

Pilot Study on Runtime Mesh Manipulation

Anurag Basavaraj Bhusare¹, Keshava S H², Anjan R Prasad³, Leharaditya Kumar⁴, Adithya Balasubramanyam⁵
and Prasad Honnavalli⁶

¹⁻⁶PES University/Department of Computer Science, Bengaluru, India

Email: anuragbhusare44@gmail.com, {keshavash2003, anjanprasad.21, arry8668}@gmail.com

Abstract— Runtime mesh manipulation is a critical challenge in domains like computer graphics, medical simulation, and advanced manufacturing. This work introduces two interrelated pipelines—Mesh Cutting Pipeline and Mesh Stitching Pipeline—that enable dynamic and real-time mesh manipulation while preserving structural and visual coherence. The Mesh Cutting Pipeline includes leading innovations in terms of precision contact detection, entry/exit point computation, dynamic vertex generation, localized re-triangulation, and progressive tracking of the cut path toward ensuring geometric continuity and reduced distortion. The Mesh Stitching Pipeline reconstructs cut edges by using path data so as to achieve seamless connectivity while avoiding maximum artifacts and to achieve topological consistency. This framework offers significant applications: in surgical simulation, it models precise virtual incisions for medical training and planning; in interactive 3D modeling, it supports intuitive real-time editing; and in computer-aided manufacturing, it facilitates tool path visualization and material removal simulation. Performance evaluations demonstrate real-time processing efficiency and high visual fidelity. Our approach provides a robust, noise-free solution for interactive mesh modification, balancing computational efficiency and visual quality.

Index Terms—Virtual reality, Mesh Manipulation, 3D Modelling, Real-time Processing.

I. INTRODUCTION

Virtual reality (VR) has gone from being a niche technology to a transformative platform that changes the way industry's function, be it gaming, education, or medical training. Immersive experiences are made possible through the simulation of realistic environments using 3D meshes composed of interconnected triangles and vertices. The meshes used determine how virtual objects will behave and appear, and thus are integral to the realism and interactivity. However, increasing the detail of the mesh for higher realism often demands significant processing power, which can hinder real-time performance, particularly in scenarios that require user interaction. The major challenge in VR systems is the real-time manipulation of meshes through cutting and stitching. Existing approaches are mostly based on pre-programmed animations or fixed operations. These solutions make it visually appealing but not flexible enough for true real-time interactivity. To solve this, we propose a new system that allows dynamic mesh cutting and stitching at runtime without disturbing the structure or appearance of the mesh. The above three capabilities, using advanced collision detection, real-time triangulation, and efficient stitching algorithms, make our system more interactive than ever before. Such interactivity is crucial in applications such as surgical simulation, where meshes need to be cut and stitched with precision, mimicking the reality of incisions and repairs (see Fig. 1).

Similarly, designers use virtual reality for intuitive manipulation of virtual prototypes in 3D modeling, while gamers are immersed in dynamic environments, such as destructible terrains.

Overview Diagram

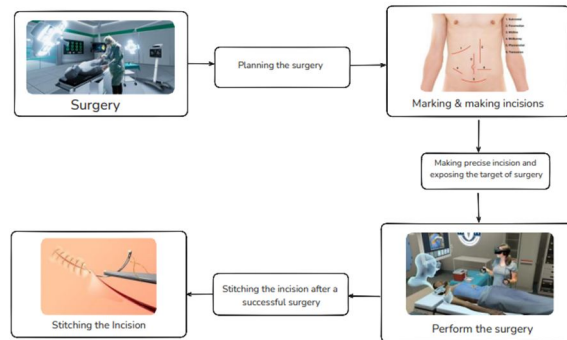


Fig 1: Overview of Surgical Simulation Process

A. Applications of VR

VR has many applications beyond video games. In medical training, precise 3D meshes of human organs and systems facilitate the risk-free practice of complex surgery among surgeons [1][2]. The same applies to the domain of education, where VR can even rebuild historical settings [3][4] so that learners interact with reconstructed cities and artifacts for their proper learning. In architecture and construction, it enables virtual tours of the building designs and allows the modification of designs before actual construction, hence saving resources and time. Its largest application is still in games [5][6], where dynamic meshes are the key to offering realistic and interactive experiences.

