VR-Doh: Hands-on 3D Modeling in Virtual Reality

Zhaofeng Luo*
Carnegie Mellon University
USA
Peking University
China
roushelfy@stu.pku.edu.cn

Zhitong Cui*
Carnegie Mellon University
USA
Zhejiang University
China
zhitongcui@zju.edu.cn

Shijian Luo
Zhejiang University
China
sjluo@zju.edu.cn

Mengyu Chu
Peking University
China
mchu@pku.edu.cn

Minchen Li
Carnegie Mellon University
USA
minchernl@gmail.com

Figure 1: In the first row, the vase is deformed using our VR-Doh system through direct hand interaction, followed by inserting and merging a bouquet of roses into it. Using hand gestures, we then bend and refine the roses, repeating the process across three stems. In the second row, the garden scene evolves as we remove the original vase, deform the table using tools, and finally place the newly decorated vase with roses as the centerpiece.

ABSTRACT

We present VR-Doh, a hands-on 3D modeling system designed for creating and manipulating elastoplastic objects in virtual reality (VR). The system employs the Material Point Method (MPM) for simulating realistic large deformations and incorporates optimized Gaussian Splatting for seamless rendering. With direct, hand-based interactions, users can naturally sculpt, deform, and edit objects interactively. To achieve real-time performance, we developed localized simulation techniques, optimized collision handling, and separated appearance and physical representations, ensuring smooth and responsive user interaction. The system supports both freeform creation and precise adjustments, catering to diverse modeling tasks. A user study involving novice and experienced users highlights the system's intuitive design, immersive feedback, and creative potential. Compared to traditional geometry-based modeling tools, our approach offers improved accessibility and natural interaction in specific contexts.

CCS CONCEPTS

• **Computing methodologies** → **Virtual reality**; *Point-based models.*

KEYWORDS

Virtual Reality, Human-Computer Interactions, 3D Modeling, Elastoplasticity Simulaton, Material Point Method

*Both authors contributed equally to this research.

1 INTRODUCTION

As the demand for digital content continues to grow, there is an increasing need for higher quality, greater quantity, and improved efficiency in 3D content creation. Traditional 3D modeling tools impose significant barriers to entry for novice users. These tools often require careful observation of real-world objects, mastery of specialized techniques, and iterative fine-tuning to achieve the desired outcome. This reliance on professional expertise limits accessibility for broader audiences in high-quality content creation.

Meanwhile, 3D modeling in Virtual Reality (VR) has become increasingly popular due to its ability to provide immersion and depth, thereby enhancing designers' creativity and collaboration [Rosales et al. 2019; Yu et al. 2021]. Commercial tools like ShapeLab [2024] and GravitySketch [2024] have advanced this vision. However, their approaches are still limited to procedural methods, such as drawing curves to form surfaces from scratch, which are restricted to geometric techniques and require significant skills.

In contrast, physics-aware shape modeling presents a novel paradigm that more closely aligns with human intuition and natural interactions, where shape deformations are caused by contact forces or boundary conditions defined by specific material properties [Fang et al. 2021]. In VR, the use of hands as an input modality is particularly compelling, as it closely mimics how users naturally manipulate objects in the physical world, such as clay, thereby leveraging users' prior experience in creating real-world objects [Arora et al. 2019; Pihuit et al. 2008; Schulz et al. 2019].

Building on these foundational advantages, our work delves into the integration of physical simulations within VR to create an intuitive and efficient hands-on 3D modeling experience grounded in realistic physics. The primary objective is to enable users to manipulate and deform virtual objects with a high degree of realism, leveraging direct hand-based contact and providing interactive feedback. To facilitate large-scale deformations and seamless material coupling, we utilize the Material Point Method [Hu et al. 2018; Jiang et al. 2016] for elastoplastic solid simulation, a robust technique that ensures the accuracy of complex material interactions. For users engaged in editing imported 3D models, we employ 3D Gaussian Splatting (3D GS) [Kerbl et al. 2023] to deliver highly realistic rendering, while also enabling shape creation from scratch through surface reconstruction methods. We enhance real-time responsiveness by optimizing hand-object collision handling and localized simulation around hands, and improve simulation efficiency by using fewer particles to drive more Gaussian splats, maintaining rendering quality. To further enhance the expressivity and versatility of the created shapes, we integrate a suite of tools that facilitate diverse and controllable deformations, providing users with creative flexibility. Additionally, we conducted a user study to evaluate the system's usability and effectiveness. The insights gained from this study offer a deeper understanding of how users leverage their hands for physics-based shape modeling, helping to inform future iterations and improvements.

The key advantage of our approach is that it makes the 3D modeling process in VR significantly more intuitive and efficient for novice users. Compared with traditional geometry-based approaches, our system provides physics-based deformations, making the modeling process more realistic. Hands-on input lowers the barrier to modeling operations, aiding in a more flexible selection of the desired operation area compared to using a mouse, thereby improving efficiency and enhancing user immersion. Users are able to predict the outcomes of their actions based on their everyday experiences, thereby reducing the need for specialized skills. Meanwhile, a diverse set of tools is offered, allowing users to begin creating immediately, without any prior training. Our contributions are summarized as follows:

- *A novel physics-based geometric modeling paradigm*: We introduce a VR modeling system with realistic physics simulations, offering immersive, intuitive, and creativity-enhancing workflows.
- *Direct and intuitive modeling with natural interactions*: Our system leverages hand-based input and supports a wide range of tools, enabling tasks from basic shape manipulations to complex modeling operations.
- *Real-time support enabled by technical innovations*: The system incorporates optimization techniques such as localized simulation techniques, separated appearance and physical representations, and hand-object collision detection, ensuring smooth real-time interaction.
- *User Study Findings & Experimental validation*: We conduct a user study and comprehensive experiments to evaluate the system's usability and effectiveness, providing insights into how users interact with physics-based VR modeling systems.

2 RELATED WORK

Virtual Clay Modeling. Virtual modeling of clay-like materials using dexterous input offers significant advantages over traditional 3D modeling, including high expressivity and a lower learning curve, effectively replicating the tactile experience of working with physical clay [Chatterjee 2024; Sheng et al. 2006]. Despite these benefits, the computational demands of simulating dexterous hand-object interactions have limited the integration of virtual clay modeling into VR environments, with most existing implementations constrained to 2D screens. Early approaches to virtual clay modeling utilized dynamic subdivision solids [McDonnell et al. 2001], followed by the application of plasticity models to capture essential physical properties of real clay, such as mass conservation and surface tension effects [Dewaele and Cani 2004]. To improve interactivity, Barreiro et al. [2021] introduced a particle-based viscoplasticity model with ultrasound haptic feedback, although the fidelity of finger-based interactions remained insufficient for detailed 3D modeling tasks. Additionally, physical proxies have been employed to enhance the virtual clay experience [Marner and Thomas 2010; Sheng et al. 2006], such as combining direct finger input with deformable physical devices to enable clay-like sculpting operations, including deforming, smoothing, pasting, and extruding [Sheng et al. 2006]. Efforts to adapt virtual clay modeling to VR environments have been limited. For example, Moo-Young et al. [2021] made preliminary attempts to integrate these techniques into VR; however, their approach lacked the robustness required for intricate hand interactions. To address this gap, our work introduces a realistic, hands-on modeling framework for VR, leveraging the Material Point Method [Jiang et al. 2016] and efficient hand-object contact handling to enable intuitive and detailed clay-like modeling in virtual environments.

