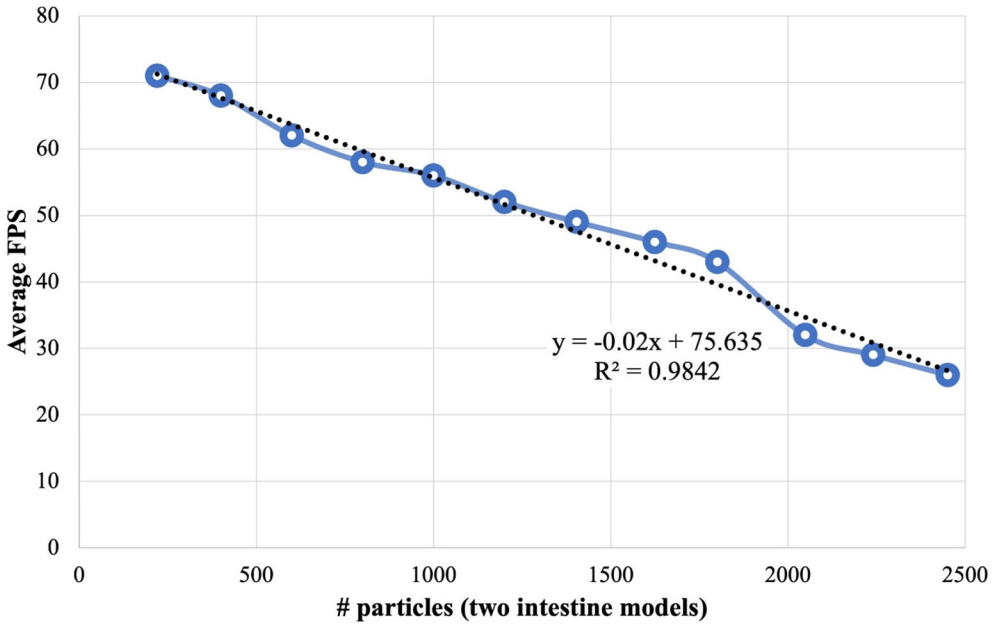


FIGURE 8 Average rendering frame rate with increasing number of particles being used to simulate the intestine models. Specific values can be found in Table 1.

model, as illustrated in Table 1. As shown in Figure 8, the average rendering speed shows an overall linear scaling with respect to the number of particles being used.

5.2 | Side-to-side anastomosis simulation using split and join operations

In the side-to-side stapled anastomosis, the linear stapler is inserted into two intestinal parts and aligned side-by-side, see Figure 7c. When closing and firing the linear stapler, the sidewalls of both intestine models are simultaneously split and

TABLE 2 Computation time for splitting and joining intestine models with increasing particle resolution and a fixed number of splitting layers (i.e., 10); time is measured in milliseconds (ms).

				Comp. time (ms)		
#Layers	#Particles/layer	#Particles (2 models)	#Split layers	Split	Join	Total
20	20	800	10	73	5	78
25	20	1000	10	75	5	80
25	24	1200	10	78	5	83
26	27	1404	10	80	5	85

Note: That the computation time here comprises the ones spent on both grid-linked particles and surface mesh models of both intestines.

joined together to form a continuous channel. For the first time, this procedure has been effectively simulated by the proposed split and join operations that are performed on both grid-linked particles models and skinned surface meshes of the two intestine models during the simulation. Figure 7d shows the simulation result of the split operation for one of the models, where the skinned mesh is split into two pieces along the region intersected by the linear stapler. Although, the grid-linked particles models have been split as well, they are not visible to the user during the simulation. As illustrated in Figure 7e, the two intestine meshes can be joined along the cut openings to form a continuous tunnel along the anastomosis region. Our result also demonstrates that the split and join algorithm allows the anastomosis to be performed based on the actual insertion depth of the linear stapler, see Figure 7e. Although, we introduce split and join as two operations and demonstrate their results separately, in the side-to-side stapled anastomosis simulation, these two operations are performed one after the other automatically. There is no intermediate result shown to the user.

To analyze the performance of the split and join operations, we further present two experiments: (a) split and join intestine models with increasing particle resolution and (b) split and join intestine models with an increasing length of the splitting line.