Runtime Mesh Manipulation

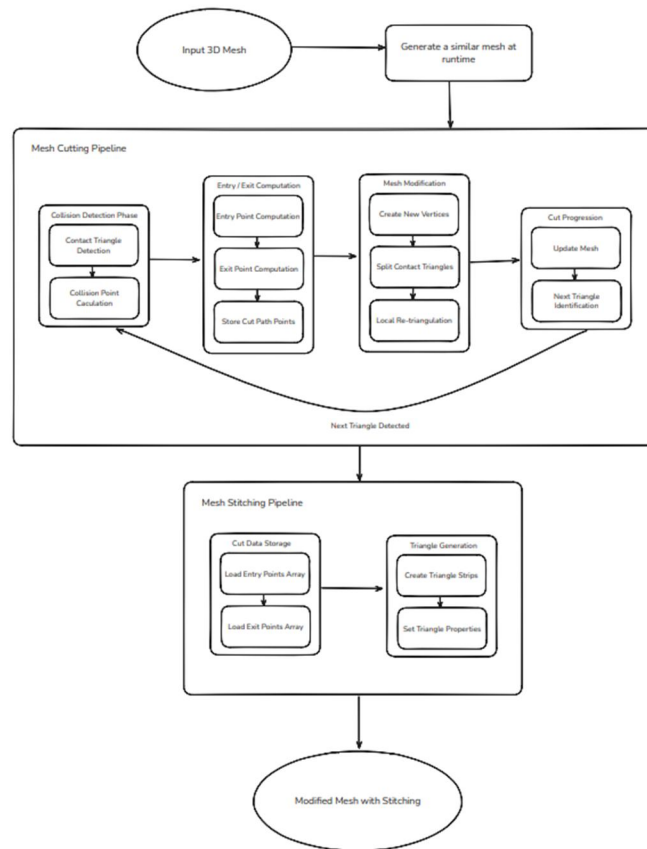


Fig 2 3D Mesh Cutting and Stitching Pipeline

Implementing an operation such as the above one in VR will require high-fidelity 3D models which are made of meshes - vertices & edges. The cutting & stitching process of the surgery demands algorithms that allow for dynamic mesh manipulations - Cutting & Stitching. This will allow for a more immersive experience compared to the traditional use of pre-cut meshes with little to no user involvement in the process. We have introduced a real-time mesh manipulation framework in this study (see Fig 2) that combines precision with flexibility for dynamic operations under the guidance of the user, marking a significant step up from static approaches toward the achievement of immersive, interactive VR in applications such as surgical training, 3D modeling, and gaming.

II. LITERATURE REVIEW

Interactive mesh manipulation is employed in a variety of leading-edge applications such as medical simulations, game development, and AR/VR experiences. Such systems require not only accuracy but also computational speed and real-time interactivity to simulate complex scenarios. Geometric algorithms have been developed to solve some of the bottlenecks faced with cut path definition, primitive removal, and re-meshing for runtime mesh manipulation. A generalized framework for interactive mesh cutting presented by [7] tackles the complexities of arbitrary cuts in virtual models. By enabling the simulation of tools like scalpels and scissors, it enhances real-time surgical simulation systems, supporting high-resolution, patient-specific models. The framework emphasizes computational efficiency and precision, critical for integrating realistic training and procedural simulations. These techniques are particularly beneficial in medical applications, where training environments must replicate real-world scenarios without compromising on interactivity or accuracy. Basic algorithms such as the Sutherland-Hodgman algorithm [8] form the basis of many mesh manipulation routines. Designed for efficient clipping convex polygons, the algorithm maintains precision in the implementation of boundary constraints and intersection logic. Such an algorithm is particularly useful in real-time physics simulation where collision detection and boundary management play important roles in slicing and clipping within interactive environments. Its applicability for maintaining the object integrity under any transformation makes it fit for games and all other applications in which accurate mesh modification is strictly required. Having established these foundations, [9] explores dynamic slicing algorithms for convex shapes as a fundamental component in the construction of deformable objects. These algorithms dramatically add to the realism in a simulation since they are capable of high-resolution interactions with user-driven changes. It fits into the fast-rising need for responsive and adaptive simulations both in gaming and training contexts. In medical applications, for instance, virtual surgery can be enhanced in learning and interaction with the real-time feedback and control during interaction. For even more complex scenarios like soft body simulations, [10] introduces a real-time deformation model by the name of position-based dynamics (PBD). Dynamic mesh modification techniques such as re-triangulation and refinement along boundary edges ensure accuracy and interactivity. This approach offers unrestricted cuts with frame rates suitable for real-time applications. It has the appeal of blending both realism and responsiveness in VR-based tools for surgical training, thus it is very valuable. Besides cutting mechanics, mesh simplification and manipulation has been pursued for improved user interaction and computations. [11] developed a strategy to directly manipulate a 3D mesh using an easy-to-use interface. The proposed method employing the edge collapse mesh simplification technique captures salient features of the mesh and allows direct deformation on those key points. In modelling and animation 3D, the nature of such modifications is often iterative. By providing real-time feedback and display, the method balances computational savings with detailed adjustments, achieving more demands for dynamic design workflows. Another area where geometric algorithms have made significant strides is in cloth simulation. [12] demonstrates a high-level method using Unity3D's Compute Shader for AR and VR applications. This technique makes use of the ACAD (adaptive constraint activation and deactivation) method in order to reduce computational loads, with stability and realistic behaviour under rapid movements. Parallel computation on the GPU allows the system to run computationally demanding tasks, such as cloth simulation, in high frame rates without a diminishment of user immersion into virtual environments. In order to enhance user interaction with 3D objects, [13] proposes a novel approach to interactive mesh cutting with virtual tools. At the cost of accuracy and flexibility, this allows such applications in the context of medical simulation and game development to be extremely responsive. Virtual tools in mesh manipulation enable intuitive cutting operations, thus providing an interactive but very accurate environment. These improvements are of critical importance for applications requiring precision, such as surgical training and virtual prototyping. Physics simulation is another domain where geometric algorithms are making an impact. [14] explores Verlet integration, a stable alternative to traditional Euler methods, for simulating dynamic systems like cloth, rigid bodies, and articulated figures. Unlike classical approaches, Verlet integration naturally ensures