Physics-Aware Interactions in VR. Physics-aware hand-object interactions in VR have been extensively studied, enabling users to engage with various virtual phenomena such as particle-based animations [Arora et al. 2019], deformable objects [Jiang et al. 2024; Moo-Young et al. 2021], fluids [Deng et al. 2023], and content creation [Eroglu et al. 2018]. These approaches often utilize hand-tracking data to map human hand motions to rigged virtual hand models, enabling dynamic interactions in immersive environments. For example, Höll et al. [2018]; Kumar and Todorov [2015]; Lougiakis et al. [2024] use tracked hand information to fully define the virtual hand geometry in each frame while incorporating frictional contact, achieving realistic hand-rigid-body interactions. Building on this, Jacobs and Froehlich [2011]; Smith et al. [2020]; Verschoor et al. [2018] simulate deformable hands using nonlinear soft tissue models, which, combined with frictional contact handling, allow for more natural and realistic interactions. While these works primarily focus on interactions between hands and rigid objects, research on hand-deformable object interactions has also gained traction. For instance, VR-GS [Jiang et al. 2024] facilitates human-centered interaction with physically simulated 3D content represented using 3D Gaussian Splatting [Kerbl et al. 2023]. However, these efforts largely emphasize physics-aware interactions rather than shape editing, which holds significant promise for creative tasks such as animation control [Arora et al. 2019]. To address this gap, we focus on physics-based 3D modeling in VR, leveraging both contact- and gesture-driven interactions to enable intuitive manipulation of deformable objects for creative applications.

3D Modeling Tools in VR. Creativity support tools for 3D modeling in VR encompass different categories. Among drawing-based modeling tools, Surface Drawing [Schkolne et al. 2001] stands as a pioneering approach, enabling users to draw strokes in space using hand motions and edit 3D models with tangible tools. SurfaceBrush [Rosales et al. 2019] and AdaptiBrush [Rosales et al. 2021] convert dense collections of artist-drawn stroke ribbons into user-intended manifold free-form 3D surfaces. Cassie [Yu et al. 2021] is a conceptual modeling system in VR that leverages freehand mid-air sketching and a novel 3D optimization framework to create a connected curve network. Besides, inspired by FiberMesh [Nealen et al. 2007], RodMesh [Schulz et al. 2019] replaces curve drawing with two-handed bending and stretching of virtual rods, allowing users to define outline shapes that are subsequently inflated into manifold mesh surfaces. Commercial tools, such as ShapeLab [2024], Substance 3D Modeler [Substance3d 2024], and Kodon [2024], mainly focus on geometric editing through polygonal or voxel-based sculpting. These tools allow users to manipulate vertex regions and apply effects with dynamic topology updates. Notably, GravitySketch [2024] introduces subdivision modeling, enabling users to create complex meshes from simple base topology while maintaining smoothness. Building upon prior work, we aim to further reduce the demand for professional 3D modeling expertise with physics-aware hands-on shape modeling, making 3D modeling in VR more accessible to novice users.

3 DESIGN RATIONALE

To make our hands-on modeling tool more intuitive, user-friendly, and efficient for novice users, we propose the following requirements and functionalities:

(1) **Realistic Physics Simulation with Large Deformations:** This provides users a more intuitive understanding of the expected deformations their models will undergo based on their inputs compared to traditional tools, allowing for faster and more accurate modeling that better aligns with the user's design intention.
(2) **Real-Time Performance:** Since the modeling process takes place in a VR environment, real-time frame rate and smooth rendering are crucial factors. Low frame rates can lead to VR motion sickness[1], significantly impairing the user experience. A real-time system ensures a seamless and immersive interaction.
(3) **Rich Input Methods:** When creating 3D models in real life, users often not only manipulate the material directly with their hands but also use various tools such as chisels and knives to shape the object. These tools allow for more diverse, refined, and controlled deformations. Supporting a wide range of tools, including customizable ones, can greatly enhance the efficiency and precision of modeling.
(4) **Hierarchical Modeling Process:** Complex models are often composed of multiple parts, with both overall structure and detailed components. An efficient modeling process should allow users to create individual components, which can be reused, copied, or merged to form a complete model. This approach keeps the workflow organized while maximizing the reuse of existing resources.
(5) **Intuitive Spatial Interface Design:** A clear and easy-to-use user interface (UI) is essential for guiding users through the modeling process without overwhelming them with complexity. An intuitive UI helps users focus on creativity and interaction, rather than learning the tool itself.

We believe that simultaneously achieving all these goals presents a significant challenge. The more accurate the simulation and the richer the input methods, the greater the demand on computational resources. In the following sections, we will explain how we built such a hands-on modeling tool in VR, including the overall system architecture, methodologies, and the technical trade-offs and innovations we employed.

4 TECHNICAL BACKGROUND

Based on our design rationale, we selected PhysGaussian [Xie et al. 2024] as the foundation of our system to enable real-time simulation of elastoplastic objects and achieve photorealistic rendering. This section provides the technical background on the Moving Least Squares (MLS) Material Point Method (MPM) [Hu et al. 2018] and 3D Gaussian Splatting [Kerbl et al. 2023], which PhysGaussian builds upon. However, directly applying these existing techniques does not fully address the challenges in developing our system. To overcome these limitations, we introduced key innovations that are critical to making our system functional and effective. These innovations will be detailed in section 5.

Algorithm 1 Sim_substep(i)

1: Particle_to_Grid()
2: Update_Grid_Velocity(i)
3: Grid_to_Particle()
4: Adjust_Particle(i) ▷ See subsection 5.2
5: Apply_Plasticity()

4.1 MLS-MPM

The Moving Least Squares Material Point Method (MLS-MPM) is a robust hybrid Lagrangian-Eulerian approach for simulating multi-material phenomena, enabling the efficient coupling and simulation of complex materials. In MPM, Lagrangian particles are used to track the geometry and material properties of the simulated object, while an Eulerian background grid facilitates the computation of forces and time integration. This dual representation allows the method to seamlessly handle large deformations and dynamic interactions by frequently transferring physical quantities, such as mass and momentum, between the particles and the grid [Jiang et al. 2016; Sulsky et al. 1995].

In MPM simulations, each time step (Algorithm 1) begins with the `Particle_to_Grid` operation, which transfers particles' mass and momentum to adjacent grid nodes using the following equations:

$$m_i^n = \sum_p w_{i,p} m_p, \tag{1}$$

$$(mv)_i^n = \sum_p w_{i,p}\{m_p v_p + [m_p C_p^n - E_p^n]\left(x_i - x_p^n\right)\}, \tag{2}$$

The elasticity stress term E is computed as:

$$E_p^n = -\frac{4\Delta t}{\Delta x^2}\sum_p V_0 P_p^n \left(F_p^n\right)^T, \tag{3}$$

To calculate the first Piola-Kirchoff stress P_p^n, two forms are available: StVK and Neo-Hookean, defined respectively as:

$$P_p^n = 2\mu\left(F_p^n - UW\right) + \lambda\left(J_p^n - 1\right)J_p^n\left(F_p^n\right)^{-T}, \tag{4}$$

$$P_p^n = \mu\left(F_p^n \cdot F_p^{nT}\right) + \mathbf{I}\cdot(\lambda\log(J) - \mu), \tag{5}$$

Here, μ and λ are the Lamé constants, representing the shear stiffness and incompressibility of the material. The terms W and U are obtained through singular value decomposition of the deformation gradient tensor: $F = U\Sigma W$.

Next, the grid velocities are updated to incorporate external forces, as given by

$$(mv)_i^n += F_{extern}\cdot\Delta t, \tag{6}$$

and to enforce boundary conditions. External forces such as gravity, air damping, etc. can be applied. Boundary conditions are defined as either slip or sticky. In slip boundaries, the normal component of the grid velocity near the boundary is set to zero, while for sticky boundaries, both tangential and normal components are set to zero.