5.2.1 | Experiment 2: Split and join intestine models with increasing particle resolution

In this experiment, we split both intestine models with the same number of particle layers while gradually increasing the particle resolution of both models; the underlying surface meshes remain the same. Our goal was to evaluate the change in the performance of split and join operations for the same split configuration but different particle resolutions. As shown in Table 2, it can be observed that, compared to the increment in the particle resolution, the split time only increases slightly while the join time remains the same. Our interpretation is that the computation time of our split and join algorithm mainly depends on the number of particle layers being split rather than the number of particles within each layer.

5.2.2 | Experiment 3: Split and join intestine models with an increasing length of the splitting line

To further evaluate the performance of the split and join algorithm, we gradually increase the number of layers being split each time, as shown in Figure 9, while fixing the particle resolution for both models to be 400, that is, 20 layers of particles with each layer of 20 particles. It is clearly shown in Table 3 that the computation was mainly spent on the split operation, while the increase in the join time is relatively small. Our interpretation is that the split operation involves splitting both the grid-linked particles and surface meshes for both models. As the splitting line increases, more particle links and mesh elements are involved in the procedure. The split time increases almost linearly with respect to the number of particle layers being split.

5.3 | Collision detection and tool interactions

In this work, the collision detection of the intestine models is handled by the particles – rigid sphere game objects and directly supported by Unity's collision module. Our implementation allows the user to interact with the intestine models

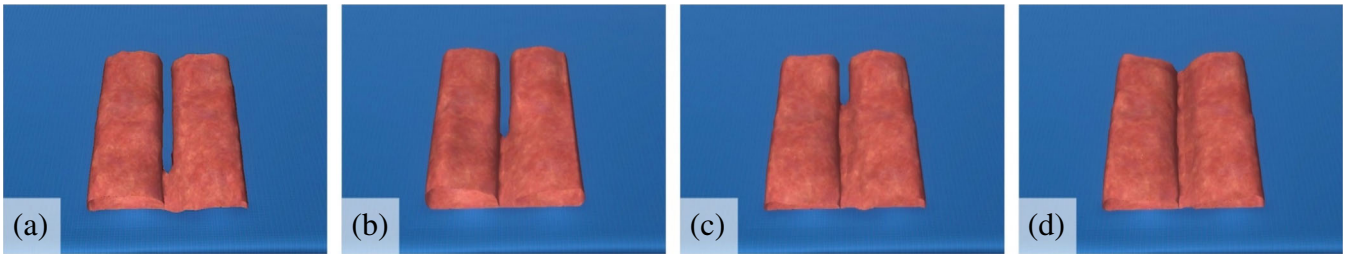


FIGURE 9 Split and join intestine models (20 layers with 20 particles each layer) with increasing the number of splitting layers, that is, 5 (a), 10 (b), 15 (c), and 20 (d) layers.

TABLE 3 Computation time for splitting and joining intestine models of a fixed particle resolution (i.e., 400 particles—20 layers, 20 particles per layer) with increasing length of splitting line (i.e., #split layers), see Figure 9; time is measured in milliseconds (ms).

	Comp. time (ms)		
#Split layers	Split	Join	Total
5	58	4	62
10	73	5	78
15	98	7	105
20	121	9	130

such as using forceps to grasp the surface of the intestine model as well as inserting linear stapler into the models (see Figure 7c,f). For example, once the collision between the forceps (with capsule-shaped colliders attached to its jaws) and some particles is detected with the forceps pushing or pulling the model, the portion of the surface mesh controlled by those particles will be pushed down or lifted up accordingly in response to the force applied to the model.

6 | CONCLUSION AND FUTURE WORK

Stapled anastomosis has become integral to routine surgical intervention for colorectal cancer. Developing an effective surgical training mechanism for such a high-skilled procedure is critical for improving patient safety. This work is the first attempt to simulate a side-to-side anastomosis procedure in virtual reality, which involves sophisticated soft tissue manipulations such as cutting and stitching. In this paper, we propose a novel *split and join* approach to simulate side-to-side stapled anastomosis surgical procedure for the first time. We simulate the intestine model using a new hybrid representation—a new particle-based representation, which uses grid-linked particles to model the physics and a surface mesh for rendering. The proposed split and join operations can efficiently handle the modifications and updates of both the grid-linked particles model and the surface mesh. Our experimental results demonstrate the feasibility of the proposed approach in simulating intestine models and the side-to-side anastomosis operation.