stability by focusing on position-based algorithms, avoiding explicit velocity updates that often lead to instability. These contact handling methods, using projection methods, eliminate the need for intricate rewinding or penalty-based schemes and thus represent a robust solution for real-time collision detection and constraint satisfaction. It has indeed proven to be a practical application in game development, such as in the case of "Hitman: Codename 47."

These advancements in geometric algorithms are shaping the future of runtime mesh manipulation by addressing key challenges such as computational efficiency, interactivity and precision. From surgical simulations to gaming and AR/VR applications, it enables the modification of meshes in real time with high accuracy to unlock new possibilities. Whether it is in the form of advanced cutting frameworks, dynamic slicing techniques, or innovative physics simulations, such tools create realities that are realistic and deeply engaging for users. As the field evolves, these algorithms will no doubt provide a base for further innovation in the interactive and immersive technologies.

III. PROPOSED METHODOLOGY

In this section, we go over the approaches we took to solve the problem at hand. There are 3 approaches - re-triangulation, quad removal and verlet integration.

The mesh cutting and re-stitching algorithm serves to create dynamic and realistic interactions within 3D environments by allowing objects, like a "cutting tool", to slice through meshes. When the tool collides with the mesh, the algorithm precisely cuts along the contact points on the surface of the mesh that come in contact with the tool.

After cutting, the re-stitching process links vertices along the cut edge and precisely forms new triangles along the cut portion of the mesh to fill up the cut part, preserving continuity and ensuring that the mesh retains its original structure. Key to this approach are vector algebra and projection calculations, which enable precise, accurate cuts that follow the intended slicing path. To make the mesh cutting process work smoothly, a few key parts need to come together. These elements help set up the mesh so it can be accurately cut when the sword makes contact, while ensuring the mesh maintains its shape and structure.

- The initial step would be to set up the mesh, getting it ready for interaction by initialising essential components like vertices, triangles, colours and colliders which will be assigned to the mesh.
- Next comes the creation of the mesh where we define what the mesh looks like. Here we create vertices that outline the mesh in 3D space, and then link these points with triangles to form its surface. These triangles are like the skin of the mesh, giving it its visible shape and structure. This is then followed by addition of colliders to the mesh which help in detection of the collision with the "cutting tool" or the "stitching tool".
- To keep track of the mesh shape in real-time, we store its vertices and triangles in separate lists. The vertices list contains the 3D points that make up the mesh, while the triangles list defines which points are connected to create the faces.