[1]VR motion sickness: a condition where users experience nausea and discomfort due to mismatches between visual and physical motion

After updating grid velocities, the `Grid_to_Particle` operation is performed, where particle velocities v and affine transformation matrices C are updated based on neighboring grid velocities. The updated particle states are defined as:

$$v_p^{n+1} = \sum_p w_{i,p} v_i^{n+1}, \tag{7}$$

$$C_p^{n+1} = \frac{\Delta t}{\Delta x^2}\sum_p w_{i,p} v_i^{n+1}\left(x_i - x_p^n\right)^T, \tag{8}$$

$$x_p^{n+1} = x_p^n + \Delta t v_i^{n+1}. \tag{9}$$

Since boundary conditions are enforced at the grid level, particles may still penetrate solid boundaries, leading to inaccuracies, particularly for fine boundary geometries that the grid cannot adequately resolve. To address these challenges, our system introduces a particle-level boundary treatment through an additional `Adjust_Particle` step, enabling more precise handling of particle-boundary interactions. More details are provided in subsection 5.2.

Plasticity is then applied to the updated deformation gradient $F_p^{n+1} = \left(I + \Delta t C_p^n\right)F_p^n$. Three types of plasticity models are often applied: Drucker-Prager, Von Mises, and clamp-based plasticity, each with specific expressions. For Drucker-Prager plasticity:

$$Z(F_p)_{druncker} = \begin{cases} F_p, & \delta\gamma \le 0 \\ U\exp\left(\epsilon - \delta\gamma\frac{\hat{\epsilon}}{\|\hat{\epsilon}\|}\right)V^T, & \text{otherwise} \end{cases} \tag{10}$$

with

$$\delta\gamma = \begin{cases} \|\hat{\epsilon}\|, & \mathrm{tr}(\epsilon) > 0 \\ \|\hat{\epsilon}\| + \alpha\frac{d\lambda+2\mu}{2\mu}\mathrm{tr}(\epsilon), & \text{otherwise} \end{cases} \tag{11}$$

For Von Mises plasticity:

$$Z(F_p)_{von} = \begin{cases} F_p, & \delta\gamma \le 0 \\ U\exp\left(\epsilon - \delta\gamma\frac{\hat{\epsilon}}{\|\hat{\epsilon}\|}\right)V^T, & \text{otherwise} \end{cases} \tag{12}$$

where

$$\delta\gamma = \|\hat{\epsilon}\| - \frac{\tau_Y}{2\mu} \tag{13}$$

Clamp-based plasticity is defined as:

$$Z(F_p)_{clamp} = U\cdot\mathrm{Clamp}(\Sigma, \Sigma_{\min}, \Sigma_{\max})\cdot V^T \tag{14}$$

4.2 3D Gaussian Splatting

3D Gaussian Splatting (3DGS) is a rendering method that employs a set of unstructured 3D Gaussian kernels to efficiently represent and render a scene. Each Gaussian kernel is characterized by its center, opacity, covariance matrix, and spherical harmonic coefficients. In contrast to Neural Radiance Fields (NeRFs), which render views by emitting rays, GS projects these 3D Gaussians directly onto a 2D image plane, enabling real-time and highly efficient rendering. Therefore, compared to the implicit representation of NeRF, 3DGS is more advantageous for editing tasks [Gao et al. 2024]. The final color of each pixel is computed as a weighted sum of these projections:

$$C = \sum_{k\in P}\alpha_k SH(d_k; C_k)\prod_{j=1}^{k-1}(1-\alpha_j), \tag{15}$$

where α_k represents the effective opacities, $SH(d_k; C_k)$ denotes the spherical harmonics of view direction d_k at Gaussian k, and P is the set of all Gaussians contributing to the pixel color. The explicit representation of the scene accelerates training and rendering speeds,

and it allows for direct manipulation of the Gaussians for dynamic adjustments.

4.3 PhysGaussian

GS primarily focuses on the visual appearance of the scene without incorporating physical properties, such as deformation or dynamics. PhysGaussian extends the capabilities of GS by integrating physical simulation directly into the Gaussian representation, allowing for a unified simulation-rendering approach. The underlying mechanics are grounded in the continuum mechanics principles, where the evolution of the deformation map $x = \phi(X, t)$ between material space Ω_0 and world space Ω_t is used to simulate material behavior over time.

In PhysGaussian, each Gaussian kernel is treated as a discrete particle that deforms according to a local affine transformation. The updated Gaussian under deformation becomes:

$$G_p(x,t) = e^{-\frac{1}{2}(x-x_p)^T (F_p A_p F_p^T)^{-1}(x-x_p)}, \tag{16}$$

where $x_p(t)$ is the transformed center of the Gaussian, and $F_p A_p F_p^T$ is the covariance matrix updated by the deformation gradient F_p. This transformation effectively integrates both kinematics and visual rendering, providing seamless transitions between simulation and visualization.

This transformation naturally provides a time-dependent version of x_p and A_p for the 3D GS framework:

$$x_p(t) = \phi(X_p, t), \tag{17}$$

$$a_p(t) = F_p(t) A_p F_p(t)^T. \tag{18}$$

5 METHOD

Algorithm 2 Overall Framework

```
while True do
    Update_Input()                    ▷ See subsection 5.1
    for i ← 1 to substeps do
        Sim_substep(i)                ▷ See subsection 4.1
    end for
    Update_Rendering()                ▷ See subsection 5.3
end while
```

Our system integrates intuitive user interactions, real-time simulation, and high-quality rendering to enable hands-on 3D modeling in VR. The system supports natural and flexible interaction modes, including direct hand manipulation, tool-based operations, and gesture inputs, designed to mimic real-world actions and reduce the learning curve (subsection 5.1). For simulation, we build on the Material Point Method (MPM) [Hu et al. 2018], which accurately simulates elastoplastic materials and large deformations. To address its computational challenges and ensure real-time performance, we introduce key innovations such as particle-level boundary treatment and local simulation optimizations (subsection 5.2). For rendering, the system combines Gaussian Splatting [Kerbl et al. 2023] for high visual fidelity and Surface Rendering using the Marching Cubes algorithm [Lorensen and Cline 1998] for creating new models. These rendering techniques are adapted to handle dynamic updates during modeling, ensuring smooth transitions and consistency during deformations (subsection 5.3). The overall system framework is shown in Figure 2, and subsequent subsections detail its core components and innovations.

5.1 Hands-on 3D Modeling

Algorithm 3 Update_Input()

```
Get_Joint_Positions()
Compute_Fields(SDF, Normal, Velocity)
Get_Gesture_Input()
Compute_Force_Field()
```

To empower users to create or modify 3D models while aligning more effectively with their operational expectations, we support two complementary input approaches, i.e., *contact-based modeling* and *mid-air gestural modeling*. The former modifies the shape of an object through direct contact using the hand or tools, while the latter alters the shape by selecting and manipulating a part of the object using pinch gestures.

Contact-based Modeling. To enable efficient contact handling between hands and tools acting on the objects for interactive feedback, we approximate the shape of hands and integrated modeling tools by medial axis transform and its corresponding medial mesh [Faraj et al. 2013; Stolpner et al. 2011]. The medial axis of a 3D model consists of a set of points that are centers of maximally inscribed spheres, i.e., the medial spheres, touching the boundary with at least two contacts. By incorporating the radius information along the medial axis, there exists a unique medial axis transform representing the original 3D model losslessly. The medial axis transform essentially forms a collection of linearly interpolated spheres or medial primitives, which are either a medial cone or a medial slab [Sun et al. 2015] by two or three vertices on the medial mesh. In our implementation, we use the quadratic error metric as in Q-MAT [Li et al. 2015] for acquiring the medial mesh and medial primitives of the original shapes of hands, and tools. Specifically, one hand shape can be approximated by 76 medial cones and 28 medial slabs (Figure 3). When handling the collision between hands and objects during simulation, we refer to the method in Medial IPC [Lan et al. 2021] to calculate the distance between the grid node and medial primitives. In addition, we use the bounding box of medial primitives to construct a spatial hash table for collision detection. This approach accelerates the computation of the signed distance field (SDF) when handling collisions between the deformable objects and hands.

