

Article

Crack-Free GPU Tessellation for Real-Time Visual Bone Drilling in Virtual Arthroscopic Simulators

Simon D. Koch ^{1,*}, Furkan Dinc ², Tansel Halic ³, Hayden Reitz ⁴ and Clayton Maddox ⁴

¹ Department of BioMedics, Medical AI Institute, College of Osteopathic Medicine, Sam Houston State University, Conroe, TX 77340, USA

² Planmeca USA Inc., Plano, TX 75082, USA

³ Intuitive Surgical, Atlanta, GA 30092, USA; tansel.halic@intusurg.com

⁴ College of Osteopathic Medicine, Sam Houston State University, Conroe, TX 77340, USA

* Correspondence: sxx144@shsu.edu

Abstract

Background/Objectives: Real-time arthroscopic bone drilling in simulators requires high graphics and haptic update rates while deforming bone visually. We present a fully GPU-resident, shader-based pipeline for the Virtual Rotator Cuff Arthroscopic Skill Trainer (ViRCAST) that avoids mesh regeneration. **Methods:** Instead of changing geometry, the method updates a vector displacement map and a tessellation amount map. The pipeline uses six GPU passes: drawing-vector calculation, displacement-map generation, tessellation-map generation via Sobel edge detection, mipmapped tessellation sampling, vertex displacement, and fragment-level normal correction. A patterned-drawing scheme reduces stretched-triangle artifacts, and an edge-symmetric UV rule assigns identical tessellation factors to shared edges. **Results:** On a Dell desktop with an Intel Core Ultra 9, NVIDIA GeForce RTX 5090, 32 GB VRAM, 128 GB RAM, and Windows 11, 1000-frame GPU timing showed that 4K texture updates completed in 0.292 ms on average, and the full drilling update completed in 0.397 ms. This is far below the 16.7 ms budget for 60 Hz rendering. The pipeline sustained 120 fps in multi-object scenes and improved frame rate by up to 24% over uniform tessellation, with SSIM ≈ 0.99992 on depth output. **Conclusions:** The method provides surface deformation, complements volumetric drilling modules, and supports continuous feedback for surgical training.

Keywords: adaptive tessellation; 3D mesh generation; real-time mesh alteration; displacement maps; GPU rendering; virtual reality; 3D graphics



Academic Editor: Pedro Alonso Jordá

Received: 10 July 2026

Revised: 4 September 2026

Accepted: 7 September 2026

Published: 20 September 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

Arthroscopic Rotator Cuff (ARC) repair is a minimally invasive treatment for tears in the shoulder's rotator cuff tendons [1–3]. The procedure is performed through a small arthroscopic portal and requires the surgeon to drill the humeral head at $\sim 45^\circ$ to seat suture anchors that will, in turn, hold the repaired tendon in place [4]. Anchor placement accuracy is a critical component of repair integrity. Rotator cuff repairs already have a reported repair rate ranging from 10 to 20% in younger patients with small tears to 40–50% or higher in patients over 70 years old with large tears [5]. However, suboptimal drill trajectories can further increase this risk by reducing cortical bone purchase and predisposing the anchor to pullout. Accurate drilling angle and depth are therefore skills that must be consistently mastered before operating independently. ARC caseloads have risen by 392% for men and 808% for women in the United States over recent years [1], yet training modalities

remain dominated by cadavers, mannequins, and the apprenticeship model—each limited in cost, realism, or patient-safety profile [6–10]. The problem is compounded by ACGME duty-hour restrictions, which compress the window in which trainees can accrue cases [10].

Virtual-reality (VR) simulators address these limitations by providing repeatable, risk-free practice with objective performance metrics [11–18]. Randomized controlled studies have demonstrated that skills acquired on VR arthroscopy simulators translate to improved surgical performance, including fewer technical errors and reduced task completion times [19,20]. Gomoll et al. [21] further established construct validity for a virtual shoulder arthroscopy simulator by showing that more experienced surgeons outperformed their less experienced counterparts, with significant correlations between experience level and faster task completion, shorter instrument paths, and fewer errors. We have been developing the Virtual Rotator Cuff Arthroscopic Skill Trainer (ViRCAST) as a VR-based alternative for ARC training [4,22–24]. ViRCAST combines photorealistic rendering with haptic force feedback, giving trainees a realistic immersive environment in which to rehearse anatomical navigation, bursectomy, bone drilling, anchor placement, and suture-to-bone fixation.

Bone drilling is a particularly demanding module to simulate. Unlike rigid-object interactions, the bone mesh itself must deform as the drill advances, and the deformation must be visible within one graphics frame (≤ 16.7 ms) and registerable in the haptic loop at ≥ 700 Hz [25]. Clinically, continuous feedback is essential during drilling to detect cortical breakthrough, recognize transitions from hard cortex to softer cancellous bone, and stop at the intended depth to avoid neurovascular injury. Interruptions causing frame drops or haptic loop stalls disrupt the simulator feedback loop, lowering perceptual fidelity and risking reinforcement of incorrect motor patterns. Therefore, sustaining stable frame rates and visual updates is essential for valid skill transfer. Crack-free surface extraction from Adaptive Mesh Refinement (AMR) data is a known challenge in visualization, and prior work has developed methods for extracting crack-free isosurfaces across refinement boundaries [26]. In surgical simulation, related volumetric and multi-resolution approaches have been explored for VR-based arthroscopic anchor-placement drilling and physics-based bone drilling and shaving [26,27]. However, such volumetric approaches can still require repeated updates to the volume and extracted surface representation during tool interaction, with computational costs that increase as resolution rises. This motivates the shader-based method proposed here, which avoids mesh regeneration by updating GPU-resident displacement and tessellation textures.

In this paper we present a complementary, shader-based approach that sidesteps mesh regeneration entirely. The bone mesh is never modified; instead, its appearance is altered through two GPU-resident textures that are updated every frame. The first, a vector displacement map, encodes where and how the surface has been drilled; the second, a tessellation amount map, encodes how finely each triangle should be subdivided to represent the local surface detail without over-tessellating flat regions. Both textures are authored, updated, sampled, and consumed entirely on the GPU—no vertex or index data ever crosses the PCIe bus during drilling.

Contributions

The principal contributions of this work are as follows:

- (1) A six-pass, fully GPU-resident pipeline for real-time bone drilling that modifies a vector displacement map and a tessellation amount map instead of the underlying mesh, avoiding CPU–GPU synchronization bottlenecks during deformation.

(2) A patterned-drawing blend operator that prevents the stretched-triangle artifact at drill-site edges by shaping the displacement to a cylindrical brush pattern rather than a radial falloff.

(3) A Sobel-filter-based curvature detector that concentrates tessellation at displacement edges; we compare five curvature estimators and show the Sobel variant achieves Structural Similarity Index Measure (SSIM) ≈ 0.99992 against uniform tessellation on depth output.

(4) An edge-symmetric UV sampling rule combined with mipmap-based tessellation-amount selection that assigns identical tessellation factors to both sides of every shared edge, verified exhaustively on a production anatomical mesh, while preserving LOD benefits at distance.

(5) Comprehensive GPU-timed performance characterization at all six pipeline stages, reporting mean, standard deviation, and p50/p95/p99 percentile latencies over 1000 measurement frames per resolution. On the target hardware the end-to-end drilling update is 0.397 ms mean at 4K texture resolution (p95 = 0.424 ms), with up to 24% higher frame rates than uniform tessellation in representative multi-object scenes. (6) An evaluation on production ViRCAST anatomical meshes that measures geometric error against the continuous displacement field rather than against a denser rendering, together with an exhaustive per-edge verification of the crack-free rule and an audit of the mesh-topology classes the rule does and does not cover.

2. Related Work

2.1. Volumetric Versus Shader-Based Bone Deformation

CPU-centric approaches to interactive bone deformation typically remove voxels from a uniform grid and re-extract the surface with Marching Cubes [28]. These methods scale poorly with grid resolution and suffer from well-known cracking artifacts. Octree acceleration [29] mitigates the cost but not the cracks. Block-structured Adaptive Mesh Refinement (AMR) [30,31] combined with dual-mesh extraction [32] can produce crack-free meshes at interactive rates, and we have recently demonstrated an AMR-based drilling module in ViRCAST [22]. Shader-based methods take the opposite view: rather than modifying geometry, they modify its shading-time appearance through textures. The pipeline presented here belongs to the same family. Schäfer et al. [33,34] established this family for general-purpose local sculpting and painting, storing per-face displacement in dynamically managed GPU memory and applying it through the hardware tessellation unit. Our contribution is therefore not texture-driven deformation as such, but its adaptation to interactive surgical bone drilling: the brush is shaped by instrument geometry rather than by an authoring stroke, the tessellation rule is edge-symmetric on a fixed anatomical asset with a production UV atlas, and the per-frame budget is set by a haptics-coupled simulator rather than by an interactive modeling tool.

2.2. GPU Tessellation

Hardware tessellation, introduced with Shader Model 5.0 [35], inserts three new stages into the graphics pipeline: the Hull Shader (programmable, computes tessellation factors), the Tessellator (fixed-function, emits tessellated UVs), and the Domain Shader (programmable, evaluates new vertex positions). Catmull–Clark subdivision [36] set the theoretical foundation; Moreton [37] and Boubekeur and Schlick [38,39] adapted it to triangle and arbitrary-topology meshes on the GPU. Phong tessellation [40] further smooths low-resolution meshes by displacing the generated vertices along averaged vertex normals. More recent work [41–43] addresses GPU-based adaptive tessellation, view-dependent tessellation for Catmull–Clark surfaces [44,45], and feature-adaptive real-time

rendering [46]. None of these methods target the particular case of surgical drilling, in which the displacement is non-uniform, small-scale, and conducted interactively by a haptic tool.

2.3. Displacement Mapping and Feature Preservation

Early displacement mapping encoded surface offsets as scalar grayscale textures [47]; full-vector displacement, which permits motion in any tangent-space direction, is encoded into three-channel textures with a sign-shift pack/unpack operator. Doggett and Hirche [48] introduced view-dependent adaptive tessellation of displacement maps and evaluated subdivision tests on triangle edges only, which is the earliest statement of the principle that Equation (9) apply here. Fisher et al. [49] made that principle exact for micropolygon rendering: in DiagSplit parallel algorithm each edge's tessellation factor is computed from information local to that edge alone, so independently processed patches cannot introduce T-junctions or cracks. Our edge-symmetric mipmap selection is the hardware-tessellation, dynamically authored-texture analog of the same argument. Nießner and Loop [50] showed how displacement mapping itself can be made analytic and seamless under hardware tessellation. Jang and Han [51,52] proposed a feature-preserving displacement method based on precomputed indirect scalar displacement maps (ISDM); while effective for static features, ISDM is not designed for on-the-fly mesh modifications such as drilling. Moule and McCool [53] introduced efficient bounded adaptive tessellation of displacement maps, which inspires multi-resolution tessellation-amount texture. Our work extends [53] with an adaptive curvature estimator that is computed entirely on the GPU on each frame from the live displacement texture.

2.4. Terrain and LOD Tessellation

Terrain renderers [54,55] combine chunk-based LOD with hardware tessellation for detail close to the viewer; Mikhaylyuk et al. [56] extend this to planetary-scale rendering. Lambert et al. [57] use tessellation for seamless LOD transitions in multi-resolution meshes, and Tiwary et al. [58] exploit foveated rendering via adaptive tessellation. These techniques inform our mipmap-based tessellation-amount selection but do not address arbitrary, interactively authored displacement maps.

3. Materials and Methods

3.1. GPU Tessellation Pipeline

Figure 1 shows the Shader Model 5.0 graphics pipeline. In our work, we program the Hull and Domain Shaders and configure the fixed-function Tessellator via four per-patch tessellation factors.

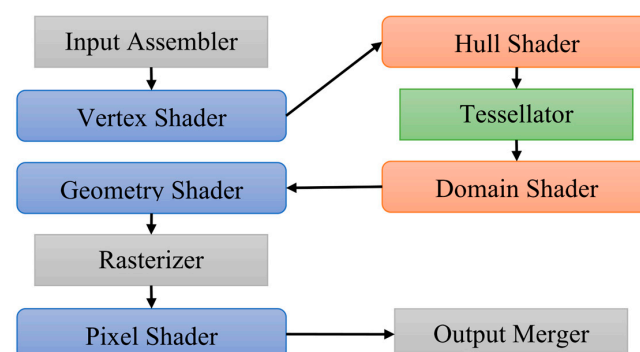


Figure 1. The Shader Model 5.0 graphics pipeline used by our method. The programmable Hull, Tessellator, and Domain Shader stages (together with the fragment-level Pixel Shader) form the substrate for six-pass drilling pipeline.

The Tessellator accepts three edge factors (one per triangle edge) and one inner factor controlling the interior grid. Edge factors determine how many segments each edge is split into; the inner factor controls the density of the grid of sub-triangles laid out inside the patch. Once the interior grid is created, the subdivided edges are stitched to it. Figure 2 shows a sample tessellation of one triangle under the factors (5, 29, 3, 9).

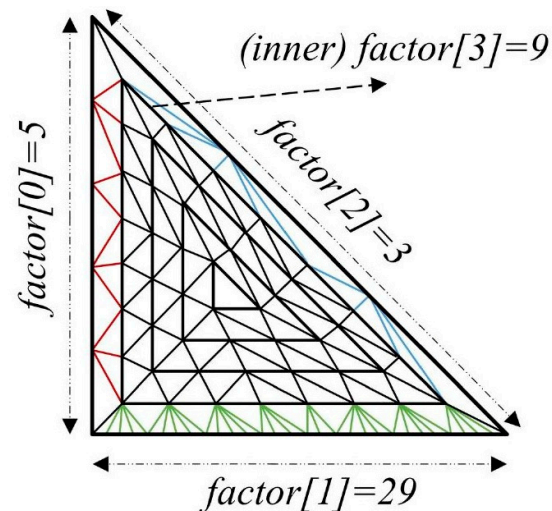


Figure 2. Sample tessellation of a single triangle with edge factors (5, 29, 3) and inner factor 9. Red, green, and blue segments show the three edge subdivisions; black lines show the interior grid and the stitching triangles that connect the two.