A. Approach 1: Re-triangulation of Mesh

Cutting of the mesh

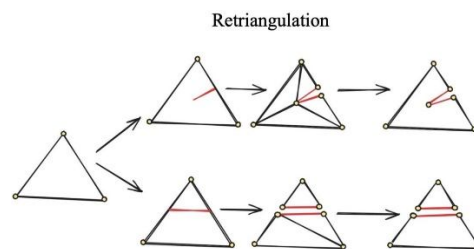


Fig. 3. Mesh Cutting using re-triangulation

We then set up trigger functions that get triggered on the detection of collision with the cutting tool and help in tracking the path the tool takes to get the cut on the mesh.

- Record the initial entry point i.e., the point at which the cut starts.
- Keep track of all the points along the edges of the triangles that come in contact with the cutting tool.
- Record the exit point of the tool at the end of the cut.
- Calculate positions at which the new vertices have to be created and also computes position and direction of the smaller triangles to be made.

During the transition of the cutting tool along the surface of the mesh the triangles that come in contact with the cutting tool are identified individually. To bring the cut the mesh individual triangles undergoes change in its structure according to one of the two conditions below:

- The cutting tool passes from one of the centres of the triangle and moves to one of the edges or passes along one of the edges of the triangle and stops at the centre of the triangle.
- The cutting tool passes from one edge to the other edge of the triangle.
- The triangle is split as shown in the fig 3, with each path representing how it is divided according to the conditions described above, respectively.

The new vertices that are created are stored separately in a temporary list, which later aids in the re-stitching process of the triangle or the mesh as a whole.

Stitching of the mesh

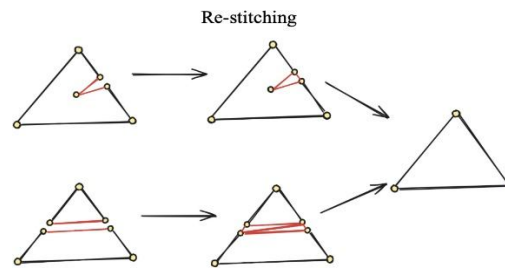


Fig. 4. Mesh Restitching using re-triangulation

For Stitching again, we set up a trigger function that get triggered on the detection of collision with the stitching tool and help in tracking the path the tool takes to stitch the mesh

- Track all newly created triangles, as illustrated in the fig 4, and modify their colour from transparent to the original colour of the mesh upon contact with the stitching tool, utilising shaders in Unity.
- Change the shader colour of the triangles from transparent to the original colour of the mesh when it comes in contact with the stitching tool.

B. Approach 2: Quad Removal method

This approach is built upon the approach we went through above. Instead of re-triangulation, we are removing the quad in the mesh to facilitate the cut.

Cutting of the mesh:

We then set up trigger functions that get triggered on the detection of collision with the cutting tool and help in tracking the path the tool takes to get the cut on the mesh.

- Identify the quadrilaterals (combination of two triangles) that come in contact with the cutting tool.
- Remove all the quadrilaterals that came in contact with the cutting tool.

Here the vertices of the quadrilaterals are stored in a separate temporary list, which helps further in the stitching process of the mesh.

Stitching of the mesh

The vertices from the temporary list mentioned above are considered and new transparent triangles are added to the mesh.

- Detect the collision of the “stitching tool” with the newly added transparent triangles.
- Change the colour of the shader attached to the newly created triangles from transparent colour to original colour of the mesh.

Drawback

While this method achieves the intended results to some extent, it lacks visual refinement. Additionally, the curves created along the cut lack smoothness and appear irregular, diminishing the overall aesthetic appeal.

C. Approach 3: Verlet Integration

$$x_{n+1} = 2x_n - x_{n-1} + A(x_n)\Delta t^2$$

Where:

- x_{n+1} is the next position.
- x_n is the current position.

- x_{n-1} is the previous position.
- $A(x_n)$ is the acceleration of the particle in the current position.
- Δt is the difference between time steps.

Basics

Verlet integration [9] works using a time step-based approach to the physics simulation. Instead of continuous time, calculations are based on a frame-by-frame manner. In each time step, it calculates the next position of the particle using its current position, previous position, and the current acceleration of the particle. It is used intensely when simulating molecular dynamics. It is quite stable since the velocity is implicitly given and consequently it is harder for velocity and position to come out of sync. Thereby allowing for efficient and stable soft body physics simulations.