In addition to direct hand manipulation for shape modeling, we provide various auxiliary tools such as a planar slab, a rod, and scissors, to facilitate more diverse and controlled deformations, akin to the creative processes observed in the physical world (Figure 4). Upon selection, each tool attaches to a specific joint, enabling users to manipulate it through hand movements. Each hand operates independently, enabling seamless switching between different tools.

In the hand tracking process, the position and rotation of each medial primitive are determined by the two nearest joints of the hand skeleton. To enhance the stability of hand tracking, we apply a moving least-squares kernel to interpolate the tracking data from the past 0.2 seconds, with weights determined by frame duration.

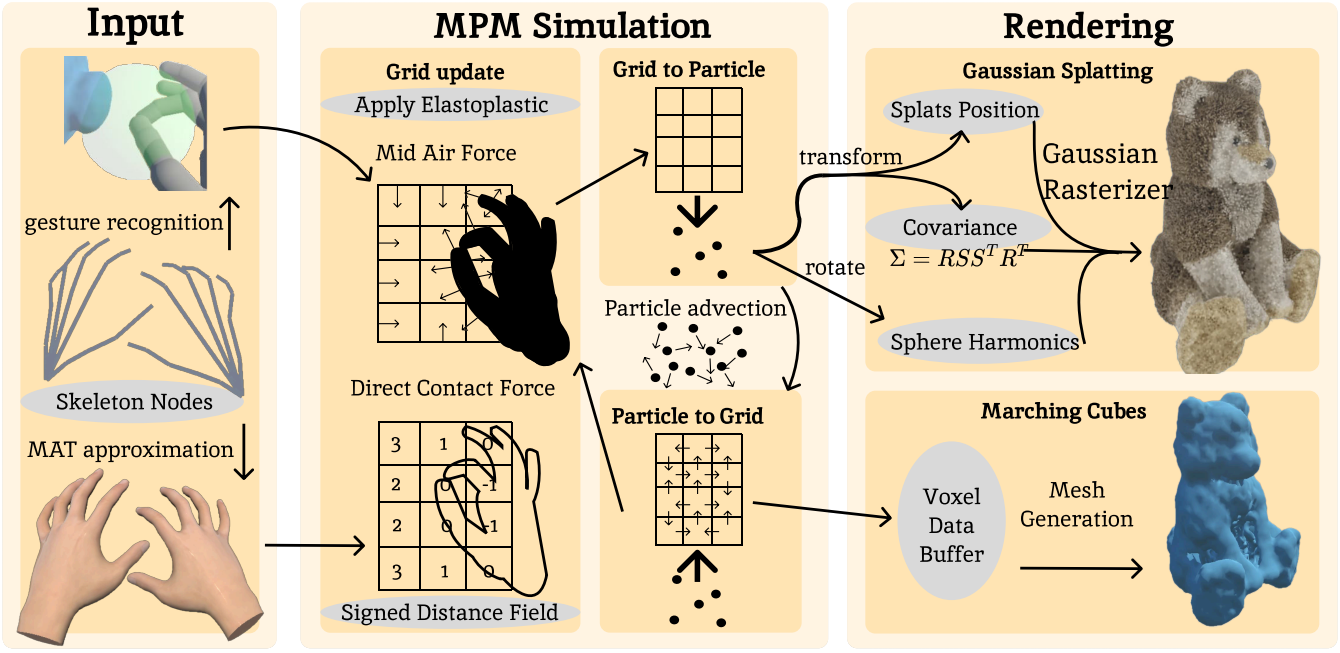

Figure 2: VR-Doh Pipeline Overview. Our interactive system enables hands-on modeling using both hand and basic tools. We adapt and accelerate MPM-based physics simulations for shape modeling on VR devices. We flexibly support rendering with both Gaussian Splatting and mesh representations, ensuring wide applicability.

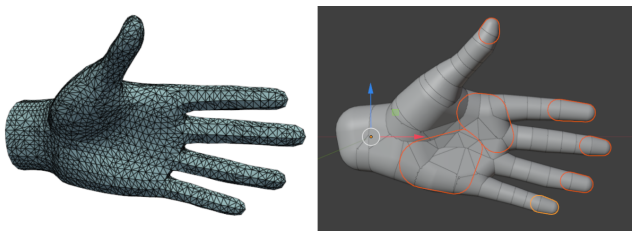

Figure 3: Medial axis approximation of hand geometry.

Figure 4: Shape modeling tools supported in VR-Doh. From left to right are the plate, the rod, the scissors, and the cone.

This ensures consistent and reliable hand-tracking performance across varying frame rates. Overall, our efficient contact-handling approaches enable interactive modeling operations on deformable objects, such as pinching, squeezing, bending, and folding, while avoiding noticeable visual artifacts. Note that we use the hand texture made by Pohl and Mottelson [2022].

Mid-air Gestural Modeling. Mid-air gestural input is advantageous when users face difficulties selecting the desired editing area on an object using contact-based modeling, which helps to avoid unintended changes to the object. We provide a mid-air pinch gesture (where the tip of the thumb touches the tip of the middle finger), enabling users to perform operations such as stretching or twisting materials by applying a force field to MPM particles. A blue highlight is used to visualize the selection area and the applied force field, providing intuitive visual feedback. Users can further refine the radius of the selection area and adjust the force ratio for more precise control, as shown in Figure 9.

Other Editing Operations. Our user interface also provides common operations to ensure the usability of our system. All objects in space can be grabbed and moved either through an external simulation domain bounding box or by forming a ray pointing to the bounding box via hand pinching. When both hands grasp the object's bounding box and move outward or inward, the object can be visually scaled up or down, without physical changes. Scaling allows for either local or global editing of the object: local editing is used for fine modeling, while global editing is used for overall shaping. Similar to common 3D modeling tools, the system integrates operations such as *Merge*, *Copy*, *Reset*, and *Delete* the modeled objects.

5.2 Simulation

Our system leverages MLS-MPM, as detailed in subsection 4.1, to handle elastoplastic simulations. However, straightforward implementations face significant challenges in achieving the accuracy

and real-time performance required for interactive modeling applications. To address these limitations, we propose three key enhancements: (1) *Grid-Level Boundary Forces*, which use customized pressure and friction forces to improve the accuracy of particle-boundary interactions; (2) *Particle-Level Boundary Projection*, which corrects particle positions and velocities to ensure particles remain outside solid boundaries; and (3) *Localized Simulation*, which dynamically restricts computations to user-interaction regions, significantly improving computational efficiency while maintaining responsiveness.

Grid-Level Boundary Forces. The standard boundary condition enforcement in MPM, which uses grid velocity projection near the boundary, struggles to provide accurate hand-object interactions. This limitation arises from penetrations and inaccuracies in resolving the boundary geometry, particularly when small grid resolutions are used to maintain real-time performance. Such resolutions fail to adequately capture the fine geometric details of the hand and modeling tools. To address these challenges, we introduce customized pressure and friction forces at the grid level to improve accuracy.