3.2. Phong Tessellation

Phong tessellation [40] smooths low-resolution meshes by nudging each generated vertex along a weighted average of the neighboring vertex normals. Figure 3 illustrates the effect: a coarse polygonal fan (a) gains interior vertices from tessellation (b), which Phong smoothing (c) then displaces slightly to approximate the underlying smooth surface. Our pipeline uses displacement mapping rather than Phong smoothing at the drilling site, because Phong assumes a smooth underlying surface and cannot represent a sharp cavity.

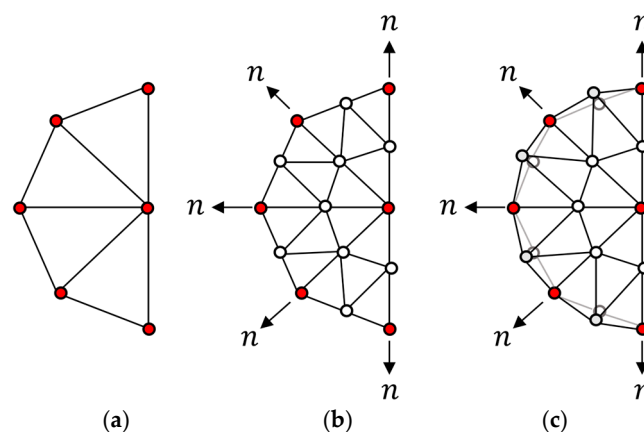


Figure 3. Phong tessellation. (a) Original low-resolution fan. (b) After tessellation, additional (white) vertices lie on the original planar faces. (c) Phong smoothing displaces the tessellated vertices along averaged normals to approximate a smooth surface.

3.3. Notation Used

Let a triangle of the base mesh have vertices $i \in \{0, 1, 2\}$ with position $x_i \in \mathbb{R}^3$, UV coordinate $u_i = (u_i, v_i) \in [0, 1]^2$, unit normal n_i , unit tangent t_i , and unit bitangent b_i . For a

tessellated point with UV coordinate u , the interpolated position and frame are denoted by $x(u)$, $t(u)$, $b(u)$, and $n(u)$. Then the local Tangent–Bitangent–Normal matrix is

$$M_{TBN}(u) = [t(u) \mid b(u) \mid n(u)] \quad (1)$$

The three columns of $M_{TBN}(u)$ are expressed in the same coordinate system as $x(u)$. If the mesh is transformed by a model matrix, $t(u)$ and $b(u)$ are transformed as directions and re-normalized. Under non-uniform scaling, $n(u)$ is transformed with the inverse-transpose normal matrix and then re-normalized.

Let $D(u) \in [0, 1]^3$ be the packed RGB vector-displacement texture sample, $h(u) \in [0, 1]$ be a scalar height/displacement sample, and $E(u) \in [0, 1]$ be the tessellation-importance map computed from displacement gradients. All texture samples used in these equations are assumed to be in linear space, not sRGB space.

3.4. Displacement Maps

A displacement map encodes per-UV surface offsets. Scalar (grayscale) maps displace each vertex along its normal by the relation in Equation (2):

$$x'_i = x_i + s_h(h(u_i) - h_0)n_i \quad (2)$$

where s_h is the maximum scalar displacement magnitude in object or world units, $h(u_i)$ is the scalar map value sampled at the vertex UV coordinate, h_0 is the neutral value, and n_i is the unit surface normal. We use $h_0 = 0$ if the stored scalar represents zero-to-positive displacement, and $h_0 = 0.5$ if the texture stores signed displacement around neutral gray. For bone removal, the sign of s_h is chosen so that the displacement points inward relative to the surface normal.

For vector displacement, the texture does not directly store a world-space vector. It stores a packed tangent-space vector in three channels. Let $\delta_t(u) \in \mathbb{R}^3$ be the signed tangent-space displacement vector, and let $s = (s_t, s_b, s_n)^T$ contain the maximum absolute displacement scales along the tangent, bitangent, and normal directions. Then the scale matrix is

$$S_d = \text{diag}(s_t, s_b, s_n) \quad (3)$$

Because texture channels are non-negative, signed tangent-space displacement vectors are stored and recovered using the component wise pack and unpack operators:

$$\text{Pack}(\delta_t) = \text{clamp}(0.5(S_d^{-1}\delta_t + 1), 0, 1) \quad (4)$$

$$\text{Unpack}(c) = S_d(2c - 1), \quad c \in [0, 1]^3 \quad (5)$$

If the displacement values are already dimensionless and limited to $[-1, 1]^3$, then $S_d = I$, $\text{Pack}(v) = 0.5(v + 1)$, and $\text{Unpack}(c) = 2c - 1$. The clamp, multiplication, addition, Pack, and Unpack operations are applied component wise. In implementation, the render texture is sampled in linear space, and a floating-point format is used when available to reduce quantization.

After unpacking, the tangent-space vector is transformed to object or world space by the local tangent frame:

$$\delta_w(u) = M_{TBN}(u)\text{Unpack}(D(u)) \quad (6)$$

Here, $D(u)$ is the packed RGB displacement-map value at UV coordinate u , $\text{Unpack}(D(u))$ is the signed tangent-space displacement vector, and $\delta_w(u)$ is the corresponding object- or world-space displacement vector.

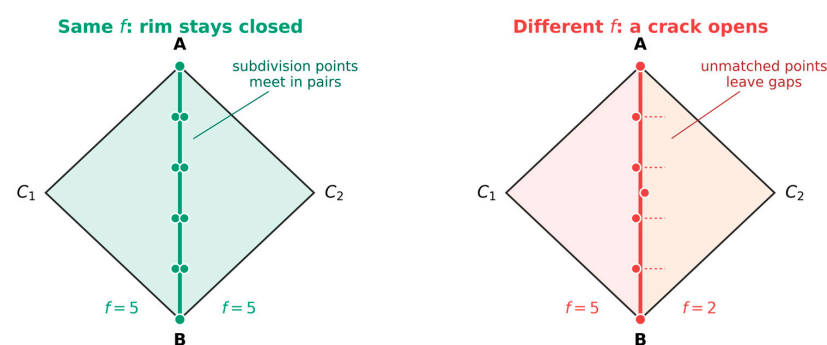
Because $M_{\text{TBN}}(\mathbf{u})$ and $\mathbf{x}(\mathbf{u})$ are expressed in the same coordinate system, $\delta_w(\mathbf{u})$ can be added directly to the surface point. The displaced vertex or tessellated-domain point is then:

$$\mathbf{x}'(\mathbf{u}) = \mathbf{x}(\mathbf{u}) + \delta_w(\mathbf{u}) \quad (7)$$

We use vector displacement throughout, because drilling generates motion both into and along the surface.

3.5. The Crack Problem Under Tessellation

When two adjacent patches receive different edge factors, the Tessellator emits different numbers of sub-vertices along their shared edge. Any non-zero displacement then opens a visible gap at the edge (Figure 4). Our pipeline suppresses this by enforcing identical tessellation amounts on shared edges through an edge-symmetric UV-based sampling rule (Section 3.10).



The edge tessellation factor f is the number of segments the tessellator cuts an edge into (range 1 to 64).

Figure 4. The edge tessellation factor and the origin of tessellation cracks. Two triangles, ABC_1 and ABC_2 , share the edge AB , and each independently computes a factor f for that edge. Left: both compute $f = 5$, so the four interior subdivision points coincide pairwise, and the tessellated surface remains closed across the edge. Right: the left triangle computes $f = 5$ while the right computes $f = 2$, so the subdivision points do not correspond, and gaps open along the shared edge. Preventing this requires that the factor for AB depends only on A and B , which is the property established by Equation (9) and verified exhaustively in Section 4.7.

3.6. Adaptive Tessellation Pipeline Overview

Our pipeline comprises six GPU passes executed per frame whenever the drill is in contact with the bone surface (Figure 5). All passes read from and write to GPU-resident textures; the CPU only supplies the current drill pose and forwards the frame draw call.

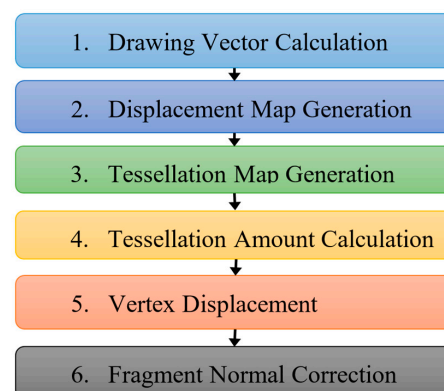


Figure 5. Overall six-pass pipeline: (1) Drawing Vector Calculation, (2) Displacement Map Generation, (3) Tessellation Map Generation, (4) Tessellation Amount Sampling, (5) Vertex Displacement, and (6) Fragment Normal Correction.

3.7. Pass 1: Drawing Vector Calculation

A ray is cast from the drill tip along the drill axis to the bone surface. If the ray hits, we read the hit UV coordinate, normal, tangent, and bitangent. The drawing vector d_v denotes the drilling direction at the hit point, and the drawing-up vector d_{vup} provides a second direction for orienting the brush pattern around d_v . These two vectors define the drilling frame used in the next pass. If the ray misses, the displacement map is left unchanged for this frame.

3.8. Pass 2: Displacement Map Generation

We store the displacement map as a persistent off-screen render texture. Each frame that records a drill contact, Pass 2 updates that map with a copy-and-update pair rather than binding the same resource simultaneously as a sampled texture and as a color render target. First, the current contents of the displacement map are copied into a full-resolution temporary render texture (no mip chain) obtained from the Unity temporary-render-texture pool. Second, a full-screen shader samples that temporary copy, adds the contribution of the current drill contact (Equation (8)), and writes the result back to the persistent displacement map; the temporary texture is then released to the pool. This is not a classical two-buffer ping-pong in which two persistent maps swap roles each frame without a copy: the extra full-resolution blit is the cost of keeping the displacement map at a stable resource address for Pass 3, the mipmap chain, and the domain-shader samples that follow. The persistent map is never read and written in the same draw, which avoids an undefined read–write feedback hazard on the Direct3D 11/Unity 2019.3.5f1 HDRP configuration used here. This iterative additive scheme yields a temporally smooth, cumulative displacement without any CPU bookkeeping of surface geometry.

A naive circular brush at the contact point produces a stretched-triangle artifact at the drilling boundary, because adjacent vertices that share a triangle can receive very different displacement magnitudes; one at the edge of the brush, one outside it (Figure 6). The resulting inhomogeneous triangles have a single per-triangle normal and thus shade with visible discontinuities along the cavity rim.

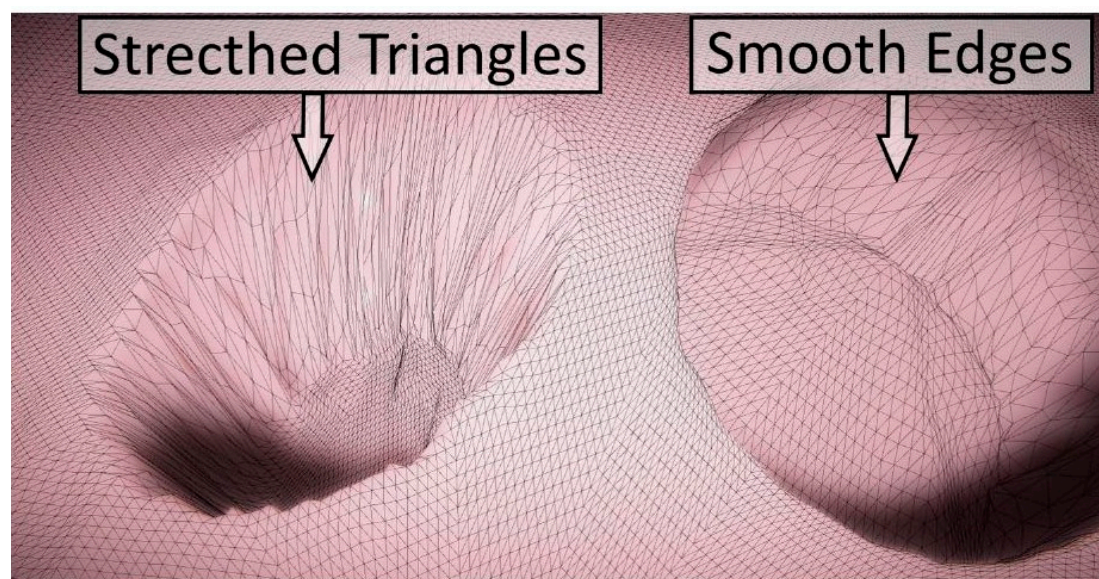


Figure 6. Stretched-triangle artifact (left of the image) at the drilling edge caused by a blunt circular brush: adjacent vertices of the same triangle experience very different displacements, producing a highly elongated triangle with a single per-triangle normal and visible shading artifacts. The right of the image shows the smooth result after we apply patterned drawing.