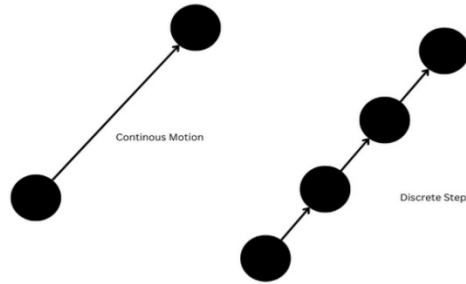


Fig. 5. Comparison of Continuous Motion vs. Discrete Steps in Verlet Integration

Setup

The mesh is setup with vertices as small node like elements, with the edges having a constraint setup between them - i.e. the max distance between any 2 nodes, as a result, there is constantly forces being applied on each node from all the directions, The Verlet formula with modifications of the distance constraints will look like

$$d^1 = x^2(t) - x^1(t)$$

$$d^2 = \|d^1\|^2$$

$$d_3 = d_2 - r^2 / d_2$$

$$x_1(t + \Delta t) = x_1(t) + 1/2 d_1 d_3$$

$$x_2(t + \Delta t) = x_2(t) - 1/2 d_1 d_3$$

Where:

- x_1 is the first node.
 - x_2 is the second node.
 - r is the fixed distance between the two nodes.
- d_1 is the vector from the first node to the second at any given time

Cutting

So, when user interacts with any of the nodes - causing it to be removed, then do to the dis-balance of the overall forces, causes the mesh to droop, thereby giving it a cloth like effect as seen below.

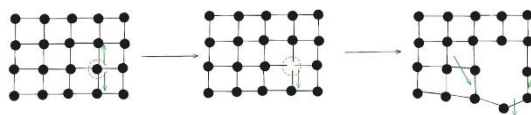


Fig. 6. Mesh Restitching using re-triangulation

Drawback

This method has many advantages. It is straightforward and easy to understand, and it needs very little space for storing necessary variables. But it has one downside: Its accuracy is not that high, but it's enough for simulating physics that may be visually appealing. Also, this method assumes that the nodes are all of the same size which may lead to inaccurate simulations. When it comes to stitching, generated nodes at stitched places may cause unpredictable results, given that each node is being acted upon by forces constantly and therefore it may not be feasible for the stitching.

D. Summary

The proposed methodology section describes an advanced real-time technique for mesh-cutting and stitching that allows complex interactions within 3D space. Based on the dual- pipeline approach, the method here ensures seamless integration of cut and re-stitching procedures that would be critical for maintaining runtime integrity and usability of the mesh. This pipeline is carefully designed for correct contact detection, dynamic creation of vertices, and local re-triangulation of the mesh to make each modification of the mesh as accurate and unobtrusive as possible. In contrast, the stitching pipeline will effectively restore the former state of the mesh after manipulation by using complex algorithms that will rebind cut edges to former glory in seamless visual continuity. This double approach will enhance not only the functionality of the mesh manipulation software but also extend the domain of use to very precise and highly interactive fields such as surgical simulations or complex design edition in computer- aided manufacturing.

IV. RESULTS

Five mesh manipulation methods, namely Quad Removal, Verlet Integration, only mesh cutting, Cut&Stitch for straight cut and curved cut, were evaluated.

A. Benchmarks

CPU Performance

- Verlet integration was the most unstable, jumping between very low (3 MB) and high (85 MB) usage
- Cut &Stitch methods (both types) were the most stable, staying steady around 70-75 MB
- Quad Removal and Mesh Cut showed some spikes but were generally more controlled than Verlet integration.

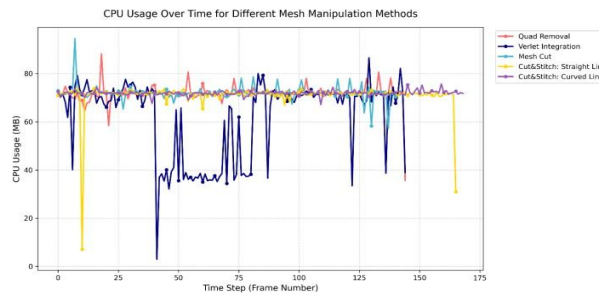


Fig. 7. CPU Usage

GPU Time

- Quad Removal was the slowest, needing 1,600ms per frame (about 1.6 seconds).
- All other methods were faster, using around 1,000ms per frame (1 second).
- Verlet Integration started slightly faster (900ms) but slowed down to match the others after frame 50.
- Most methods maintained consistent speeds throughout the test.