Pressure Force: The pressure force F_p is proportional to the penetration depth of the grid node inside the contact geometry, with the force vector directed towards the closest point on the contact boundary. Mathematically, this can be expressed as:

$$\mathbf{F}_p = -k_p \cdot d_i \cdot \hat{\mathbf{n}}$$

where k_p is a proportionality constant controlling the intensity of the pressure, d_i is the penetration depth of the grid node i, and $\hat{\mathbf{n}}$ is the unit vector pointing to the closest point on the contact boundary. This force tries to push particles away from the boundary to avoid penetration.

Friction Force: The friction force F_f is proportional to the pressure force F_p and acts in the direction opposite to the relative motion. It can be formulated as:

$$\mathbf{F}_f = -\mu \cdot |\mathbf{F}_p| \cdot \hat{\mathbf{t}}$$

where μ is the friction coefficient and $\hat{\mathbf{t}}$ is the unit vector in the direction of relative motion along the boundary surface. This friction force opposes the sliding motion, enhancing stability and control over particle movement at the boundary.

By replacing the standard boundary condition enforcement in MPM near the hands and modeling tools with these customized forces applied during the `Update_Grid_Velocity` phase, we achieve more effective particle repulsion within the contact boundary. The use of controllable stiffness in these forces mimics the realistic effects of pressure and friction, while precomputing the SDF of the boundaries ensures computational efficiency.

Particle-Level Boundary Projection. Despite the application of grid-level boundary forces, particles may still penetrate solid boundaries due to limitations in grid resolution. To address this issue, we introduce a particle-level boundary projection step. This approach adjusts the position and velocity of any particles that remain inside the boundary after their positions are updated based on the grid velocities.

The positions and velocities of penetrating particles are corrected as follows:

$$x_p = x_p + D, \tag{19}$$

$$v_p = v_{\text{boundary}}, \tag{20}$$

where D represents the distance vector to the boundary, and v_{boundary} is the velocity at the boundary. This step ensures particles remain outside the boundary and adhere to the geometry of the boundary.

Localized Simulation. While MPM excels at handling large deformations, real-time simulation is constrained by the computational limitations of off-the-shelf hardware, which can typically support only around 100K particles. However, large-scale Gaussian splatting scenes often exceed 500K particles, making it infeasible to simulate the entire scene in real time. To address this, we propose a localized simulation approach tailored to user interactions. Since the primary input comes from hand tracking, our system restricts the simulation to a cubic region centered around the user's hands, keeping particles outside this region stationary. This strategy significantly reduces the computational load, enabling higher frame rates without compromising the user experience. The simulation region dynamically follows the user's hand movements, ensuring that any area the user can interact with remains fully responsive and interactive.

5.3 Rendering

Algorithm 4 Update_Rendering()

```
if Using_Gaussian_Splatting then
    Update_Gaussian_Data()
else if Using_Marching_Cubes then
    Update_Marching_Cubes_Data()
end if
```

Improved Gaussian Splatting for Large Deformations. The original GS is completely static, with fine detail only on the surface of objects, while the interior is typically filled with large Gaussians. This static approach works well when the object is undeformed, but during large deformations, the interior of the object might become exposed, resulting in blurry rendering artifacts. To address this, we introduce a loss function that penalizes the volume ratio between the largest and smallest Gaussians:

$$L_{\text{vol\_ratio}} = \max\left(\frac{\text{mean}(V_{\text{top }\alpha})}{\text{mean}(V_{\text{bottom }\alpha})}, r\right) - r, \tag{21}$$

where r is the allowed maximum volume ratio between the largest and smallest Gaussians. In practice, we find that a value of r = 2 works well. This loss encourages the volume of all Gaussians to be as uniform as possible, improving MPM simulation quality. By reducing the disparity in Gaussian volumes, we enhance the consistency of the simulation and rendering, especially during large deformations where internal Gaussians may become visible.

Separated Appearance and Physical Representations. In PhysGaussian, each MPM particle corresponds to a Gaussian. To accurately render the original model, the density of Gaussians is usually high, exceeding the precision required by MPM (8 particles per grid), resulting in too many particles. Therefore, we maintain separate MPM

particles for simulation and Gaussians for rendering. Fewer MPM particles drive a larger number of Gaussians through the MPM grid, preserving rendering accuracy while improving simulation speed. Specifically, for each MPM particle, the complete `Sim_Substep` is executed. For the Gaussians, however, they only need to utilize the grid velocity information obtained from the MPM particle simulation to execute `Grid_to_Particle`, `Adjust_Particle`, and `Apply_Plasticity`.

Surface Rendering. Gaussian splatting is only suitable for adjusting existing reconstructed or generated 3D models. For scenarios where users want to create models from scratch, we provide the option to render directly into a mesh using the Marching Cubes algorithm. Specifically, we extract the density field information of particles from the MPM simulation and determine an isosurface with a threshold to construct the mesh surface. Then, we apply Laplacian smoothing to improve the surface quality.

However, during the modeling process, it is quite common to merge two separate parts together. The basic Marching Cubes algorithm can cause unwanted connections or "sticking" between the adjacent regions of different parts. To address this issue, we assign a "category" attribute to each particle. When performing the `Particle_to_Grid` transfer, we maintain a separate density field for each category of particles. The Marching Cubes algorithm is then executed independently on each density field, ensuring that each category is rendered separately. This approach prevents different parts of the model from sticking together during the rendering process.

5.4 Implementation

We selected the Meta Quest 3 VR headset with hand-tracking functionality as our development platform. Meta provides a comprehensive and user-friendly Unity API, which led us to choose Unity as our development framework. In addition to Unity, we integrated the rendering techniques of Gaussian Splatting and Marching Cubes, implementing them within the Unity environment.

For the simulation portion of our framework, we used Taichi as our compilation engine. Taichi's high-performance computational capabilities are well-suited for physics-based simulations. We compiled the simulation code into binary files and imported them into Unity, where they were executed via C# bindings. This allowed us to seamlessly integrate the physical simulation with the rendering process, ensuring smooth and real-time performance in the VR environment.

6 EVALUATION

In this section, we evaluate the real-time performance, rendering quality, and usability of our system, and compare it with existing modeling tools. All experiments were conducted on a 16-core 4.3GHz AMD Ryzen 9 9950X machine with an Nvidia RTX 4090 GPU.

6.1 Real-Time Performance

We first evaluate the real-time performance using a basic scenario. The test conditions are as follows: the simulation domain is a cube with an edge length of 1 unit, containing a sphere with a radius of 0.3 units in the center. The MPM simulation uses a grid resolution of 64^3, with 8 particles per grid cell at initialization. Each frame performs 5 substeps, using a Neo-Hookean elastic model and a Von Mises plasticity model. Rendering is done via the Marching Cubes algorithm with a resolution of 128^3, and Laplacian smoothing is applied for 5 iterations. The input tools are two hands, each represented by a set of MATs. We vary the grid resolution, number of substeps per frame, and the number of particles per grid cell while keeping other conditions constant, and the overall FPS is shown in Figure 5.

6.2 Convergence Under Spatial Refinement

We recorded a hand movement trajectory and applied it to models with different grid resolutions to examine if the deformation converges as the resolution increases. In this experiment, we simply moved our hand horizontally and pressed it onto a spherical model to leave a palm print. As the resolution of the model exceeds 64, the deformation stabilizes, and we can clearly observe the shape of the fingers in the resulting deformation. This is shown in Figure 6, where the deformation converges and stabilizes after spatial refinement, demonstrating that the system captures detailed interactions, especially at higher resolutions, without significant changes.

6.3 Ablation Studies

We conducted several ablation studies to assess the impact of our technical contributions.

FPS w/o localized simulation	FPS w localized simulation	scene	splats count
12.2	39.8	garden	660k
13.1	43.5	bicycle	480k
13.6	44.5	room	382k

Table 1: Comparison of results w & w/o localized simulation in large 3D GS scenes.