We fix this with a patterned-drawing scheme. Instead of a uniform radial brush, we use a cylindrical brush texture (Figure 7b) whose three color channels pack a tangent-space displacement vector. At each drilling step, the brush pixel value $p \in [0, 1]^3$ is unpacked, rotated into the current drilling frame, and blended with the previous displacement-map texel. We perform the blending in unpacked signed-vector space and then pack the result back into RGB texture space. Let c_{prev} and c_{new} be the previous and updated packed texel values, respectively; let $\alpha \in (0, 1]$ be the blend factor. The update assumes d_{vup} is not parallel to d_v , so that a nonzero bitangent direction can be formed. Then the update is:

$$\begin{aligned} d_v &\leftarrow \text{normalize}(d_v) \\ d_{vup} &\leftarrow \text{normalize}(d_{vup} - (d_{vup} \cdot d_v)d_v) \\ d_{vbi} &= \text{normalize}(d_{vup} \times d_v) \\ d_p &= [d_v \mid d_{vbi} \mid d_{vup}]\text{Unpack}(p) \\ c_{new} &= \text{Pack}\left((1 - \alpha)\text{Unpack}(c_{prev}) + \alpha d_p\right) \end{aligned} \quad (8)$$

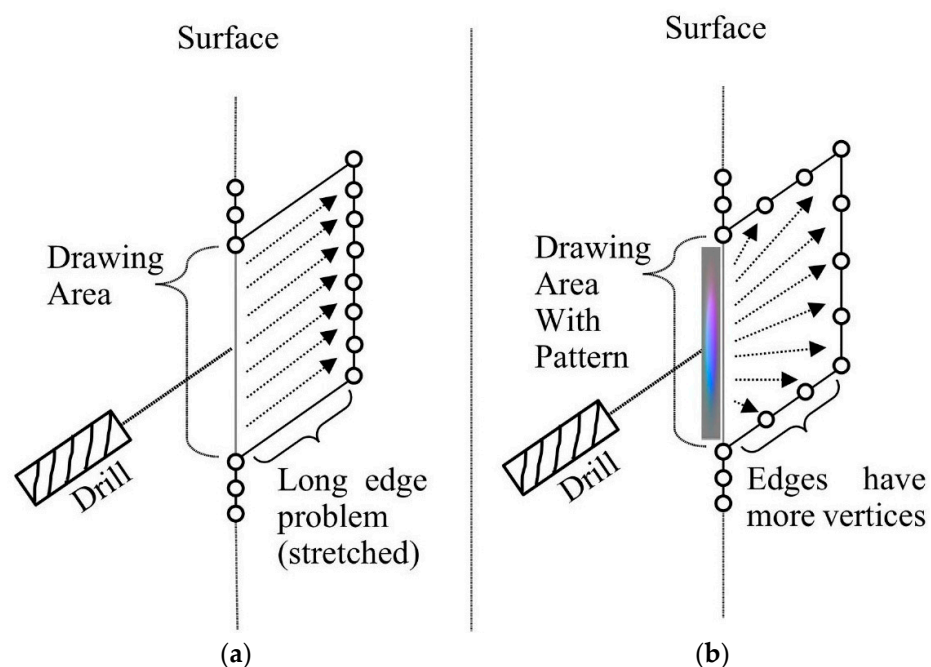


Figure 7. Patterned-drawing scheme. (a) Without a pattern, a uniform radial brush produces one elongated (“stretched”) triangle at the drilling edge; (b) with a patterned brush, the density of displaced vertices is concentrated along the cavity rim, producing clean triangles with consistent per-triangle normals.

In Equation (8), d_v is the unit drilling direction returned by the ray cast of Pass 1, expressed in the tangent frame of the hit point; d_{vup} is the drill’s reference up-direction, re-orthogonalized against d_v by Gram–Schmidt projection in the second line so that the frame is orthonormal; and d_{vbi} is the resulting bitangent, which completes the right-handed drilling frame $[d_v \mid d_{vbi} \mid d_{vup}]$. The brush sample $p \in [0, 1]^3$ is the packed RGB texel of the cylindrical brush texture at the current brush-space coordinate; d_p is that sample unpacked by Equation (5) and rotated into the drilling frame; $\alpha \in (0, 1]$ is the per-frame blend weight controlling how quickly successive contacts accumulate; and c_{prev} and c_{new} are the packed displacement-map texel values before and after the update.

Applied to a planar surface, the resulting cavity has the shape of the cylindrical brush pattern (Figure 8) and is free of the stretched-triangle artifact.

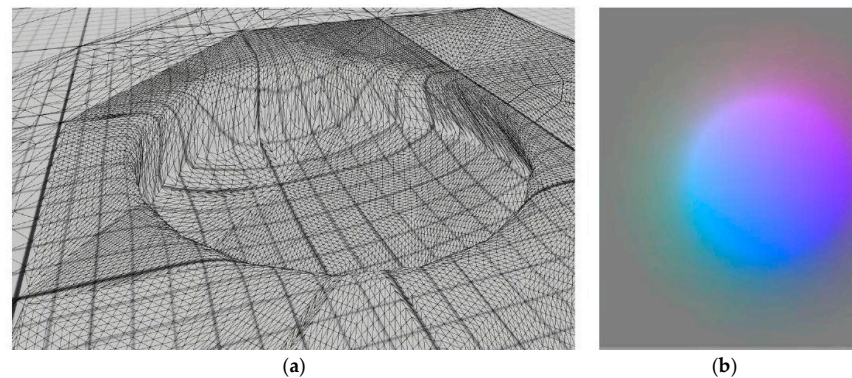


Figure 8. Patterned drawing in practice. (a) Rendered result on a tessellated surface: a clean cylindrical cavity with smooth rim triangulation. (b) The cylindrical brush texture; each color channel encodes one tangent-space component of the displacement vector.

3.9. Pass 3: Tessellation Map Generation

Uniform tessellation at the highest supported factor yields the best visual fidelity but destroys performance by subdividing flat regions that need no detail. We instead compute a tessellation amount map whose pixel values are proportional to the local curvature of the displacement map, so that subdivision concentrates where the surface bends more. Curvature is estimated by an edge-detection operator applied to the displacement texture.

We evaluated five curvature estimators: implicit curvature, slope curvature, derivative curvature, circular curvature, and the Sobel filter [59]. Figure 9 shows each estimator's output for a test drilling site; Table 1 reports per-estimator mean squared error (MSE) and structural similarity measure (SSIM) [60] against the uniform-tessellation reference, on both color and depth output. We use uniform tessellation at the highest supported factor as a reference rather than as a ground truth, because it applies the maximum subdivision to every triangle regardless of curvature and therefore produces the highest-fidelity geometric approximation available under the same shader path. Under this framing, the reported MSE on depth output is itself a geometric error measure, expressed as squared surface displacement in normalized depth-buffer units, while SSIM measures pixel-level similarity.

Table 1. Curvature-estimator comparison against the uniform-tessellation reference, on color and depth output. Uniform tessellation at the highest supported factor is used as a high-fidelity reference (not ground truth in an absolute sense). Lower MSE and higher SSIM indicate closer agreement with that reference; MSE on the depth output serves as a geometric error measure in normalized depth-buffer units, and SSIM measures pixel-level similarity.

Estimator	Color MSE	Depth MSE	Color SSIM	Depth SSIM
Implicit curvature	5.460	2.780	0.99586	0.99740
Slope curvature	0.550	6.690	0.99785	0.99478
Derivative	0.550	3.660	0.99786	0.99653
Circular curvature	0.200	0.080	0.99865	0.99944
Sobel filter	0.820	0.004	0.99708	0.99992

All five estimators were evaluated under identical conditions and without per-estimator tuning. Each operator was applied in its standard 3×3 kernel form with unit texel spacing, its response was taken as the gradient magnitude of the displacement texture, and that magnitude was linearly normalized over the $[0, 1]$ range of the tessellation-amount texture before being mapped to the tessellation-factor range. No sensitivity threshold, hysteresis, or non-maximum suppression was applied to any estimator, because thresholding would discard the graded response that the max-pooled mipmap chain of Pass 4 depends on. The single free parameter shared by all five operators is therefore the normalization

scale, and that scale is fixed by the texture range rather than chosen. Table 1 is consequently a default-parameter comparison: it establishes which operator localizes displacement edges best under a common normalization, not which operator is best after per-operator optimization. A tuned study, in which each estimator is allowed its own threshold and kernel width, could narrow the gap between the leading estimators and is left as future work.

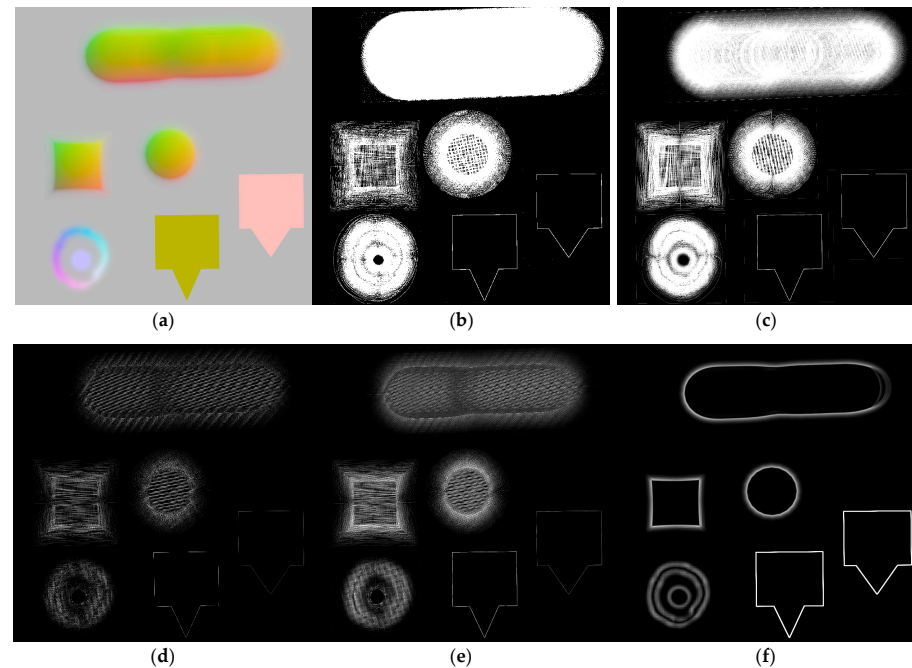


Figure 9. Tessellation-amount map from five curvature estimators. (a) Displacement map. (b) Implicit curvature. (c) Slope curvature. (d) Curvature by derivative. (e) Circular curvature. (f) Sobel filter. The Sobel filter produces the cleanest edge localization with minimal interior noise.

The Sobel filter produces the highest depth-output SSIM (0.99992) and the lowest depth-output MSE (0.004), making it our default. Once the tessellation map is produced, we compute its mipmap pyramid (Figure 10) using max pooling rather than average pooling, so that high-curvature edges are preserved at coarser levels.

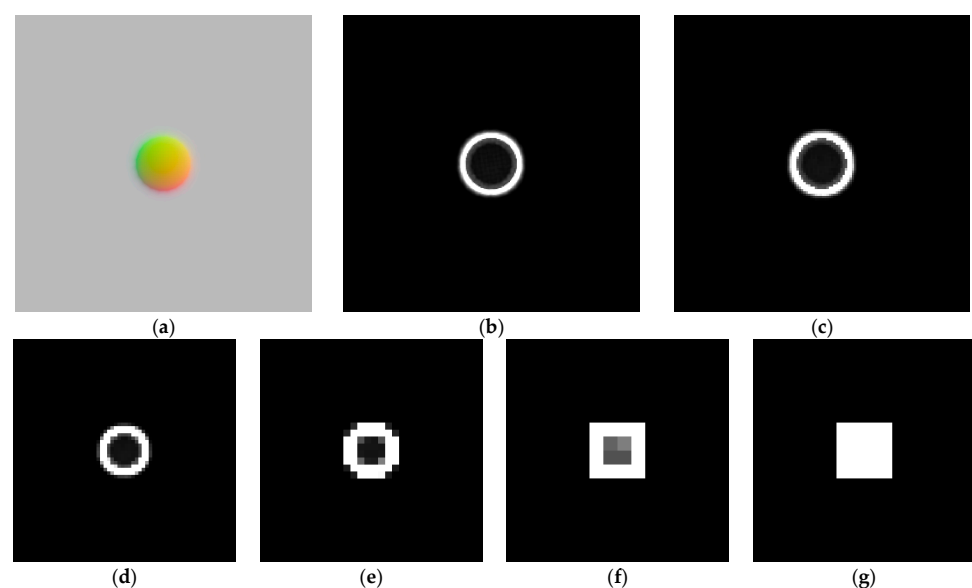


Figure 10. Edge detection and tessellation-map mipmap construction. (a) Input displacement map. (b) Sobel-filtered tessellation amount map. (c–g) 2nd through 6th mipmap levels, computed with max pooling to preserve edge values.

3.10. Pass 4: Tessellation Amount Sampling

Each triangle receives three edge factors and one inner factor. Crack-freeness across a shared edge requires that both adjacent triangles produce the same factor for that edge. Because the two triangles share only two vertices (call their UVs u_i and u_j), every input to the edge-factor computation must be a function of u_i and u_j alone; any dependence on a triangle's third, non-shared vertex introduces an asymmetry that would manifest as a crack.

This requirement has two consequences that we handle with one rule applied independently to each of the three edges. First, the sampled tessellation-map values at the endpoints must be identical for both triangles. Second, the mipmap level from which those values are sampled must also be the same for both triangles. We therefore assign a mipmap level to each edge, not to the triangle as a whole, using only that edge's endpoints:

$$L_{ij} = \frac{1}{2} \max(|u_{i,x} - u_{j,x}|, |u_{i,y} - u_{j,y}|)$$

$$m_{ij} = \min(M - 1, \min\{m \in \{0, \dots, M - 1\} : L_{ij} < s_m\}), \quad s_m = 2^m / N \quad (9)$$

$$f_{ij} = \max(T_{m_{ij}}(u_i), T_{m_{ij}}(u_j))$$

where L_{ij} is the length of the edge between vertex i and vertex j , measured in UV space, N is the base texture resolution in texels, M is the number of pre-computed mipmap levels (each stored as a separate render texture, see Section 3.9 and Figure 11), s_m is the texel size in UV units at mipmap level m , and $T_m(u)$ samples the tessellation-amount texture of level m at UV u using unfiltered lookup (tex2Dlod at the base level of the pre-computed target). The Chebyshev norm in Equation (9) is chosen so that L_{ij} bounds the axis-aligned span of the edge in UV space (the quantity relevant to texel coverage) and the factor $1/2$ provides a one-level oversampling margin against feature aliasing. Equation (9) selects the finest mipmap whose texel is at least as large as L_{ij} , so a small displaced feature enclosed by a long edge is never missed. If the admissible set in Equation (9) is empty, the selected level is clamped to $M - 1$ as detailed next.