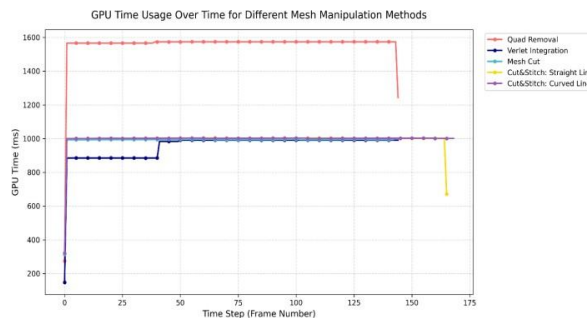


Fig. 8. GPU Time Usage

GPU Memory Usage

- Cut&Stitch methods and Mesh Cut steadily used more memory as time passed:
- Started at 660 MB.
- Curved Line version reached 735 MB.

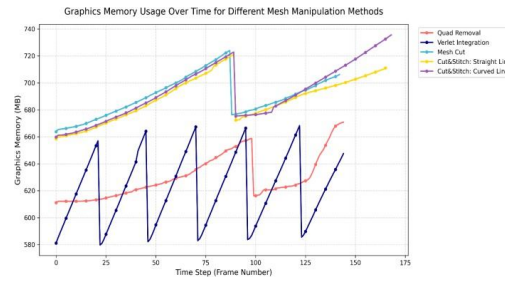


Fig. 9. Graphics Memory Usage

- Straight Line version reached 712 MB.
- Verlet Integration regularly cleaned up its memory, creating a saw-tooth pattern between 580-665 MB.
- Quad Removal was the most efficient, using the least memory (610-670 MB).

TABLE I: 3D RENDERING TECHNIQUE PERFORMANCE

Method	CPU Usage (MB)	GPU Time (ms)	Graphics Memory (MB)
Cut & Stitch: Curved Line			
Average	71.89	997.86	693.33
Std Dev	1.01	52.31	20.27
Cut & Stitch: Straight Line			
Average	71.04	995.86	688.37
Std Dev	6.01	58.51	N/A
Mesh Cut			
Average	72.00	989.72	689.74
Std Dev	3.15	56.87	15.07
Quad Removal			
Average	71.74	1560.69	630.01
Std Dev	4.04	111.00	16.13
Verlet Integration			
Average	61.82	955.40	621.23
Std Dev	16.19	82.40	24.84

A. Performance Trade-offs

The analysis reveals distinct trade-offs between methods:

- Verlet Integration: Offers superior GPU time and memory efficiency but at the cost of unstable CPU usage.
- Cut&Stitch methods: Provide the most stable CPU performance but require higher memory overhead.
- Quad Removal: Demonstrates moderate CPU stability and efficient memory usage but demands significantly more GPU time.
- Mesh Cut: Presents balanced performance across all metrics but shows higher memory requirements.

B. Visual Results

Here are some images which show how the cut and stitch made on 2d and 3d objects look.

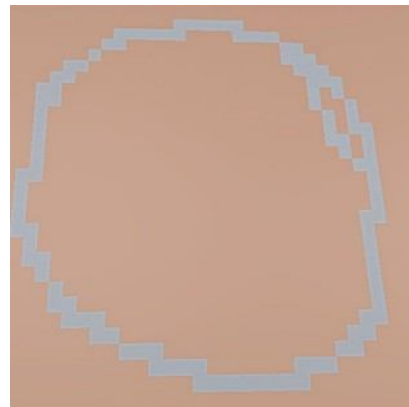


Fig. 10. Cut made on a 2D Mesh using Approach 1 – Circle Fig. 11. Cut made on a 2D Mesh using Approach 2 – Circle