Local Simulation. We tested the frame rate with and without the localized simulation optimization in several typical Gaussian scenarios. As shown in Table 1, the frame rate improved by a factor of 3.3 when localized simulation was enabled.

Figure 7: Volume Regularizer. Our results (in the middle) exhibit more visual realism when interior regions are exposed due to large deformations.

Volume Regularizer. The original Gaussian splats produced large volumes, which worked well for small deformations but led to blurry artifacts during large deformations. We introduced a loss function to penalize the size ratio between the largest and smallest Gaussians, encouraging uniform volumes. As shown in Figure 7,

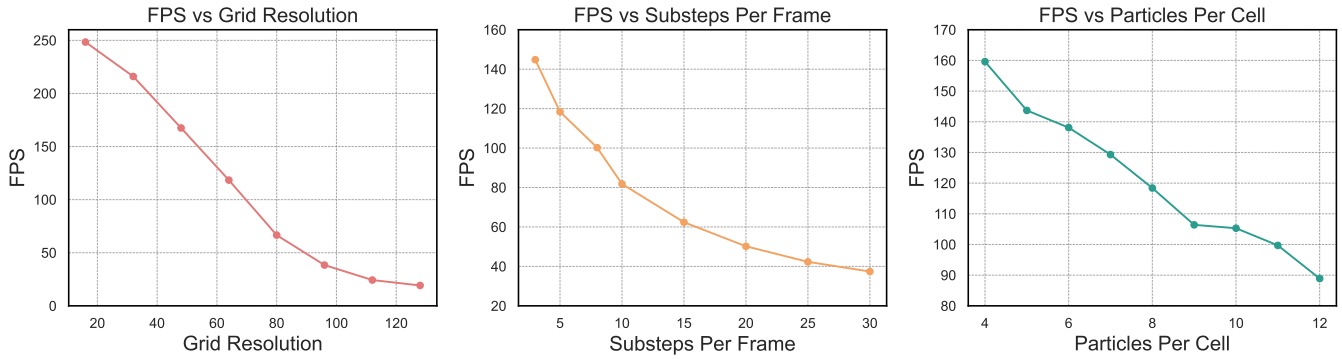

Figure 5: Real-Time Performance Evaluation. On the left is the frame rate curve under different spatial resolutions, in the middle is the frame rate curve with varying simulation steps per frame, and on the right is the frame rate curve with different numbers of particles initialized in each MPM grid.

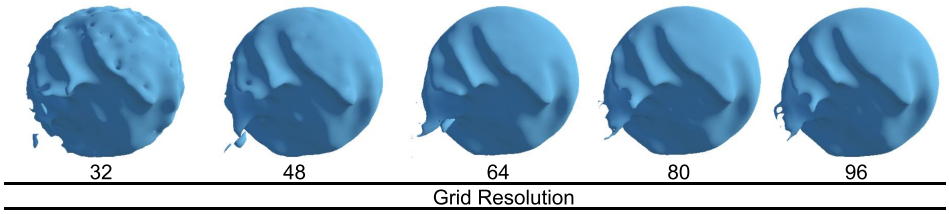

Figure 6: Convergence under Spatial Refinement. From left to right, the figure shows the deformation results of the same object under the same hand trajectory, with increasing spatial resolution.

the additional loss function effectively eliminated the blurriness caused by exposed interior regions during large deformations.

Figure 8: Separated Appearance and Physical Representations. The left side shows the models before deformation, with the sample ratio displayed on the left and the frame rate on the right beneath each model

Separated Appearance and Physical Representations. We compared the performance of our method with the original PhysGaussian, and tested the frame rate and simulation quality under different sampling rates. Here, the sampling rate refers to the ratio between the number of MPM particles and the number of Gaussians. The results are shown in Figure 8, which illustrate how varying the sampling rate affects both the frame rate and the quality of the simulation.

7 CASE STUDY

7.1 Basic Operations

Figure 9 demonstrates common operations in the modeling process, such as extrusion, stretching, twisting, merging, and using tools like plates and rods. The objects behave similarly to real-world interactions, maintaining volume conservation. This resemblance helps users intuitively model by leveraging real-life experiences.

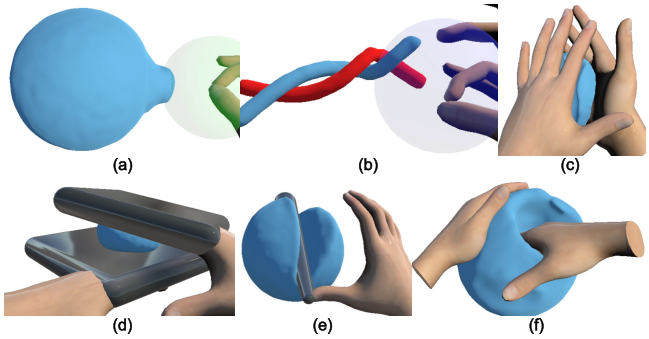

Figure 9: Basic Operations. (a) and (b) depict mid-air gestures, namely stretching and twisting; (c) and (f) illustrate direct hand manipulation; (d) and (e) represent tool manipulation.

7.2 3D GS Object Editing Examples

In this section, we showcase the process of editing 3D GS objects using our tool. In the first example, we selected a wooden frame, adjusted the angle of its top, duplicated it, and arranged the copies to freely stack together under gravity. In the second example, we "slimmed down" a toy mouse and transformed the decoration on its head into a heart shape, plaing it in the mouse's hands.

In the following two examples, we brought statues to "life": a terracotta warrior joyfully lifted his head and started drumming, while the Discobolus threw his discus with great force. In the next row, animated characters came to life as well: a pig performed ballet, and a girl began eating watermelon.

In another example, we intertwined two flowers from the decoration. Finally, in the last example, we added two more rice dumplings to a plate and modified the shape of the plate.

7.3 Modeling From Scratch Examples

In this section, we demonstrate modeling from scratch, as shown in Figure 11. In the first row, we used a snow material to start with a blank snowfield, molded a snowman, and added a hat, scarf, eyes, nose, and arms to complete it. In the second row, we shaped several flat objects with different folds and forms, then stacked them under the influence of gravity to form a hamburger. The third row shows the process of crafting an adorable frog face, where we molded a smile by hand and shaped two heart-shaped decorations to attach to the face. Finally, in the last row, we leveraged the advantages of VR's 3D space to easily arrange spheres into a neat structure and shaped a few leaves to place on top. Tasks like these, requiring precise arrangements, are extremely challenging to perform with a mouse on a 2D screen, but in our tool, they are as intuitive as arranging objects in the real world.

Figure 11: Modeling From Scratch Examples. Each row, from left to right, showcases the process of the model being built from scratch.

8 USER STUDY

To understand how experts and novices use our hands-on 3D modeling system for creative tasks in VR, we invited 6 participants (P1-P6) with 3-5 years of prior 3D modeling experience, and 6 participants without prior experience (P7-P12). The study included two open-ended tasks to evaluate system usability, performance, user satisfaction, and to gather their feedback and suggestions.

8.1 Tasks & Procedure

Each participant was required to conduct two tasks, with a 5-minute break in between: (1) *Editing a 3D Model*: Participants were free to modify a pre-existing 3D model represented by 3D GS, e.g., reshaping a plush toy or a cartoon character, to match their preferred design; (2) *Creating a 3D Model from Scratch*: Participants could freely use basic shapes such as spheres, cubes, tori, and cylinders to create a new 3D model, such as a snowman or a hamburger.