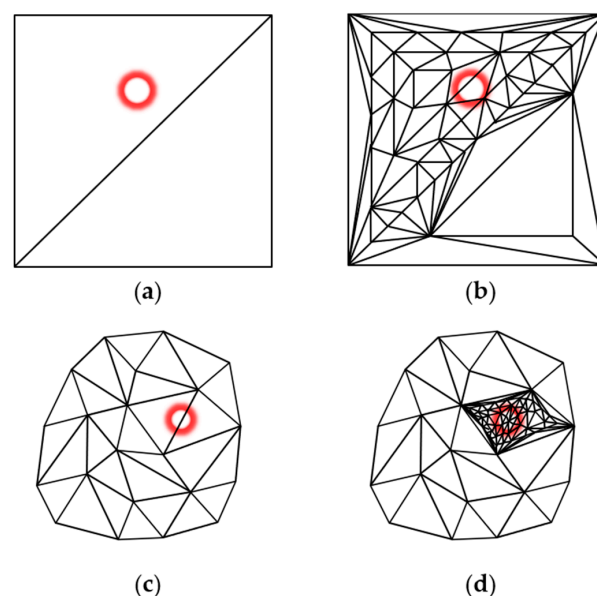


Figure 11. Tessellation amount sampling with mipmaps. (a) A large triangle that encloses the drilling cavity (marked as a red circle); its UV span is large, so a coarse mipmap is sampled. (b) The resulting tessellated version. (c) A small triangle entirely inside the cavity; its UV span is small, so the finest mipmap is sampled. (d) The resulting highly tessellated version.

As written without a fallback, Equation (9) is undefined when $L_{ij} \geq s_{[M-1]}$, which at 4096^2 with $M = 11$ means $L_{ij} \geq 0.25$, a range Equation (9) admits. Equation (9) is therefore amended to clamp explicitly to the coarsest stored level: $m_{ij} = \min(M - 1, \min\{\min\{0, \dots, M - 1\}; L_{ij} < 2^{m/N}\})$, where the inner minimum is taken to be $M - 1$ when the set is empty. The $m \in \{0, \dots, M - 1\}$ set can be empty for a long edge. The clamp preserves the crack-free property, because both incident triangles compute the same L_{ij} and therefore clamp together to the same level, leaving f_{ij} a function of the two shared endpoints alone. The reported level counts include the base level $m = 0$, and the max-pooled chain terminates at 4×4 rather than at 1×1 . We measured the maximum UV-edge span on each production asset: 0.154 on R_Humerus, where “R” denotes the right anatomical side; 0.011 on R_Scapula, and 0.006 on Glenoid_Fossa, all below the 0.25 threshold at which the clamp would engage, so the empty case does not arise on the assets evaluated here. The distribution is strongly concentrated: on R_Humerus the median span is 0.004, the 99th percentile is 0.013, and the second largest is 0.021. The 0.154 maximum is a single edge, the closing chord of a 35-sided polygon in the source asset whose endpoints are geometrically adjacent, so the large span is an atlas-layout artifact rather than a long edge in three dimensions. That edge is incident to one face only and therefore lies among the index-boundary edges of Section 4.7 rather than the 4271 shared edges, so no agreement is required of it, and it lies at the shaft end of the asset, outside the greater-tuberosity territory evaluated in Section 4.5. It is the only edge on any asset that samples the coarsest stored level, which under max-pooling is conservative rather than incorrect, since a coarser level reports a larger tessellation amount and cannot cause a displaced feature to be missed.

Because L_{ij} , m_{ij} , T evaluated at level m_{ij} at either endpoint, and f_{ij} are all functions of only the two endpoint UVs, both triangles that share an indexed edge (v_i, v_j) compute the bitwise-identical factor f_{ij} . Under fractional_odd partitioning with consistent endpoint attributes and shared texture state, that factor equality is the condition required to prevent T-junctions on that edge. This holds independently of the two triangles shapes, of the camera position, and of any variation in the non-shared vertices, so the rule extends without modification to irregular triangulations, varying camera distances, and adversarial adjacent-triangle configurations on indexed shared edges. It does not by itself couple geometrically coincident but index-duplicated UV-seam copies; those are handled separately in Section 3.14. The inner factor is derived from the three edge factors by standard triangle-tessellation bookkeeping. Because it governs subdivision strictly inside a patch it never appears on any shared edge, and so does not need to be a function of any single edge endpoints.

3.11. Pass 5: Vertex Displacement

In the Domain Shader, each generated vertex samples the displacement map at its UV coordinate u , computes object- or world-space displacement vector, $\delta_w(u)$, from Equation (6), and updates the vertex position with Equation (7). This pass operates identically on tessellated and original vertices, so that the rim of the cavity is topologically seamless.

3.12. Pass 6: Fragment Normal Correction

Because the underlying mesh is unchanged, per-vertex normals computed from the original geometry are no longer correct after displacement. We reconstruct a corrected fragment normal from finite differences in the displaced surface. Let $u_A = (u, v)$, $u_B = (u, v + \Delta v)$, and $u_C = (u + \Delta u, v)$, where Δu and Δv are one-texel offsets in UV space. Let P_A , P_B , and P_C be the displaced positions computed by Equation (7). Then the corrected normal is:

$$P_i = x(u_i) + \delta_w(u_i), \quad i \in \{A, B, C\}$$

$$n_f = \text{normalize}((P_B - P_A) \times (P_C - P_A)) \quad (10)$$

where A , B , and C are three sample points, $P_i \in \mathbb{R}^3$ is the resulting displaced position of sample i , evaluated by Equation (7), $P_B - P_A$ and $P_C - P_A$ are the two finite-difference tangent vectors spanning the displaced surface at the fragment, approximating $\partial P / \partial v$ and $\partial P / \partial u$ scaled by Δv and Δu , $\times$ denotes the vector cross product; $\text{normalize}(\cdot)$ rescales its argument to unit length; and n_f is the corrected unit fragment normal used for shading in place of the interpolated base-mesh normal $n(u)$, which no longer describes the surface after displacement.

3.13. Implementation Details

The drill ray is cast against a collider built from the undisplaced base mesh rather than from the displaced surface. The cast returns the hit UV coordinate together with the interpolated normal, tangent, and bitangent at that point, which are orthonormalized before use as the drilling frame of Equation (8). Because the collider is the base mesh, the reported contact point lags the visible cavity floor by at most the current displacement magnitude, which is bound by s_n . That bound is one reason the method is presented for shallow cortical preparation rather than for very deep sockets (see Section 5).

The displacement map, the tessellation-amount map, and every level of the mipmap chain are allocated as off-screen render textures with a floating-point channel format where the device supports it, and every sample and arithmetic operation in Equations (4)–(9) is performed in linear space with sRGB conversion disabled on both read and write. Fixed-point 8-bit storage is insufficient here because the iterative update of Equation (8) accumulates quantization error across frames, which becomes visible as terracing on shallow cavity walls.

Pass 2 does not use fixed-function output blending. Through the copy-and-update of Section 3.8, the previous texel is read from the temporary copy, unpacked by Equation (5), combined with the rotated brush sample by the linear interpolation in the last line of Equation (8), and written packed to the persistent displacement map, with output blending disabled, so that the blend weight α is the only mechanism governing accumulation. A classical two-buffer ping-pong, in which two persistent displacement maps swap roles each frame with no copy, would avoid that extra full-resolution blit; we retain copy-and-update instead so that the displacement map remains at a stable resource address for Pass 3, the mipmap chain, and the domain-shader samples that follow, at the cost of one temporary full-resolution copy (without a mipmap chain) per contact frame.

Equation (9) returns a floating-point factor within the range accepted by the Tessellator stage, and that value is passed to the hull shader unquantized. Rounding to integers is deliberately avoided because rounding introduces a discontinuity at which two adjacent triangles whose float factors straddle a rounding boundary are assigned different integers, reintroducing precisely the mismatch that Equations () exist to prevent. Hardware tessellation uses the `fractional_odd` partitioning mode, declared explicitly on the Unity HDRP hull shader. Edge factors are passed to the hull shader as unquantized floating-point values, as stated above.

3.14. Mesh Topology, UV Discontinuities, and Boundary Edges

Equation (9) are defined on indexed shared edges, that is, on pairs of triangles that reference the same two vertex indices. Production anatomical assets contain three further edge classes that must be treated separately. Their handling audited on the production of humerus in Section 4.7.

A UV discontinuity is an atlas (texture image) chart seam. Because the brush of Pass 2 is a disk in UV space, a brush centered close enough to a chart border writes part of its footprint into a neighboring island and produces a torn or duplicated imprint. The pipeline

does not stitch charts. Instead, admissible drill sites are constrained to lie inside a single UV island with a margin of at least one brush radius, obtained once per asset from a distance transform of the atlas mask.

Duplicated vertices are the geometric consequence of the same seam: one 3-D position carries a distinct UV on each chart side, so the indexed edge along the seam appears to be a boundary even on a topologically closed bone. Because the crack-free rule is defined on shared indices, the two copies are never coupled; keeping the brush off the seam leaves both copies undisplaced and the rendered surface continuous.

Non-manifold regions, in which three or more faces meet a single indexed edge, are not repaired. The hull shader operates per triangle and never welds, splits, or re-indexes geometry, so mesh topology and genus are invariant under the pipeline. Non-manifold input would therefore remain non-manifold, and is excluded by asset preparation rather than by the method.

A boundary edge has no adjacent triangle, so no agreement is required, and its factor is still computed from its two endpoints alone. Displacement cannot create a new boundary loop; that topological limitation is quantified in Section 5.

4. Results

All experiments ran on Intel Core Ultra 9 2.50 GHz, NVIDIA GeForce RTX 5090 with 32 GB RAM, 128 GB system RAM, Windows 11 under Unity 2019.3.5f1 with the HDRP renderer. We report six categories of results: (i) a comparison of five curvature estimators for the tessellation-amount map (Section 4.1); (ii) a GPU-timed execution-time breakdown of every pipeline pass as a function of displacement-map resolution, with mean, standard deviation, and p50/p95/p99 percentile latencies (Section 4.2); (iii) end-to-end frame rates across three synthetic test scenes compared against uniform (distance-based) tessellation (Section 4.3); (iv) visual quality of the Sobel-driven adaptive tessellation against uniform tessellation on a pixel-difference basis (Section 4.4); (v) evaluation on production ViRCAST anatomical meshes under clinically derived drilling trajectories, including geometric error measured against the continuous displacement field rather than against a denser rendering (Sections 4.5 and 4.6); and (vi) empirical verification of the crack-free rule, an audit of the mesh-topology classes of Section 3.14, and a component ablation (Sections 4.7 and 4.8).

4.1. Curvature Estimator Comparison

Table 1 reports MSE and SSIM of the five curvature-based tessellation maps against ground-truth uniform tessellation, measured both on the color output and the depth output. The circular-curvature estimator achieves the best color-output similarity (SSIM 0.99865; MSE 0.200), while the Sobel filter achieves the best depth-output similarity (SSIM 0.99992; MSE 0.004). Because the depth output more directly represents the physical surface shape and because the Sobel filter is trivially parallelizable, it is our default.

4.2. Per-Pass Execution Time

Table 2 reports per-pass wall-clock costs for every stage of the pipeline as a function of displacement-map resolution. All measurements were collected on the target hardware over 1000 consecutive drilling frames per resolution, with a 100-frame warm-up discarded, using the instrumentation described in Section Measurement Methodology. At 4K displacement-map resolution, the update that precedes the draw, comprising the CPU ray cast and drawing-vector computation (Pass 1) together with the three blit stages (Pass 2, Pass 3, and the mipmap chain), completes in 0.281 ms mean, or 0.292 ms including the Sync column. Adding the tessellated draw (Passes 4 to 6, measured collectively) brings the arithmetic end-to-end sum to 0.397 ms mean (p95 = 0.424 ms, p99 = 0.433 ms). These end-to-end

values are the per-frame sums of the reported components; they are not an independent single-interval GPU timestamp. Column values are rounded independently, so the end-to-end figure can differ from the displayed component sum by up to 0.001 ms; at 4096² the displayed components sum to 0.396 ms against a reported end-to-end mean of 0.3968 ms. The mipmap count grows logarithmically with resolution and saturates at 11 levels for 4096 × 4096 textures.

Table 2. Per-pass wall-clock time versus displacement-map resolution, reported as mean +/− standard deviation (in milliseconds) over N = 1000 drilling frames per resolution after 100 warm-up frames. Pass 1 is CPU-timed (Physics.Raycast plus tangent-space setup). Passes 2, 3, and the mipmap chain report the CPU-observed interval from dispatch through GL.Flush(); on the Direct3D 11 path used here, Flush submits the queue and does not wait for GPU completion, so those columns bound command-issue/submission cost rather than isolated GPU execution latency. Passes 4–6 are captured collectively as the Unity Render.Mesh recorder delta on the tessellated bone draw (GPU marker when available). Sync reports empty-Flush driver overhead. End-to-end is the arithmetic per-frame sum of all preceding columns and is therefore a composite accounting quantity, not a separate hardware timestamp. Percentile latencies at 4K resolution appear in Table 3.