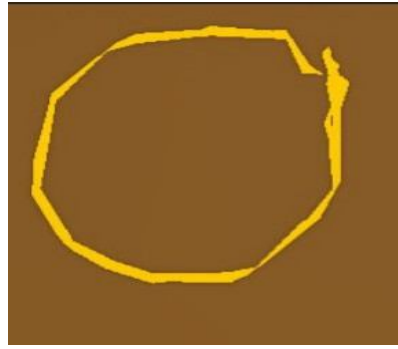


Fig. 12. Stitch made on a 2D Mesh using Approach 1 – Circle

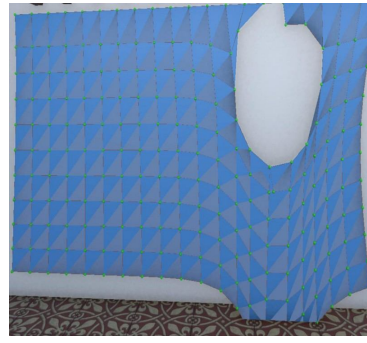


Fig. 13. Cut made on Verlet Mesh using Approach 3

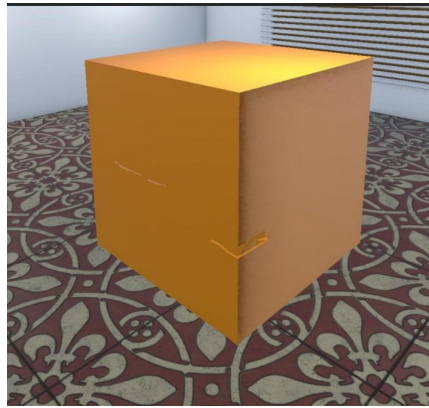


Fig. 14. Cut made on a Cube using Approach 1

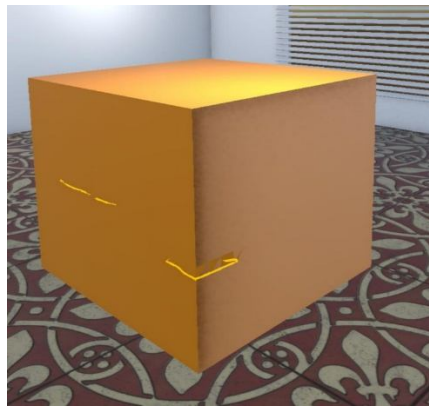


Fig. 15. Stitch made on a Cube using Approach 1

V. CONCLUSIONS

This paper describes how we have implemented a dual-pipeline system to address the critical challenges in handling meshes in real-time manipulation, focusing mainly on the tasks of mesh cutting and stitching. The Mesh Cutting Pipeline was engineered to make precise alterations to the mesh, through techniques such as contact detection, dynamic vertex creation, and local re-triangulation such that all cuts based on user interactions are precise. In the meantime, the ability of the Mesh Stitching Pipeline proved to be good in reconnecting pieces of the mesh cut, without any problem in preserving the structural integrity or topological consistency of the models. This hence implied that this was effective in realism and the interactivity of simulations in many domains, considering the achievements of the system in fulfilling its main goals. Nevertheless, the study also made public some weaknesses of the system, especially on the processing of complex mesh topologies and optimized computational performance toward real-time processing demands. These are challenging when there is a need for high-fidelity models; this also requires substantial computing resources in order to manipulate them without sacrificing performance, since their designs are intricate. Our approach enables runtime mesh manipulation in a general framework manner and focused on two very important processes: mesh cut and mesh stitch. In the Mesh Cutting Pipeline, it did the right cuts on the mesh by first using contact detection, vertex creation, and then re-triangulation. This pipeline handled integrity with dynamic cut advancements and a minimum visual distortion. The Mesh Stitching Pipeline cached the information of cut paths and generated connecting triangles so that visually the mesh restores after cuts. The cutting and stitching pipeline will be merged to offer a powerful solution for applications needing dynamic and interactive mesh manipulations.

FUTURE WORK

- Getting the actual contact point: One of the difficulties we faced while implementing this project is that we were not able to get the actual contact point of the mesh on which the cut had to be made. The method for detection for collisions is by the use of colliders which has an offset of ~1mm. Thus, further research in this direction to get the actual point for cutting/stitching will be more beneficial.