We plan to further improve and extend our proposed method along three technical directions.

6.1 | Surface collision detection

Our current implementation of grid-linked particles does not support surface collision detection. Although, collision detection and handling are not the focus of this work, effective surface collision is crucial for side-to-side stapled anastomosis surgical simulation. Incorporating colliders on the surface of the intestine models will allow more robust self-collision handling and better tool interaction support. In the future work, we plan to implement colliders based on the *particle-quads* on the surface of the grid-linked particles to detect and respond to the collisions coming from other particles (from either the same or different intestine models) and the tools more robustly.

6.2 | Tool-model interactions

This work allows the user to perform some simple interactions with the intestine models such as using forceps to grasp the intestine surface and inserting the linear stapler inside the intestine models. Although, particles could handle collisions coming from large-scale objects, slender objects like a linear stapler may still penetrate the intestine surface when the gaps between the particles become too large when the particle model is lifted up at the same time. We consider this issue can be effectively addressed by incorporating surface collision as mentioned above. After implementing the surface colliders based on particle-quads, our next step is to improve tool interactions with the grid-linked particles model, in order to support linear stapler insertion more robustly.

6.3 | Split and join based on particle-based dynamics

Realizing stapled anastomosis simulation in VR is the main focus of this paper. For proof of concept and simplicity, we used a mass-spring system available in Unity to simulate intestines and implemented the split and join algorithm on MSS. On the other hand, particle-based dynamics (PBD) may produce more realistic results than MSS due to the representation of small volumes. It can be parallelized to improve the simulation performance further. We plan to implement the PBD algorithm for simulating intestines and adapt the proposed split and join algorithm to handle different types of constraints (distance, volume, etc.) in PBD.

Upon the completion of surface collision detection and tool interaction improvements, we plan to further evaluate the validity of the VR simulation system as a surgical training tool with medical professionals. We intend to integrate actual surgical instruments, such as a linear stapler, into the user interface of the simulation system to allow the user to manipulate virtual intestines and perform stapled anastomosis directly. The next step is to conduct face and construct validation studies by recruiting experienced surgeons and medical students to evaluate the usability of the simulation system for surgical training.

ACKNOWLEDGMENTS

The authors would like to thank Dr. Rahul—Senior Research Scientist at Rensselaer Polytechnic Institute, for providing insightful comments on the manuscript. Research reported in this article was supported by the National Institute of Biomedical Imaging and Bioengineering (NIBIB) of the National Institutes of Health under Award Number R01EB005807. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health.

ORCID

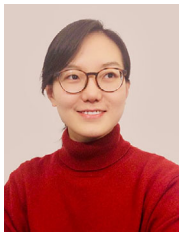
Di Qi https://orcid.org/0000-0002-0467-6628

REFERENCES

1. U.S. Department of Health and Human Services, Center for Disease Control and Prevention and National Cancer Institute. U.S. Cancer Statistics [Online]; 2022. Available from: www.cdc.gov/cancer/dataviz
2. Goulder F. Bowel anastomoses: the theory, the practice and the evidence base. World J Gastrointest Surg. 2012;4(9):208-13.
3. Feng J-S, Li J-Y, Yang Z, Chen X-Y, Mo J-J, Li S-H. Stapled side-to-side anastomosis might be benefit in intestinal resection for Crohn's disease: a systematic review and network meta-analysis. Medicine. 2018;97(15):e0315.
4. LapSim. Surgical science [Online]; 2022. Available from: http://www.surgical-science.com/portfolio/lapsim-basic-skills/
5. Lap Mentor. Simbionix [Online]; 2022. Available from: http://simbionix.com/simulators/lap-mentor/
6. Jayasudha K, Kabadi MG. Soft tissues deformation and removal simulation modelling for virtual surgery. Int J Intell Sustain Comput. 2020;1:83-100.
7. Bro-Nielsen M. Finite element modeling in surgery simulation. Proc IEEE. 1998;86(3):490-503.
8. Koschier D, Bender J, Thuerey N. Robust eXtended finite elements for complex cutting of deformables. ACM Trans Graph. 2017;36(4):1–13.
9. Cotin S, Delingette H, Ayache N. Real-time elastic deformations of soft tissues for surgery simulation. IEEE Trans Vis Comput Graph. 1999;5:62–73.
10. Xu L, Lu Y, Liu Q. Integrating viscoelastic mass spring dampers into position-based dynamics to simulate soft tissue deformation in real time. R Soc Open Sci. 2021;5(2):171587.
11. Shewchuk JR. What is a good linear finite element? - interpolation, conditioning, anisotropy, and quality measures; 2022.
12. Muller M, Heidelberger B, Hennix M, Ratcliff J. Position based dynamics. J Vis Commun Image Represent. 2007;18(2):109–18.