Res. (px)	Mipmap Levels	Pass 1 (CPU)	Pass 2 (Submit)	Pass 3 (Submit)	Mipmap (Submit)	Passes 4–6 (Draw)	Sync	End-to-End (Sum)
128 ²	6	0.011 ± 0.001	0.173 ± 0.007	0.017 ± 0.001	0.007 ± 0.000	0.073 ± 0.005	0.006 ± 0.001	0.287 ± 0.012
256 ²	7	0.012 ± 0.001	0.174 ± 0.007	0.028 ± 0.001	0.011 ± 0.001	0.074 ± 0.005	0.006 ± 0.001	0.306 ± 0.012
512 ²	8	0.012 ± 0.001	0.186 ± 0.007	0.033 ± 0.002	0.013 ± 0.001	0.076 ± 0.005	0.007 ± 0.001	0.326 ± 0.013
1024 ²	9	0.014 ± 0.001	0.188 ± 0.008	0.036 ± 0.002	0.014 ± 0.001	0.080 ± 0.006	0.007 ± 0.001	0.338 ± 0.014
2048 ²	10	0.016 ± 0.001	0.193 ± 0.008	0.042 ± 0.002	0.016 ± 0.001	0.088 ± 0.006	0.008 ± 0.001	0.364 ± 0.015
4096 ²	11	0.021 ± 0.001	0.194 ± 0.008	0.048 ± 0.003	0.018 ± 0.001	0.105 ± 0.008	0.010 ± 0.001	0.397 ± 0.016

Table 3. Percentile latencies at 4096² displacement-map resolution (N = 1000 measurement frames after 100 warm-up frames). Reporting the p50, p95, and p99 percentiles complements the mean and standard deviation of Table 2, characterizing tail behavior that a mean alone can obscure. All values are milliseconds.

Pass	Mean	σ	p50	p95	p99	N
Pass 1: Drawing-vector calculation (CPU)	0.0213	0.0013	0.0213	0.0235	0.0244	1000
Pass 2: Displacement-map generation (submit)	0.1941	0.0078	0.1939	0.2070	0.2120	1000
Pass 3: Tessellation-map/Sobel (submit)	0.0481	0.0025	0.0481	0.0524	0.0542	1000
Mipmap chain (submit)	0.0185	0.0009	0.0185	0.0201	0.0209	1000
Passes 4–6: Tessellated draw (draw marker)	0.1051	0.0077	0.1044	0.1184	0.1250	1000
GPU sync/empty-Flush overhead	0.0101	0.0008	0.0101	0.0114	0.0119	1000
End-to-end drilling update (arithmetic sum)	0.3968	0.0160	0.3965	0.4241	0.4331	1000

Measurement Methodology

All performance measurements reported in Tables 2 and 3 were collected with a dedicated Unity profiling component instrumenting each of the six pipeline passes on the same test platform used elsewhere in this study (Unity 2019.3.5f1 with the built-in HDRP renderer, Windows 11). The rendered resolution was fixed at 1920 × 1080 throughout. For every displacement-map resolution $N \in \{128, 256, 512, 1024, 2048, 4096\}$, we ran 100 warm-up frames (discarded) followed by 1000 measurement frames, forcing a synthetic drill contact every frame so that all six passes executed unconditionally.

Pass 1 executes entirely on the CPU: it casts a single ray against the base mesh collider, computes the drawing vectors, and writes the resulting shader uniforms. It dispatches no Graphics.Blit and writes no texture, and was timed directly with a System.Diagnostics.Stopwatch. A single ray does not amortize the cost of a GPU dispatch and readback, and the contact point is needed on the CPU by both Pass 2 and the haptic subsystem, so the cost of this pass is independent of texture resolution.

Pass 2 (displacement-map generation), Pass 3 (tessellation-map generation), and the max-pool mipmap chain each dispatch one or more Graphics.Blit calls. Pass 2 specifi-

cally comprises the copy-and-update pair of Section 3.8 (copy into a temporary render texture, then full-screen update back to the persistent map); both blits are inside the timed Pass 2 interval, so the reported Pass 2 cost already includes the extra copy. The Pass 2 figure of 0.194 ms at 4096², which is the largest single component in Table 2 and roughly four times Pass 3 (0.048 ms), reflects both blits rather than a single update pass. Each was timed with a CPU stopwatch covering the dispatch, Graphics.ExecuteCommandBuffer of a matched BeginSample/EndSample marker pair, and Unity GL.Flush() before the stopwatch was stopped. Because GL.Flush() on Direct3D 11 submits queued commands without blocking until the GPU has finished those commands, the Pass 2, Pass 3, and Mipmap columns in Tables 2 and 3 should be interpreted as CPU submission/issue costs for those stages, not as isolated GPU execution latencies. We time the stages sequentially with a Flush between them, so the measurement protocol does not allow GPU overlap among those timed intervals; in production rendering without intervening Flush fences the same blits may pipeline on the GPU. The end-to-end column is the arithmetic sum of Pass 1, Pass 2, Pass 3, Mipmap, Passes 4–6, and Sync on each frame; it is not derived from a separate start-end GPU timestamp query. Column values are rounded independently, so the end-to-end figure can differ from the displayed component sum by up to 0.001 ms; at 4096² the displayed components sum to 0.396 ms against a reported end-to-end mean of 0.3968 ms.

Passes 4 (tessellation-amount sampling), 5 (vertex displacement), and 6 (fragment normal correction) execute inside a single tessellated draw call of the bone mesh, dispatched by Unity's rendering loop through the Hull, Domain, and Pixel shader stages, respectively. They cannot be individually resolved by any CPU-side timer without editing the HLSL to introduce conditional-compilation variants that toggle each stage on and off. Such a modification would alter the shader relative to the production build and was therefore avoided. Their combined cost is reported as a single row (Passes 4–6), captured via the Unity Render.Mesh recorder API and isolated to the drilled bone by baseline subtraction against a control frame in which the drilled object is culled but all other scene content is preserved. The row labeled Sync in Table 2 reports the cost of an empty GL.Flush() issued after the blit stages and is included in the end-to-end sum for full accounting. Passes 4–6 are obtained from a different timing reference (the Render.Mesh recorder or frame-delta fallback) than Passes 2/3/Mipmap (CPU stopwatch around dispatch + Flush); the end-to-end sum therefore combines CPU submission intervals, a draw-marker interval, and Sync, and should be read as a composite budget estimate rather than a single-instrument GPU latency.

From the 1000 samples per pass per resolution, Table 2 reports the mean and sample standard deviation, and Table 3 reports the p50, p95, and p99 percentiles at 4096 × 4096. These statistics characterize within-run, frame-to-frame variability under a single continuous capture. We did not repeat each resolution as independent cold-start runs in this study, so run-to-run confidence intervals across separately launched sessions are not reported; the N = 1000 within-run percentiles remain the variability evidence for Tables 2 and 3.

Table 3 shows that even the p99 tail of the end-to-end update at 4K resolution (0.433 ms) is more than 38× below the 16.7 ms graphics-frame budget of a 60 Hz display, giving the rendering pipeline substantial headroom before it can become a bottleneck for the overall graphics-frame rate. This paper characterizes rendering performance only; the ViRCAST haptic loop, its cycle time, jitter, and end-to-end visual-haptic latency are properties of a separate subsystem and are addressed in Section 5 and deferred to a follow-up study.

4.3. End-to-End Frame Rates

We evaluated three test scenes (Figure 12), each containing objects of 121 vertices and 200 triangles. Scene I (control) places 60 objects far enough from the camera that no triangle

is tessellated; Scene II views the same 60 objects from an oblique angle so that roughly half are semi-tessellated and ten are densely tessellated; Scene III contains six objects close enough to the camera to always be maximally tessellated. For each scene we compared (a) no drilling, (b) drilling with uniform (distance-based) tessellation, and (c) drilling with adaptive tessellation.

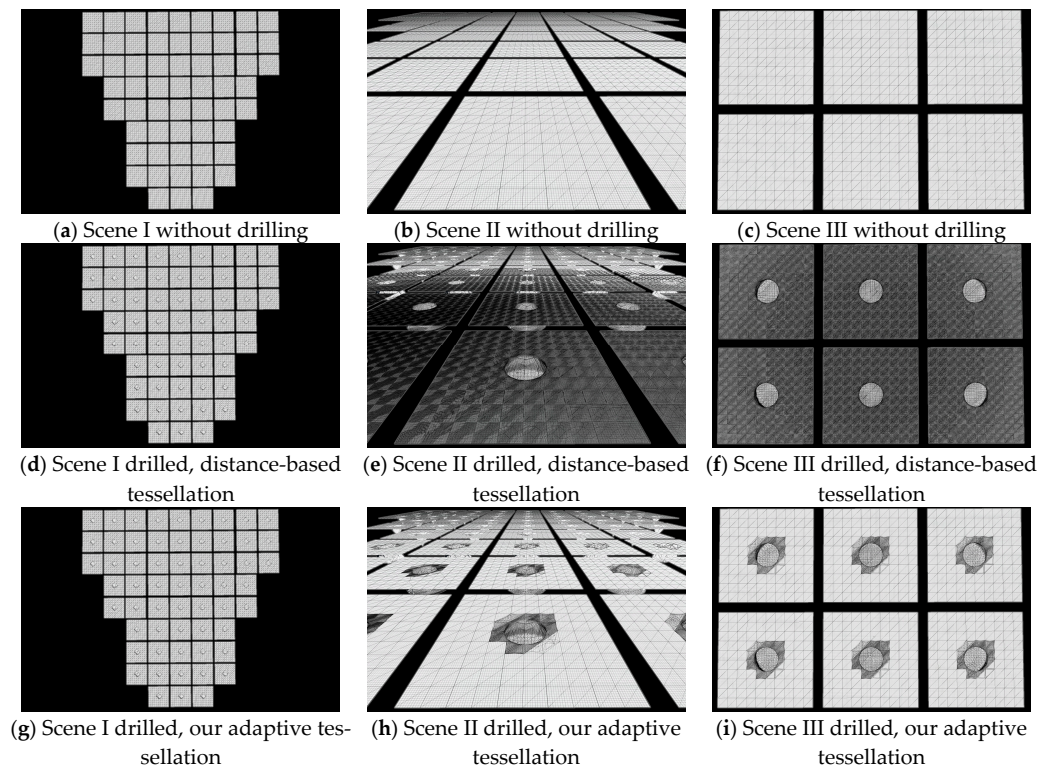


Figure 12. Three test scenes under three conditions. Columns: Scene I, II, III. Rows: (a–c) no drilling; (d–f) drilled with uniform distance-based tessellation; (g–i) drilled with our adaptive tessellation. These are synthetic load scenes rather than surgical scenarios; they isolate tessellation cost from base-mesh rasterization. Visual output is closely similar between uniform and adaptive tessellation while the frame rates differ substantially (Table 4).

Table 4. End-to-end frame rates (frames per second) across the three test scenes and three drilling conditions, with relative differences expressed as percentages against the no-drilling baseline and against uniform tessellation.

Condition	Scene I (fps)	Scene II (fps)	Scene III (fps)
No drilling	120.14	120.18	158.07
Uniform tessellation	120.11	100.07	119.89
Adaptive (ours)	120.07	119.78	157.96
Δ vs. no-drilling (%)	−0.06	−0.33	−0.07
Δ vs. uniform (%)	−0.03	+16.46	+24.10

Table 4 reports the resulting frame rates. On Scene I (control) both methods match the no-drilling baseline, as expected. On Scene II, uniform tessellation loses 16.5% of its frame rate to over-tessellation of semi-visible objects, while our adaptive method loses only 0.33%. On Scene III, where every object receives maximum tessellation, our method delivers 24.1% higher frame rate than uniform tessellation by avoiding tessellation of the flat surfaces that surround the drilling sites.

The frame rates in Table 4 are steady-state means recorded over a fixed capture window with the same drilling trajectory replayed in each condition. Run-to-run dispersion was not logged for this capture, so no standard deviation or confidence interval can be reported for Table 4. The dispersion characterization of the pipeline itself, which is the quantity

this paper claims, is given instead in Tables 2 and 3, where every entry carries a standard deviation and, at 4K resolution, p50/p95/p99 latencies over $N = 1000$ frames. We record this as a limitation of Table 4 rather than repair it by re-running the synthetic capture, because Section 4.5 supersedes it with measurements on production anatomical meshes.

The scenes in this subsection are deliberately synthetic and small, at 121 vertices and 200 base triangles per object, so that the tessellation stage rather than base-mesh rasterization dominates the measurement. They are not representative of anatomical load, and the 24.10% figure should not be extrapolated to production meshes. Section 4.5 repeats the comparison on the ViRCAST humerus, where the base mesh is an order of magnitude larger and the drilling trajectory is clinically derived.

4.4. Pixel-Level Visual Assessment

To characterize the pixel-level cost of adaptive tessellation relative to the high-fidelity uniform-tessellation reference of Section 3.9, we subtracted the two drilled-scene renderings pixel by pixel (Figure 13). Differences are concentrated in small, localized patches along the drill cavity's rim, where the adaptive method uses slightly coarser tessellation than the reference. The magnitude of these differences is quantified by the SSIM and MSE values in Table 1: for the Sobel variant used as the default, SSIM is 0.99708 on color output and 0.99992 on depth output, and MSE is 0.820 on color output and 0.004 on depth output. Interpreted geometrically, the depth-MSE of 0.004 corresponds to a mean squared surface displacement of 0.004 in normalized depth-buffer units, which is well below the per-texel resolution of the underlying displacement map.

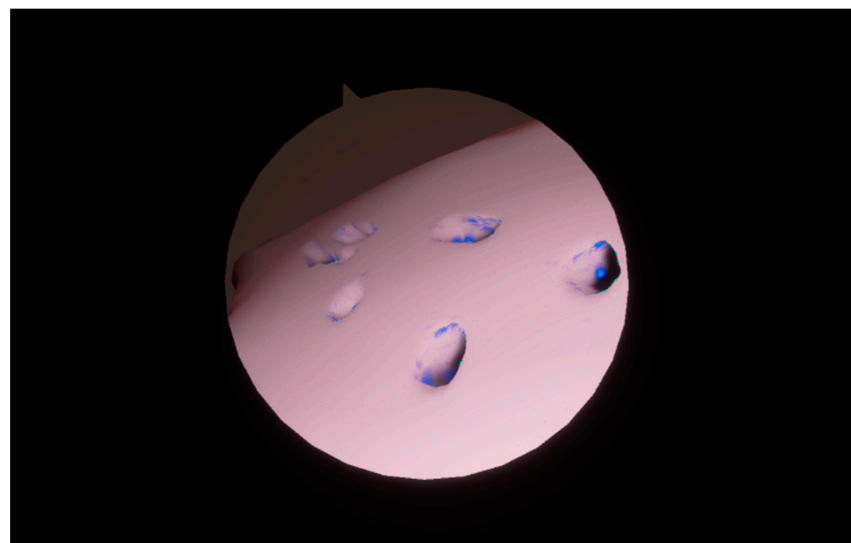


Figure 13. Pixel-level difference between uniform tessellation and our adaptive tessellation. Residual differences (blue) appear only as small patches at the drill-cavity rim, corresponding to the coarser local triangulation under adaptive tessellation.