- Medical Simulation and Training: Through surgical simulations, trainees can practice incisions on virtual tissues that react realistically to cuts and stitches which will help them improve their skills without the risks that come with live patient surgery.
- Optimisation of Computational Performance: Reduction of the computational complexity during the re-triangulation phase to ensure smooth real-time performance can be worked on further and also exploring parallel processing techniques - GPU acceleration, to handle highly detailed meshes efficiently.
- Enhanced Handling of Complex Mesh Topologies: Emphasis should be given to the development of advanced techniques for manipulating hard-to-handle mesh structures, such as non-manifold edges and complex geometrical configurations. Introduce proper algorithms also for validation and repairing of meshes for curing common defects like uneven triangulations, gaps, and missing elements so that the system may effectively handle a variety of complex mesh inputs.
- Real time feedback: Implementation of real-time feedback as to whether a cut is made properly is one of the directions to be explored. The system can give directions to make the cut in a specified way and after the cut is done, verify it. This will be extremely helpful in adding more functionality to the Medical Simulation and Training mentioned above.
- Support for Varied Mesh Quality and Formats: Incorporating preprocessing pipelines to standardize the mesh quality which includes re-meshing, decimation, or subdivision techniques.
- Localised Level of Detail for Optimised Performance: Rendering higher number of polygons and increased triangulation at points of interest on the surgical body will help create an even more immersive experience.
- Implementation of Physics for better cut simulation: Integrating advanced physics-based soft body dynamics to simulate realistic tissue deformation and response during cutting.

ACKNOWLEDGMENT

This project was carried out using the resources available in the CAVE Labs, Center for IoT, PES University, Bengaluru. Alongside CSE department which provided us with the necessary Administrative and academic support.

REFERENCES

- [1] Desselle, M.R., R.A. Brown, A.R. James, M.J. Midwinter, S.K. Powell, and M.A. Woodruff. 2020. "Augmented and Virtual Reality in Surgery." *Computing in Science and Engineering* 22, no. 3 (May-June): 18-26. doi: 10.1109/MCSE.2020.2972822.
- [2] Covaciu, F., B. Gherman, G. Rus, C. Vaida, I. Zima, and D. Pislă. 2024. "Development of a Virtual Reality Simulator for Robotic-Assisted Laparoscopic Surgery." In *2024 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR)*, 1-6. Cluj-Napoca, Romania. doi: 10.1109/AQTR61889.2024.10554158.
- [3] Zhang, G. 2019. "Virtual Simulation for History Education." In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, 1646-1651. Osaka, Japan. doi: 10.1109/VR.2019.8797734.
- [4] Chan, C.S., J. Bogdanovic, and V. Kalivarapu. 2022. "Applying Immersive Virtual Reality for Remote Teaching Architectural History." *Education and Information Technologies* 27: 4365-4397. doi: 10.1007/s10639-021-10786-8.
- [5] Chen, G., X. Xie, Z. Yang, R. Deng, K. Huang, and C. Wang. 2023. "Development of a Virtual Reality Game for Cultural Heritage Education: The Voyage of 'Gotheborg'." In *2023 9th International Conference on Virtual Reality (ICVR)*, 531-535. Xianyang, China. doi: 10.1109/ICVR57957.2023.10169671.
- [6] Yeasmin, S., and L.A. Albabtain. 2020. "Implementation of a Virtual Reality Escape Room Game." In *2020 IEEE Graphics and Multimedia (GAME)*, 7-12. Kota Kinabalu, Malaysia. doi: 10.1109/GAME50158.2020.9315039.
- [7] Gaul, Randy. 2013. "Understanding Sutherland-Hodgman Clipping for Physics Engines."
- [8] Gaul, Randy. 2014. "How to Dynamically Slice a Convex Shape."
- [9] Khan, Lyudmila, Yoo-Joo Choi, and Min Hong. 2022. "Cutting Simulation in Unity 3D Using Position Based Dynamics with Various Refinement Levels." *Electronics* 11: 2139. doi:10.3390/electronics11142139.
- [10] Hua, S., Q. Zhong, and Q. Jiang. 2010. "Direct Manipulation of 3D Mesh Deformation." In *2010 Third International Symposium on Information Science and Engineering*, 202-206. Shanghai, China. doi: 10.1109/ISISE.2010.89.
- [11] Va, Hongly, Min-Hyung Choi, and Min Hong. 2021. "Real-Time Cloth Simulation Using Compute Shader in Unity3D for AR/VR Contents." *Applied Sciences* 11, no. 17: 8255.
- [12] Bruyns, Cynthia, S. Senger, A. Menon, K. Montgomery, S. Wildermuth, and R. Boyle. 2002. "A Survey of Interactive Mesh-Cutting Techniques and a New Method for Implementing Generalized Interactive Mesh Cutting Using Virtual Tools." *Journal of Visualization and Computer Animation* 13: 21-42. doi: 10.1002/vis.275.
- [13] Jakobsen, Thomas. 2001. "Advanced Character Physics." In *Game Developers Conference Proceedings*.
- [14] Jakobsen, Thomas P. "Advanced Character Physics." (2003).