13. Camara M, Mayer E, Darzi A, Pratt P. Soft tissue deformation for surgical simulation: a position-based dynamics approach. Int J Comput Assist Radiol Surg. 2016;11(6):919–28.
14. Chang J, Yang X, Pan JJ, Li W, Zhang JJ. A fast hybrid computation model for rectum deformation. Vis Comput. 2011;27:97–107.
15. Chang J, Shepherd DX, Zhang JJ. Cosserat-beam-based dynamic response modelling. Comp Anim Virtual Worlds. 2007;18:429–436.
16. France L, Lenoir J, Angelidis A, Meseure P, Cani MP, Faure F, et al. A layered model of a virtual human intestine for surgery simulation. Med Image Anal. 2005;9:123–32.
17. Lorensen WE, Cline HE. Marching cubes: a high resolution 3D surface construction algorithm. In: Proceedings of the 14th annual conference on computer graphics and interactive techniques – SIGGRAPH '87; 1987. p. 163–9.
18. Pan JJ, Chang J, Yang X, Liang H, Zhang JJ, Qureshi T, et al. Virtual reality training and assessment in laparoscopic rectum surgery. Int J Med Robot Comput Assist Surg. 2015;11(2):194–209.
19. Qi D, Panneerselvam K, Ahn W, Arikatla V, Enquobahrie A, De S. Virtual interactive suturing for the fundamentals of laparoscopic surgery (FLS). J Biomed Inform. 2017;75:48–62.
20. Unity. [Online]. 2022. Available from: unity.com
21. I. 3D Systems. Geomagic touch haptic device. *3D Systems* [Online]; 2014. Available from: https://s3.amazonaws.com/dl.3dsystems.com/binaries/Sensable/UserGuide/Geomagic+Touch_User_Guide_Ethernet.pdf

AUTHOR BIOGRAPHIES



Di (Trudi) Qi is an Assistant Professor of Electrical Engineering and Computer Science at the Fowler School of Engineering at Chapman University. Her research is at the intersection of computer graphics, virtual reality (VR), and artificial intelligence (AI). She is particularly interested in integrating AI with visual computing technologies and connecting her research to human-centered endeavors such as education, training, and healthcare. Before her faculty appointment in 2021, she was a Postdoctoral researcher at Rensselaer Polytechnical Institute, where she laid the groundwork for developing real-time physics-based simulation techniques for virtual surgery and being part of multiple NIH-funded projects of VR medical and surgical training systems.

Suvranu De is the Dean of the joint College of Engineering of Florida A&M University and Florida State University. His research interests include the development of novel, robust, and reliable computational technology to solve challenging and high-impact problems in engineering, medicine, and biology. He is the recipient of the ONR Young Investigator Award (2005), the Rensselaer School of Engineering Research Excellence Award (2008), the James M. Tien '66 Early Career Award for Faculty (2009), the Rensselaer School of Engineering Outstanding Research Team Award (2012), the J. Tinsley Oden Medal of the U.S. Association for Computational Mechanics (2019) and the Edwin F. Church Medal from the American Society of Mechanical Engineers (2022). He serves on the editorial boards of multiple journals as well as scientific committees of numerous national and international conferences. He is a Senior Member of IEEE and serves as Vice-Chair (Awards) of the IEEE Technical Committee on Haptics and leads/co-leads several committees of the Society of American Gastrointestinal and Endoscopic Surgeons (SAGES). He is an elected Fellow of four professional societies: the American Society of Mechanical Engineers (ASME), the American Institute for Medical and Biological Engineering (AIMBE), the International Association for Computational Mechanics (IACM), and the United States Association for Computational Mechanics (USACM).

How to cite this article: Qi D, De S. Split and join: An efficient approach for simulating stapled intestinal anastomosis in virtual reality. Comput Anim Virtual Worlds. 2023;e2151. https://doi.org/10.1002/cav.2151