These pixel and geometric measures describe how closely the adaptive result approaches the reference under a fixed viewing condition; they do not by themselves establish that the residual differences are imperceptible to a human observer. Whether the differences meet a specific perceptual threshold under supervised viewing, in the presence of expected surgical instruments, or in a trained-observer evaluation is a matter for a formal perceptual or expert-evaluation study, which is identified as future work in Section 5. What the measurements above support is a narrower and defensible claim: the residual differences are small in absolute magnitude, localized to the drill-cavity rim rather than distributed over the surface, and stable across the drilling depths, sites, viewing distances and instrument ori-

entations evaluated on the production anatomical meshes of Sections 4.5 and 4.6 (humeral head and scapular acromion) under clinically derived tool trajectories.

4.5. Anatomical Evaluation on Production ViRCAST Meshes

To close the load and realism gap of the synthetic scenes in Section 4.3, the evaluation was repeated on the anatomical assets used by ViRCAST itself. The primary asset is the right humerus (R_Humerus, 1536 vertices and 2896 base triangles); the scapular acromion and the glenoid fossa supply the high curvature and thin shell conditions of Sections 4.6 and 5. The trajectory follows the clinical medial row sequence: a 6 mm round burr prepares three bone beds on the greater tuberosity to a depth of 2.0 mm, at the approach used for footprint preparation prior to suture anchor placement. Figure 14 shows the sequence. The hardware triangle count after adaptive tessellation rises from 2896 to 243,024 once all three beds are open, and the increase is confined to the prepared footprint: cortex away from the working site is never refined.

Cortical bone-bed preparation at the greater tuberosity - 6 mm round burr, 2.0 mm deep, R_Humerus (1,536 v / 2,896 tri)

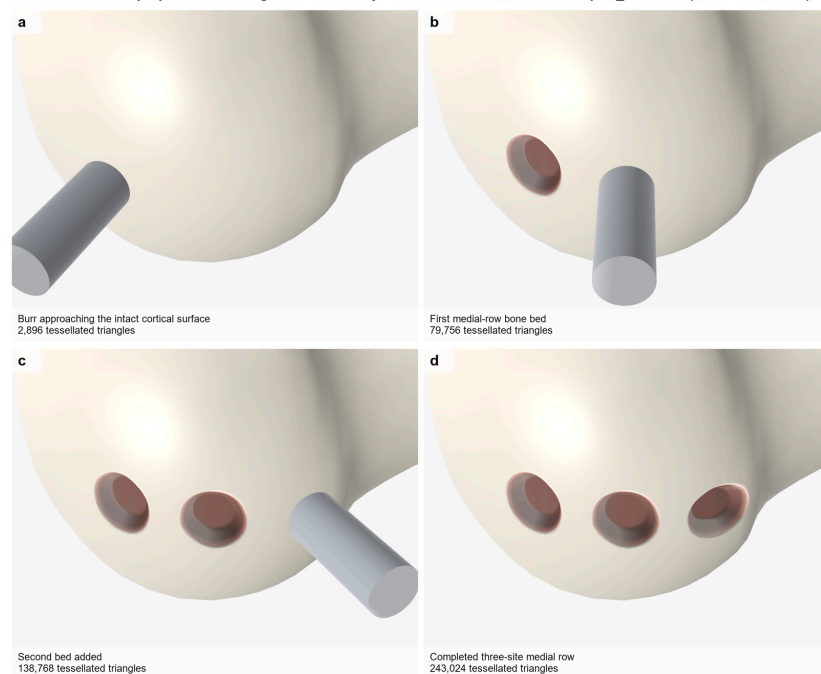


Figure 14. Cortical bone-bed preparation on the production ViRCAST R_Humerus asset (1536 vertices, 2896 base triangles). A 6 mm round burr cuts three medial-row beds to 2.0 mm depth. (a) Burr approaching intact cortex; (b–d) one completed bed added per panel. Sites are placed inside a single UV island so that the brush never straddles a chart boundary (Section 3.14). Hardware triangle count after adaptive tessellation rises from 2896 to 243,024 across the sequence, with refinement confined to the prepared footprint.

A cut depth of 2.0 mm is representative of clinical footprint preparation. Ruder et al. [61] decorticated the greater tuberosity to a mean depth of 1.7 mm in a cadaveric study of all suture anchor fixation, and reported that even this minimal decortication measurably reduced ultimate load to failure, which is why some surgeons limit or omit the step entirely to preserve cortical integrity. The beds modeled here therefore sit at the upper end of reported practice rather than below it, and the wall gradient bound of $g \leq 2.2$ established in Section 5 covers the full clinical range of this operation.

Figure 15 compares adaptive against uniform distance-based tessellation on the same prepared humerus under two matched conditions. At approximately equal cost, uniform factor 4 produces 107,152 triangles against 60,720 for the adaptive method, and the

wireframes show why the costs are not comparable in effect: the uniform configuration under-resolves the cavity rim while over-resolving the surrounding flat cortex. At approximately equal quality, uniform factor 50 requires 11,297,296 triangles, 186 $\times$ the adaptive count. Figure 16 examines one bed at close range. The wall gradient is bounded, so wall triangles remain near-isotropic, and the rim closes without cracks or T-junctions; this is the visual counterpart of the exhaustive agreement test reported in Section 4.7.

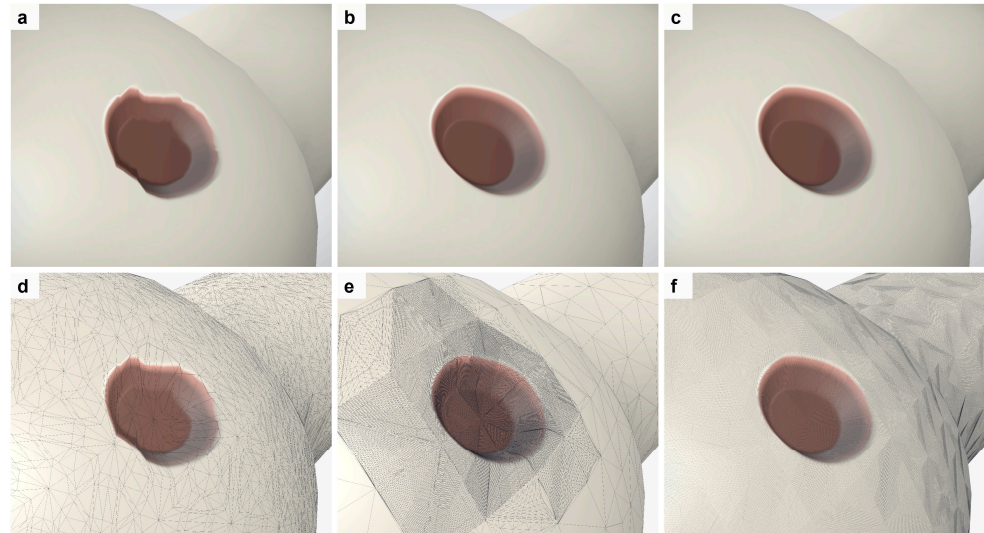


Figure 15. Adaptive versus uniform distance-based tessellation on the same prepared humerus. (a,d) Uniform factor 4: 107,152 triangles, 1.8 $\times$ the adaptive triangle budget. (b,e) Adaptive: 60,720 triangles. (c,f) Uniform factor 50: 11,297,296 triangles, the smallest uniform factor that matches adaptive quality at the rim. Top row shaded, bottom row wireframe. Uniform tessellation loses the cavity rim even when given 1.8 $\times$ the budget, because subdivision is spent uniformly across flat cortex rather than concentrated at the drill site; matching the adaptive result requires 186 $\times$ the triangles.

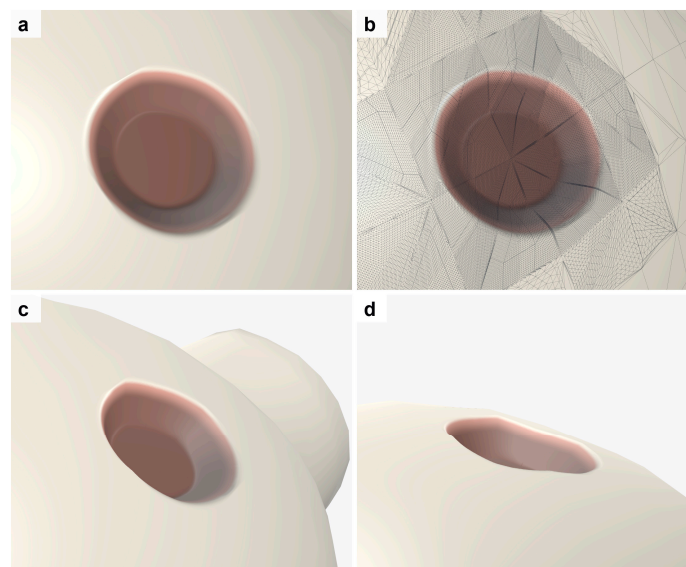


Figure 16. Detail of one 6 mm $\times$ 2.0 mm cortical bed on the prepared humerus. (a) Shaded, viewed down the burr axis. (b) The same view as a tessellated wireframe: refinement concentrates on the cavity and its rim while the surrounding cortex stays coarse. (c) Oblique view showing a closed rim with no cracks or T junctions. (d) Grazing view, in which the wall gradient and cut depth are visible. The wall gradient is capped at $g = 2.2$, bounding triangle stretch at 2.4 $\times$ and keeping wall triangles near isotropic. The absence of cracks along the rim is the visual counterpart of the exhaustive per edge agreement test reported in Section 4.7.

Table 5 lists the parameters for anatomical evaluation.

Table 5. Parameters of the anatomical evaluation reported so that the measurements in Sections 4.5–4.7 can be reproduced. Triangle counts are hardware counts after tessellation, measured on the target platform. Texture-memory entries are computed from resolution and channel format, not read back from the driver: one row for resident pipeline maps and one for the transient full-resolution temporary copy used by the Pass 2 copy-and-update of Section 3.8; values are given for scale rather than as a driver measurement.

Parameter	Value
Primary anatomical asset	ViRCAST R_Humerus (right humeral head, production asset)
Base mesh size	1536 vertices/2896 triangles
Secondary assets	Scapula (acromion site, Section 4.6); Glenoid_Fossa (≈ 0.19 mm shell, Section 5)
Instrument/brush	6 mm round burr; cylindrical brush texture, tangent-space vector-packed
Cut depth	2.0 mm for the medial-row beds; swept over 0.8–3.0 mm in Section 4.6
Approach angle	Surface-normal for the beds; swept to 50° off-normal in Section 4.6
Displacement map	4096^2 atlas, floating-point channels, linear color space
Tessellation-amount map	4096^2 base plus 10 max-pooled mipmaps (11 levels total); coarsest 4×4 (not 1×1); clamp $m = \min(M - 1, \min \text{qualifying})$; max $L_{ij} = 0.154 < 0.25$ (empty case not reached)
Edge tessellation-factor range	1–64, fractional (passed unquantized, Section 3.13)
Triangles, base mesh	2896
Triangles, adaptive, one prepared bed	60,720
Triangles, adaptive, three prepared beds	243,024
Triangles, uniform factor 4 (equal cost)	107,152
Triangles, uniform factor 50 (equal quality)	11,297,296
Triangle-count reduction vs. uniform-50	$186\times$ at the greater tuberosity; $67\text{--}304\times$ across the nine conditions in Figure 17 and Table 6 (mean $186.4\times$)
Resident pipeline texture memory	≈ 0.36 GB (displacement map, tessellation-amount map, and mip chains at 4096^2 , four-channel half-float), computed from resolution and format rather than read back from the driver
Pass 2 peak (transient)	One full-resolution temporary copy without a mip chain (≈ 0.13 GB), released after the copy-and-update of Section 3.8

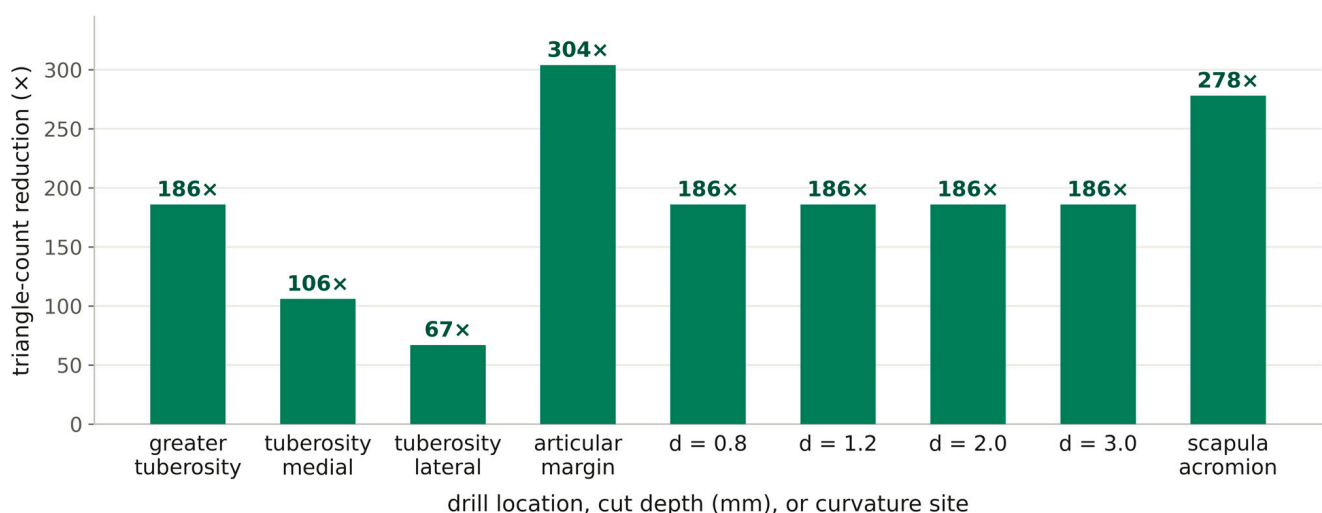


Figure 17. Triangle-count reduction by condition, defined as uniform factor-50 triangles divided by adaptive triangles; higher means fewer adaptive triangles at matched quality. The nine bars are the four humeral drill locations, four cut depths, and the scapular acromion—the same conditions as Table 6. Reductions range from $67\times$ (lateral tuberosity) to $304\times$ (articular margin), with a mean of $186.4\times$; the scapular acromion reaches $278\times$. The lowest values occur where the surface is already curved and the uniform reference is less wasteful; the highest occur where a small high-curvature feature sits in a large low-curvature neighborhood.