Before starting, participants practiced basic operations with a provided shape until they became familiar with the system. After completing the tasks, participants filled out the USE questionnaire [Lund 2001] to evaluate the system's usefulness, ease of use, ease of learning, and overall satisfaction. They also completed the NASA-TLX questionnaire [Hart 2006] to assess task load and the Simulator Sickness Questionnaire (SSQ) [Kennedy et al. 1993] to measure any discomfort experienced during the study. Finally, participants took part in a 10-minute semi-structured interview to provide more in-depth feedback on usability.

8.2 Results

Both expert and novice participants found the system to be highly intuitive, immersive, and efficient in their design process. We report their subjective feedback as below.

Intuition and Realism in 3D Modeling. Particularly, novices appreciated the low learning curve, allowing them to quickly engage with the tool and start editing or creating 3D models with their hands. P1 and P3 mentioned that directly manipulating the shape of objects with their hand was more flexible than deforming the object by setting control points. P1 also found the rod tool particularly useful and entertaining for poking holes and shaping details on a strawberry shape, and said it felt like *"walking inside the strawberry."* The mid-air pinch interaction was highlighted, which allows users to select the area of the object they want to deform or sculpt. P2 considered that the ability to freely rotate the viewpoint in VR is more intuitive than the traditional three-view approach.

In addition, our participants highly praised the modeling process for its realistic deformation and real-time rendering. As P4 mentioned, *"Usually, I rarely look at the texture rendering while editing shapes because it's very inefficient."* P2 described the process of editing the 3D model of the mermaid, saying, *"It feels like the mermaid is really swimming in front of me."* P3 and P8 appreciated the real-time material parameter adjustment feature, which allowed them to edit the shape based on different deformation effects under varying material parameters. P9 favored the use of gravity during the modeling process to naturally stack objects together. Besides, the interactive feedback was another major advantage, allowing participants to immediately see the deformation results of their actions and make corresponding adjustments.

Figure 10: 3D GS Objects editing Examples. On the far left of each example is the initial state of the 3D GS object, with the editing process displayed from left to right.

Encountered Problems and Suggested Features. Our participants also identified a few areas for improvement. Hand-tracking instability was mentioned most frequently (P4, P9, and P11), particularly when adjusting small or detailed parts of the object, resulting in unintended shape changes. Even though we have applied interpolation to the hand movements, this limitation is largely due to the constraints of the hand-tracking hardware. Therefore, participants expressed the need for an undo feature to help them recover from mistakes. Meanwhile, P2, P9, and P11 also suggested reducing the degrees of freedom for moving or rotating objects during the modeling process, similar to the approach used in pottery making. P1 and P7 also noted occasional performance issues in high-particle scenarios. For instance, P1 found that merging multiple objects together led to a loss of surface details. P2 and P10 mentioned that objects would unintentionally collide with the surrounding box during editing, and suggested adding a one-click centering feature to avoid this issue. P1 and P8 suggested adding an operation to separate objects.

In summary, participants generally appreciated the system's efficiency in using hands, tools, or mid-air gestural input during their 3D modeling experience. Even expert users mentioned that for more detailed modeling scenarios, traditional modeling tools are still necessary. Nevertheless, our system is extremely well-suited for quickly expressing design concepts, supporting creative exploration, and facilitating communication. While there were some concerns regarding precision and the lack of an undo feature, this did not overshadow the tool's potential for creating high-quality 3D models for both novice and expert users.

9 DISCUSSION & FUTURE WORK

Modality of Feedback. Currently, our tool only provides visual feedback, lacking the haptic feedback present in real-world interactions. As a result, users may accidentally damage areas of the model outside their line of sight (e.g., occluded regions). In reality, this issue does not arise because the user's hand would encounter tactile resistance, prompting them to stop before causing further unintended modifications. However, our existing VR hardware does not support haptic feedback. Dedicated haptic feedback gloves could address this limitation, as they not only provide tactile sensations but also enable more accurate hand tracking. Nevertheless, this approach would significantly raise the accessibility barrier for our tool. An alternative solution is to incorporate auditory feedback, where sounds are played in response to the force exerted by the user's hand on the model, enabling users to indirectly sense the effects of their actions on unseen areas.

Undo Functionality. In modeling software, especially those focused on content creation, undo is a crucial function. In traditional modeling software, actions are discrete, allowing the undo function to simply record each action, then reverse it to restore the model to a previous state. However, our tool is based on physical simulation, where user interactions are nearly continuous. Recording every user input and attempting to reconstruct the model's previous state would effectively require rerunning the entire simulation, which is

time-consuming and would detract from the user experience. Our current approach involves periodically saving the model state at fixed intervals in the background and writing it directly to disk, while also allowing users to save manually at any point. This way, undo functionality equates to restoring the last saved state, though this method is more storage-intensive. Finding an efficient way to record changes between two simulated states remains an open research question worthy of exploration, as it would contribute to a smoother modeling experience while minimizing storage requirements.

10 CONCLUSION

In this work, we introduced VR-Doh, a VR-based 3D modeling system that combines advanced elastoplastic simulation with intuitive hand-based interaction. By leveraging MPM and introducing technical optimizations such as localized simulation, optimized collision handling, and separated appearance and physical representations, the system achieves real-time responsiveness while maintaining realistic simulation and rendering quality. These innovations enhance accessibility for novice users while supporting expert users in detailed and expressive modeling tasks. Through a user study, we demonstrated the system's ability to deliver an intuitive and immersive modeling experience. Compared to traditional modeling tools, the system offers a more natural and accessible approach in specific scenarios. Future work will focus on further optimizing performance and expanding the toolset to support more complex modeling workflows, laying the groundwork for physics-driven content creation in VR environments.

REFERENCES

Rahul Arora, Rubaiat Habib Kazi, Danny M Kaufman, Wilmot Li, and Karan Singh. 2019. Magicalhands: Mid-air hand gestures for animating in vr. In *Proceedings of the 32nd annual ACM symposium on user interface software and technology*. 463–477.

Héctor Barreiro, Joan Torres, and Miguel A Otaduy. 2021. Natural tactile interaction with virtual clay. In *2021 IEEE World Haptics Conference (WHC)*. IEEE, 403–408.

Shounak Chatterjee. 2024. Free-form Shape Modeling in XR: A Systematic Review. *arXiv preprint arXiv:2401.00924* (2024).

Hanchen Deng, Jin Li, Yang Gao, Xiaohui Liang, Hongyu Wu, and Aimin Hao. 2023. PhyVR: Physics-based Multi-material and Free-hand Interaction in VR. In *2023 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE, 454–462.

Guillaume Dewaele and Marie-Paule Cani. 2004. Interactive global and local deformations for virtual clay. *Graphical Models* 66, 6 (2004), 352–369.

Sevinc Eroglu, Sascha Gebhardt, Patric Schmitz, Dominik Rausch, and Torsten Wolfgang Kuhlen. 2018. Fluid sketching—Immersive sketching based on fluid flow. In *2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. IEEE, 475–482.

Yu Fang, Minchen Li, Chenfanfu Jiang, and Danny M. Kaufman. 2021. Guaranteed Globally Injective 3D Deformation Processing. *ACM Trans. Graph. (SIGGRAPH)* 40, 4, Article 75 (2021).

Noura Faraj, Jean-Marc Thiery, and Tamy Boubekeur. 2013. Progressive medial axis filtration. In *SIGGRAPH Asia 2013 Technical Briefs*. 1–4.

Xinyu Gao, Ziyi Yang, Bingchen Gong, Xiaoguang Han, Sipeng Yang, and Xiaogang Jin. 2024. Towards Realistic Example-based Modeling via 3D Gaussian Stitching. *arXiv preprint arXiv:2408.15708* (2024).