Table 6. Geometric fidelity summary for the same nine conditions as Figure 18 (four drill locations, four cut depths, and the scapular acromion). Each method column reports error against the continuous displacement field in millimeters at common UV query coordinates (Euclidean positional difference, not closest-point distance): N is the evaluation sample count in the cut window; RMS is the unweighted root-mean-square; p95, p99, and Max summarize the same sample set for every configuration. Uniform 4 is the near-budget reference; uniform 50 is the smallest uniform factor at least as accurate as adaptive. Because sample locations depend on each method’s tessellation factors, RMS values are not area-weighted surface integrals; the sampling bias inflates the adaptive RMS relative to an area-weighted integral, so the adaptive-versus-uniform-4 advantage is conservative. Reduction is uniform-50 triangles divided by adaptive triangles for the same asset (range $67\times$ – $304\times$; mean $186.4\times$). We measure depth SSIM against the uniform-50 rendering. All rows use production anatomical assets.

Condition	Adaptive RMS (mm)	Adapt N; p95/p99/Max (mm)	Uniform 4: RMS; N; p95/p99/Max (mm)	Uniform 50: RMS; N; p95/p99/Max (mm)	Adaptive Triangles	Reduction	Depth SSIM
Greater tuberosity	0.0020	N = 39,795; 0.0016/0.0077/0.1537	0.0668; N = 944; 0.1323/0.3320/0.7021	0.0009; N = 135,936; 0.0002/0.0039/0.0349	60,720	186×	0.99983
Tuberosity medial	0.0010	N = 53,218; 0.0010/0.0051/0.0295	0.0721; N = 912; 0.1831/0.3448/0.5471	0.0008; N = 131,328; 0.0005/0.0039/0.0295	114,030	99×	0.99999
Tuberosity lateral	0.0009	N = 71,487; 0.0010/0.0044/0.0204	0.0497; N = 1504; 0.1147/0.2463/0.4969	0.0006; N = 216,576; 0.0003/0.0026/0.0204	169,610	67×	0.99986
Articular margin	0.0019	N = 43,448; 0.0017/0.0081/0.1178	0.0787; N = 880; 0.1795/0.3974/0.6288	0.0012; N = 126,720; 0.0007/0.0049/0.0964	37,116	304×	1.00000
Depth 0.8 mm	0.0008	N = 39,795; 0.0006/0.0031/0.0615	0.0267; N = 944; 0.0529/0.1328/0.2808	0.0004; N = 135,936; 0.0001/0.0016/0.0140	60,720	186×	0.99996
Depth 1.2 mm	0.0012	N = 39,795; 0.0009/0.0046/0.0922	0.0401; N = 944; 0.0794/0.1992/0.4213	0.0006; N = 135,936; 0.0001/0.0024/0.0209	60,720	186×	0.99996
Depth 2.0 mm	0.0020	N = 39,795; 0.0016/0.0077/0.1537	0.0668; N = 944; 0.1323/0.3320/0.7021	0.0009; N = 135,936; 0.0002/0.0039/0.0349	60,720	186×	0.99983
Depth 3.0 mm	0.0031	N = 39,795; 0.0024/0.0115/0.2305	0.1002; N = 944; 0.1984/0.4980/1.0532	0.0014; N = 135,936; 0.0004/0.0059/0.0523	60,720	186×	0.99996
Scapula acromion (high curvature)	0.0083	N = 164,820; 0.0004/0.0033/0.2474	0.0164; N = 3776; 0.0316/0.0806/0.2590	0.0002; N = 543,744; 0.0001/0.0008/0.0171	204,790	278×	0.99998

4.6. Geometric Fidelity Against the Continuous Displacement Field

Uniform tessellation is a useful visual reference, but it is not geometric ground truth: it is a denser approximation of the same surface produced by the same shader path, and it inherits the same approximation error in kind if not in degree. For a geometric measure we therefore compare each tessellated surface against the continuous displacement field itself, which Equations (5)–(7) define analytically at every UV coordinate. The reported quantity is the Euclidean positional difference, at a common UV coordinate, between (i) the piecewise-linear surface obtained by displacing only the tessellation vertices (as the domain shader does) and interpolating within each generated micro-triangle, and (ii) the continuous displaced position obtained by sampling the displacement map at that same UV. It is not the shortest Euclidean distance to the continuous surface under a different closest-point correspondence. Query points are placed on a barycentric UV lattice over each base triangle that intersects the cut window; the lattice density increases with the local edge tessellation factors, so adaptive, uniform-factor-4, and uniform-factor-50 configurations are evaluated at different sample sets and sample counts. The tabulated RMS is an unweighted root-mean-square over those query samples and is therefore not, in general, an area-weighted surface integral that is strictly commensurate across methods. We note further that the direction of this sampling bias is unfavorable to the proposed method. Adaptive tessellation concentrates samples in high-curvature regions, where residuals are largest, so its unweighted RMS is inflated relative to an area-weighted integral, whereas the near-uniform UV sampling of the two uniform configurations is already close to area-weighted. The reported advantage of adaptive tessellation over uniform factor 4 is therefore

conservative: correcting for sampling density would widen rather than narrow it. Absolute error against the continuous field remains the primary geometric quantity reported for each method separately. The previously reported RMS values are unchanged; only their description was corrected (the metric was neither triangle mid-points nor a closest-point distance to the surface), and Table 6 now additionally lists N, p99, and Max for the uniform configurations from the same evaluation runs.

Figure 18 reports this error across the four conditions that vary in a real drilling task: cut depth, drill location on the bone, camera distance, and instrument approach angle. In every condition the adaptive result tracks the high-factor uniform configuration closely and is one to two orders of magnitude more accurate than the equal-cost uniform-4 configuration; given the sampling bias above, that cross-method advantage is a lower bound rather than an overclaim. Error grows with cut depth for all three configurations, from 0.0008 mm at 0.8 mm depth to 0.0031 mm at 3.0 mm depth for the adaptive method, which is consistent with the wall-anisotropy analysis in Section 5. Error is essentially independent of camera distance, confirming that the residual is a property of the mesh approximation rather than of the view.

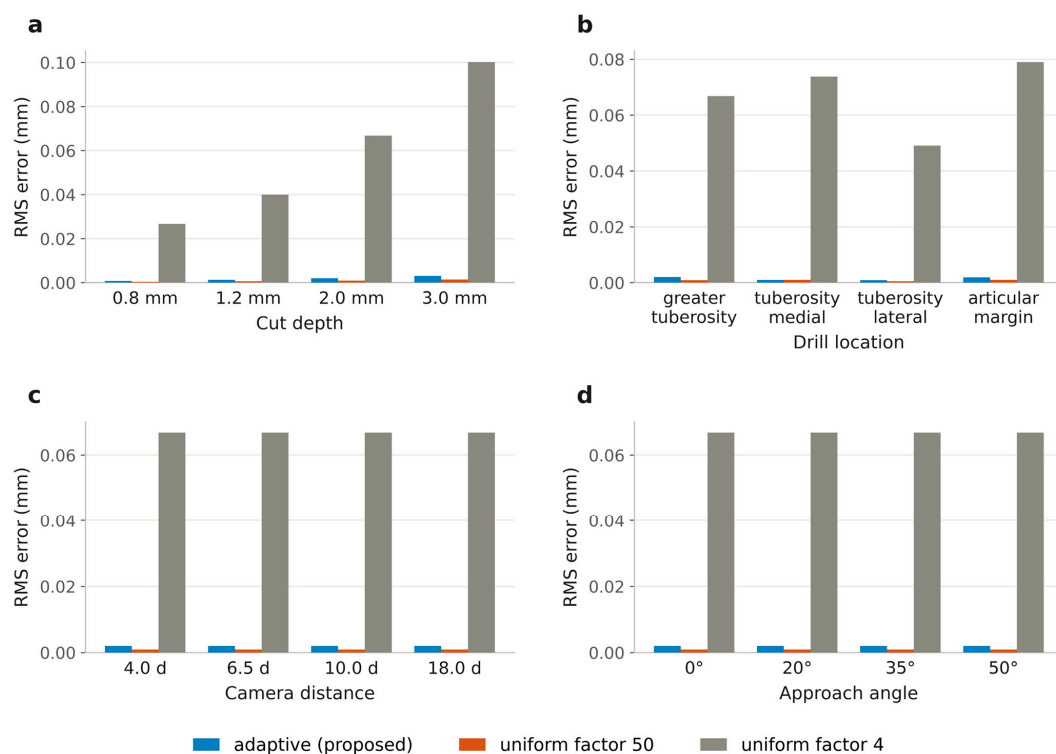


Figure 18. Geometric RMS error against the continuous displacement field, measured as the Euclidean positional difference at common UV query coordinates against the analytic field of Equations (5)–(7); lower is better. Blue—adaptive; orange—uniform factor 50; gray—uniform factor 4. Each panel is a one-factor sweep with the others held at the clinical default, not the nine-condition set of Table 6 and Figure 18: (a) cut depth in millimeters (four depths); (b) drill location on the anatomical surface (four humeral sites: greater, medial, and lateral tuberosity, and articular margin); (c) camera distance as a multiple of cavity diameter d; (d) instrument approach angle in degrees from the surface normal. The scapular acromion appears in Table 6 and Figure 18, not in this figure. Error grows with cut depth and is essentially independent of camera distance and approach angle, confirming that the residual is a property of the mesh approximation rather than of the view. Uniform factor 50 attains roughly half the error of adaptive but renders 67 to 304 times more triangles depending on the site (Figure 18); the comparison at matched cost is against uniform factor 4, which uses 1.8 times the adaptive triangle budget and is one to two orders of magnitude less accurate (a conservative lower bound under the sampling bias noted in the text).

Figure 18 quantifies what that fidelity costs. Against the uniform factor-50 reference, which is the smallest uniform factor at least as accurate as adaptive, the adaptive method uses between $67\times$ and $304\times$ fewer triangles depending on the site, with a mean reduction of $186.4\times$ across the nine conditions plotted in Figure 18 (four drill locations, four cut depths, and the scapular acromion). The lowest reduction is at the lateral tuberosity ($67\times$), where the surface is already curved and the uniform reference is therefore less wasteful; the highest is at the articular margin ($304\times$), where a small high-curvature feature sits in a large low-curvature neighborhood, with the scapular acromion next ($278\times$). Table 6 reports the same nine conditions as Figure 18. Viewing-distance and instrument-orientation sweeps are retained in Figure 17 for geometric error; they are omitted from Figure 18 and Table 6 because viewing distance does not change the adaptive triangle count under this protocol, and orientation-driven counts remain inside the $67\times$ – $304\times$ band, reaching $117\times$ at 50° off-normal (96,354 adaptive triangles).

Several conditions in Table 6 show p95 below RMS. That ordering is expected for a heavy-tailed error distribution: RMS is inflated by rare large residuals; whereas, p95 reflects the bulk of the mass. Rather than adding a separate CDF or histogram, Table 6 reports a five-point summary for every configuration (sample count N, RMS, p95, p99, and Max) in place of a plotted distribution. The scapular acromion row is the clearest case. With $N = 164,820$ evaluation samples, the adaptive mean is 0.0006 mm, p95 = 0.0004 mm, and p99 = 0.0033 mm, but the maximum residual is 0.247 mm, so RMS rises to 0.0083 mm. Thus, the bulk of the adaptive surface is more accurate than the RMS alone suggests; the RMS is dominated by a small high-error tail. The drill site for this condition is on the acromion ridge itself, and the high-error samples are clustered there: the top 0.1% of residuals (error from about 0.16 mm to the 0.247 mm maximum) lie within one brush radius of the drill UV rather than being scattered over the cut window. The maxima of the adaptive and uniform-4 configurations are comparable (0.247 mm adaptive, 0.259 mm uniform 4), since both are set by the same high-curvature ridge geometry, but the distributions differ sharply below that point: adaptive p99 is 0.0033 mm against 0.0806 mm for uniform 4, a factor of twenty-four. The advantage of adaptive tessellation is therefore in the body of the distribution rather than in the extreme tail. Uniform 50 keeps a much smaller tail (p95 = 0.0001 mm, p99 = 0.0008 mm, Max = 0.0171 mm). The maximum of 0.247 mm is roughly 12% of the 2 mm cut depth and about thirty times the adaptive RMS.

Computed GPU texture-memory footprint. Table 5 reports resident and Pass 2 transient memory at the production 4096^2 resolution. Because both quantities are determined by texture resolution and channel format (four-channel half-float), the same accounting extends immediately to the six resolutions used in Table 2 (Table 7). Resident cost covers the displacement map, the tessellation-amount map, and the associated mip storage under the same convention as Table 5; the Pass 2 transient is one full-resolution temporary copy without a mip chain. Values are computed, not read back from the driver.