GravitySketch. 2024. Gravity Sketch VR. https://help.gravitysketch.com/hc/en-us/articles/5902302194077-What-is-SubD-Subdivision-Modelling

Sandra G Hart. 2006. NASA-task load index (NASA-TLX); 20 years later. In *Proceedings of the human factors and ergonomics society annual meeting*, Vol. 50. Sage publications Sage CA: Los Angeles, CA, 904–908.

Markus Höll, Markus Oberweger, Clemens Arth, and Vincent Lepetit. 2018. Efficient physics-based implementation for realistic hand-object interaction in virtual reality. In *2018 IEEE conference on virtual reality and 3D user interfaces (VR)*. IEEE, 175–182.

Yuanming Hu, Yu Fang, Ziheng Ge, Ziyin Qu, Yixin Zhu, Andre Pradhana, and Chenfanfu Jiang. 2018. A moving least squares material point method with displacement discontinuity and two-way rigid body coupling. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–14.

Jan Jacobs and Bernd Froehlich. 2011. A soft hand model for physically-based manipulation of virtual objects. In *2011 IEEE virtual reality conference*. IEEE, 11–18.

Chenfanfu Jiang, Craig Schroeder, Joseph Teran, Alexey Stomakhin, and Andrew Selle. 2016. The material point method for simulating continuum materials. In *Acm siggraph 2016 courses*. 1–52.

Ying Jiang, Chang Yu, Tianyi Xie, Xuan Li, Yutao Feng, Huamin Wang, Minchen Li, Henry Lau, Feng Gao, Yin Yang, et al. 2024. VR-GS: a physical dynamics-aware interactive gaussian splatting system in virtual reality. In *ACM SIGGRAPH 2024 Conference Papers*. 1–1.

Robert S Kennedy, Norman E Lane, Kevin S Berbaum, and Michael G Lilienthal. 1993. Simulator sickness questionnaire: An enhanced method for quantifying simulator sickness. *The international journal of aviation psychology* 3, 3 (1993), 203–220.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Trans. Graph.* 42, 4 (2023), 139–1.

Kodon. 2024. KODON. https://www.kodon.xyz/#about

Vikash Kumar and Emanuel Todorov. 2015. Mujoco haptix: A virtual reality system for hand manipulation. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*. IEEE, 657–663.

Lei Lan, Yin Yang, Danny Kaufman, Junfeng Yao, Minchen Li, and Chenfanfu Jiang. 2021. Medial IPC: accelerated incremental potential contact with medial elastics. *ACM Transactions on Graphics* 40, 4 (2021).

Pan Li, Bin Wang, Feng Sun, Xiaohu Guo, Caiming Zhang, and Wenping Wang. 2015. Q-mat: Computing medial axis transform by quadratic error minimization. *ACM Transactions on Graphics (TOG)* 35, 1 (2015), 1–16.

William E Lorensen and Harvey E Cline. 1998. Marching cubes: A high resolution 3D surface construction algorithm. In *Seminal graphics: pioneering efforts that shaped the field*. 347–353.

Christos Lougiakis, Jorge Juan González, Giorgos Ganias, Akrivi Katifori, Maria Roussou, et al. 2024. Comparing Physics-based Hand Interaction in Virtual Reality: Custom Soft Body Simulation vs. Off-the-Shelf Integrated Solution. In *2024 IEEE Conference Virtual Reality and 3D User Interfaces (VR)*. IEEE, 743–753.

Arnold M Lund. 2001. Measuring usability with the use questionnaire12. *Usability interface* 8, 2 (2001), 3–6.

Michael R Marner and Bruce H Thomas. 2010. Augmented foam sculpting for capturing 3D models. In *2010 IEEE Symposium on 3D User Interfaces (3DUI)*. IEEE, 63–70.

Kevin T McDonnell, Hong Qin, and Robert A Wlodarczyk. 2001. Virtual clay: A real-time sculpting system with haptic toolkits. In *Proceedings of the 2001 symposium on Interactive 3D graphics*. 179–190.

Joss Kingdom Moo-Young, Andrew Hogue, and Veronika Szkudlarek. 2021. Virtual materiality: Realistic clay sculpting in vr. In *Extended Abstracts of the 2021 Annual Symposium on Computer-Human Interaction in Play*. 105–110.

Andrew Nealen, Takeo Igarashi, Olga Sorkine, and Marc Alexa. 2007. Fibermesh: designing freeform surfaces with 3d curves. In *ACM SIGGRAPH 2007 papers*. 41–es.

Adeline Pihuit, Paul G Kry, and Marie-Paule Cani. 2008. Hands on virtual clay. In *2008 IEEE International Conference on Shape Modeling and Applications*. IEEE, 267–268.

Henning Pohl and Aske Mottelson. 2022. Hafnia Hands: A Multi-Skin Hand Texture Resource for Virtual Reality Research. *Frontiers in Virtual Reality* 3 (2022).

Enrique Rosales, Chrystiano Araújo, Jafet Rodriguez, Nicholas Vining, Dongwook Yoon, and Alla Sheffer. 2021. AdaptiBrush: adaptive general and predictable VR ribbon brush. *ACM Trans. Graph.* 40, 6 (2021), 247–1.

Enrique Rosales, Jafet Rodriguez, and Alla Sheffer. 2019. SurfaceBrush: from virtual reality drawings to manifold surfaces. *arXiv preprint arXiv:1904.12297* (2019).

Steven Schkolne, Michael Pruett, and Peter Schröder. 2001. Surface drawing: creating organic 3D shapes with the hand and tangible tools. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 261–268.

HJ Schulz, M Teschner, and M Wimmer. 2019. Rodmesh: Two-handed 3D surface modeling in virtual reality. In *Vision, modeling and visualization*. 1–10.

ShapeLab. 2024. Shapelab. https://shapelabvr.com/

Jia Sheng, Ravin Balakrishnan, and Karan Singh. 2006. An interface for virtual 3D sculpting via physical proxy.. In *GRAPHITE*, Vol. 6. 213–220.

Breannan Smith, Chenglei Wu, He Wen, Patrick Peluse, Yaser Sheikh, Jessica K Hodgins, and Takaaki Shiratori. 2020. Constraining dense hand surface tracking with elasticity. *ACM Transactions on Graphics (ToG)* 39, 6 (2020), 1–14.

Svetlana Stolpner, Paul Kry, and Kaleem Siddiqi. 2011. Medial spheres for shape approximation. *IEEE transactions on pattern analysis and machine intelligence* 34, 6 (2011), 1234–1240.

Substance3d. 2024. substance3d. https://www.adobe.com/products/substance3d/apps/modeler.html

Deborah Sulsky, Shi-Jian Zhou, and Howard L Schreyer. 1995. Application of a particle-in-cell method to solid mechanics. *Computer physics communications* 87, 1-2 (1995), 236–252.

Feng Sun, Yi-King Choi, Yizhou Yu, and Wenping Wang. 2015. Medial meshes–a compact and accurate representation of medial axis transform. *IEEE transactions on visualization and computer graphics* 22, 3 (2015), 1278–1290.

Mickeal Verschoor, Daniel Lobo, and Miguel A Otaduy. 2018. Soft hand simulation for smooth and robust natural interaction. In *2018 IEEE conference on virtual reality and 3D user interfaces (VR)*. IEEE, 183–190.

Tianyi Xie, Zeshun Zong, Yuxing Qiu, Xuan Li, Yutao Feng, Yin Yang, and Chenfanfu Jiang. 2024. Physgaussian: Physics-integrated 3d gaussians for generative

dynamics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 4389–4398.

Emilie Yu, Rahul Arora, Tibor Stanko, J Andreas Bærentzen, Karan Singh, and Adrien Bousseau. 2021. Cassie: Curve and surface sketching in immersive environments. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–14.