Multi-structure scaling. Each simultaneously drillable anatomical asset carries its own displacement and tessellation-amount maps, so resident texture memory scales approximately linearly with the number of such assets at fixed resolution. The Pass 2 temporary is transient in the accounting sense: a single full-resolution temporary is drawn from the pool per contact frame regardless of how many assets are resident, because only one asset is in contact under the single-contact protocol of Section 3.7, so that temporary cost does not grow with the number of drillable structures. On the 32 GB workstation GPU used here, three concurrent 4096^2 drillable assets would therefore require roughly 1.1 GB of resident texture memory (3×358 MB) plus one 134 MB transient during an update, which remains a small fraction of device memory; the present study does not measure driver-reported VRAM under multi-asset scenes.

Table 7. Computed GPU texture-memory footprint versus displacement-map resolution (four-channel half-float), under the same accounting as Table 5. Resident storage covers the displacement map, tessellation-amount map, and mip chains; Pass 2 transient is one full-resolution temporary copy without a mip chain. Values are derived from resolution and format, not from driver readback.

Resolution	Resident (Both Maps + Mips)	Pass 2 Transient
128 ²	0.35 MB	0.13 MB
256 ²	1.40 MB	0.52 MB
512 ²	5.59 MB	2.10 MB
1024 ²	22.4 MB	8.39 MB
2048 ²	89.5 MB	33.6 MB
4096 ²	358 MB (≈ 0.36 GB)	134 MB (≈ 0.13 GB)

4.7. Crack-Free Rule: Empirical Verification and Topology Audit

Section 3.10 argues that Equation (9) assign the same factor to both sides of a shared edge, because every input to that factor is a function of the two shared endpoint UVs alone. We tested this argument directly on the production humerus after a 6 mm $\times$ 2.0 mm cortical bed had been cut. For every indexed shared edge, the factor was computed independently from each incident triangle, and the two values were compared. Under Equation (9), all 4271 shared edges agree exactly, with $|\Delta f| = 0$ everywhere. As a control, the same test was run on a variant that selects the mipmap level from the maximum pairwise UV span of the whole triangle, and which therefore depends on the third, non-shared vertex: 36 of the same 4271 edges disagree, and the worst disagreement reaches $|\Delta f| = 31$, meaning one triangle requested 31 more subdivisions of the shared edge than the other, on a factor range of 1 to 64. These measurements verify equality of the input edge factors. Under the fractional_odd partitioning mode declared explicitly on the Unity HDRP hull shader, identical floating-point factors produce identical parametric subdivision points along a shared edge. Given consistent endpoint attributes and common tessellation-amount and displacement texture state, the domain shader then evaluates the same displaced position at each of those points on both sides. We report factor agreement rather than a GPU readback of tessellated vertex buffers, and we treat watertight final geometry as following from factor equality together with that specified hardware-tessellation configuration, not as an independently instrumented mesh-buffer proof across APIs or GPU architectures.

The control establishes that the test can detect failure, and it identifies the specific dependency that must be avoided. Any selection rule that reads the third vertex can assign different factors to the two sides of an edge, which is the concrete failure mode of a triangle scoped level of detail rule and the reason the selection was moved from the triangle to the edge. Because the test enumerates every shared edge of an irregular production mesh at a fixed camera, and because the factor is by construction independent of camera position and of the non-shared vertices, the result also covers irregular triangulations and varying viewing distance without further measurement. The corresponding visual result is Figure 16c, in which the rim of the cut closes with no cracks or T junctions.

The same asset was audited for the topology classes of Section 3.14. R_Humerus contains 4271 shared indexed edges, 146 index boundary edges arising from atlas chart splits, 87 duplicated seam split vertices, no non-manifold indexed edges, and no true geometric boundary loops: the cortex is closed, and every apparent boundary in the index buffer is a UV seam rather than a surgical opening. Each class is handled as described in Section 3.14. The crack free rule acts on true shared indices only, so the two copies of a duplicated seam vertex are never coupled, and both remain undisplaced while the brush stays inside its island, which is why the 72 geometric edges lying on seams remain in agreement at baseline. Boundary edges have no neighbor to match, and their factor is still computed from the two endpoints alone. Non-manifold regions are excluded by asset preparation rather than by the method, since the hull shader operates per triangle and

never welds, splits, or re-indexes geometry, leaving topology and genus invariant. A brush that straddles a chart border tears the imprint, which is a limitation of UV space brushing rather than a tessellation crack, and it is discussed with the other limitations in Section 5. The empirical agreement test was repeated across multiple drilling states on the same asset (intact cortex; light, nominal, and heavy stroke trains at the greater tuberosity; and a second medial-row UV site). In every state all 4271 indexed shared edges agreed with $|\Delta f| = 0$. Factor-boundary cases are covered by construction: because both incident triangles receive the identical unquantized float, they cannot straddle a fractional_{odd} rounding boundary relative to each other. The same instrumentation reports 36 disagreements under the whole-triangle variant of Section 4.7, confirming that the test discriminates.

Scope of the crack-free claim. Identical tessellation factors on both sides of an indexed shared edge are a necessary condition for preventing T-junctions under hardware tessellation. Section 4.7 verifies that condition exhaustively for the production R_Humerus asset under the drilling states above. That evidence supports the edge-symmetric rule on shared indices; it is not offered as a general proof of watertight geometry for every asset, frame, graphics API, or GPU architecture. The guarantee is conditional on (i) indexed shared edges (not duplicated seam copies), (ii) consistent endpoint UV and tangent-frame attributes on those shared indices, (iii) a common tessellation-amount and displacement texture state, and (iv) the fractional_{odd} hardware-tessellation configuration used here. Geometrically coincident but index-duplicated UV-seam edges are a separate class: the rule does not couple the two copies, and continuity there is maintained by keeping the brush inside a single UV island (Section 3.14), not by cross-seam factor agreement.

4.8. Component Ablation

Table 8 isolates the contribution of each component of the pipeline. Two components have been measured independently, and their removal is reported quantitatively. The remaining three fail structurally rather than by degree: removing them does not degrade a metric, it removes a capability, and we have not constructed artificial metrics to give those rows a number. In each case the table states what was substituted, what changes as a result, and where in the paper the evidence appears.

Table 8. Component ablation. Each row states which component is removed, what replaces it, what changes as a result, and where the evidence is. Quantitative rows report measured differences; structural rows report a capability that is lost rather than a metric that degrades.

Component Removed	Substituted Configuration	Observable Effect	Evidence
Patterned drawing (Pass 2)	Uniform radial brush	Adjacent vertices of one triangle receive very different displacements, producing an elongated triangle with a single per triangle normal and a visible shading discontinuity along the cavity rim.	Figures 6 and 7a (structural)
Edge symmetric UV sampling, Equation (9)	Mipmap level from the maximum pairwise UV span of the whole triangle	36 of 4271 shared edges disagree on the tessellation factor, worst disagreement $ \Delta f = 31$; cracks open along those edges.	Section 4.7 (quantitative)
Sobel curvature estimator (Pass 3)	Circular curvature estimator	Depth output MSE rises from 0.004 to 0.080 (20×) and depth SSIM falls from 0.99992 to 0.99944; color output similarity improves, which is why the estimator choice is argued on depth rather than color.	Table 1 (quantitative)
Mipmap based LOD selection (Pass 4)	Always sample level 0	The factor is no longer bounded by the UV span of the edge, so large distant triangles request maximum subdivision, and the frame rate advantage of Table 4 disappears; the crack free property itself is unaffected.	Section 3.10, Figure 11 (structural)
Fragment normal correction (Pass 6)	Per vertex base mesh normals	The cavity shades as though undisplaced: the drilled geometry is present but not legible, because the normals still describe the original surface.	Section 3.12 (structural)

5. Discussion

Every measurement in this paper comes from one desktop configuration (Intel Core Ultra 9, NVIDIA GeForce RTX 5090, 32 GB VRAM, 128 GB RAM, Windows 11, Unity 2019.3.5f1 HDRP). The pipeline's cost is dominated by texture resolution dependent blit work in Passes 2 and 3 together with the mipmap chain, and by the tessellated draw in Passes 4 to 6, which scale with fill rate and tessellation throughput respectively, so a lower tier discrete GPU would be expected to increase the reported times roughly in proportion to those units and an integrated GPU considerably more. We have not measured this. The 0.397 ms end-to-end figure should therefore be read as a result for the stated RTX 5090 configuration rather than as a hardware-independent property, and the characterization across integrated graphics, lower-tier discrete GPUs, and standalone headsets, with variance reported per platform, is deferred to the follow-up study that also instruments the haptic loop.

Relative to the volumetric AMR pipeline we have reported previously [22], the shader-based method trades true volumetric authority for a large performance headroom and stable graphics-frame timing during drilling. Specifically, the method described in this paper models only the visual displacement of the drilled bone surface. It does not represent volumetric bone removal, spatially varying cortical or trabecular structure, spatially varying bone density, or the material-dependent drilling forces that those internal structures would produce. Tasks that require this internal authority, including deep cortical-breach detection during pilot-hole placement, density-dependent haptic feedback along a drill trajectory, and planning-grade volumetric material removal for surgical rehearsal, remain in the domain of the volumetric AMR module or an equivalent physics-based approach; the capability boundary between the two representations is set out in Table 7 and discussed in Section 4.5. The two representations are complementary, and both modules share the same ViRCAST haptic loop [4,25]; the selection rule between them is given below.

The visual assessment in Section 4.4 is based on pixel-level similarity (SSIM) and depth-MSE against a high-fidelity uniform-tessellation reference. These are objective similarity and geometric-error measures under a fixed viewing condition; they do not by themselves establish perceptual indistinguishability. A supervised perceptual study, using a defined viewing protocol and either a general-observer forced-choice paradigm or a trained-observer expert evaluation, would be needed to determine whether the residual differences meet a specific perceptual threshold in a clinically relevant task. Such a study is outside the scope of the present paper and is identified as future work.

Several avenues remain open. First, the patterned-drawing brush is currently authored from a cylindrical texture tuned to suture-anchor geometry; a future version could load instrument-specific brushes from the surgical toolset and switch between them on the fly. Second, the Sobel filter operates on a single scalar derived from the vector-displacement magnitude; a vector-aware edge detector could further reduce tessellation in regions where only one displacement axis is active. Third, integrating this pipeline with the cloud-streaming prototype reported in [24] would allow training on lightweight clients without local GPUs. Finally, per-vertex bone density sampled from CT imagery could drive a patient-specific brush amplitude, opening the door to pre-operative rehearsal.

The performance results reported in Section 4.2 characterize the rendering pipeline only. In ViRCAST, the haptic loop runs on a dedicated thread with its own contact-detection, contact-resolution, and force-computation stages that are separate from and asynchronous to the rendering pipeline described here. Determining whether the ViRCAST haptic loop sustains 700 Hz under drilling therefore requires separate instrumentation of the haptic subsystem. Specifically, this would require measurement of the haptic-thread cycle time and cycle-time jitter, of contact and collision-computation cost under drilling load, of the

synchronization overhead between the haptic and graphics threads (both of which read from the drilled bone's state), and of the end-to-end visual-haptic latency perceived by the user. Such instrumentation is not attempted in the present study and is left as future work. Consequently, none of the performance figures reported here are interpreted as haptic-loop timings; the fact that the rendering update at 4K (0.397 ms mean, 0.433 ms p99) is well below the 1.4 ms budget of a 700 Hz haptic loop is a necessary but not sufficient condition for haptic-loop stability. The claim supported by the measurements is narrower: the rendering pipeline does not, by itself, consume enough time to starve a properly scheduled haptic thread of dispatch opportunities on the shared hardware, and it comfortably meets the 16.7 ms budget of a 60 Hz display with two orders of magnitude of headroom.

Table 9 makes the trade-off operational as a qualitative selection rule rather than as a controlled benchmark under comparable conditions. The shader pipeline is the right choice when the cut is shallow relative to local cortical thickness, when the wall gradient stays bounded (in our assets $g \leq 2.2$, a triangle stretch of at most $2.4\times$, Figure 19), when the footprint lies inside a single UV island, and when the binding constraint is update rate. The volumetric AMR module is the right choice when the operation changes topology, when the cut must expose or respond to internal structure, when the target is a thin shell that the cut may perforate, or when the haptic layer needs a volumetric collision proxy rather than a surface. Applied to ViRCAST, this divides cleanly: tendon-footprint decortication, bursectomy surface preparation, and acromioplasty are served by the shader pipeline; whereas, anchor-tunnel drilling that penetrates substantial bone stock is served by AMR. No matched-condition runtime or geometric-error contest between the two modules is reported in this paper.

Arthroscopic Rotator Cuff repair involves two distinct bone operations. Surface preparation of the tendon footprint by burring or shaving removes cortex to a depth of roughly 1 to 2 mm [61], and is the operation this method targets. Socket drilling for suture anchor placement is a separate step, with all suture anchor insertion drills measuring 20 to 24 mm in length [62], which lies well outside the wall gradient regime characterized below and belongs to the volumetric module.

Three limitations of tangent-space displacement on a fixed base mesh are quantitative rather than conceptual, and adaptive tessellation does not remove them. First, displacing the undisplaced base surface stretches wall triangles by approximately $\sqrt{1 + g^2}$ on a wall of slope g . Figure 19 shows measured 95th-percentile stretch tracking that prediction on the humerus. The 6 mm $\times$ 2 mm beds used throughout this paper sit at $g \leq 2.2$, a stretch of at most $2.4\times$, but a deep cylindrical anchor socket would leave that regime. Second, the iterative update of Equation (8) renormalizes the packed displacement at unit length, so a long train of overlapping strokes progressively flattens a rounded tool profile into a flat-floored cylinder; the operating plunges in our trajectories saturate at approximately eight strokes with a peak packed magnitude of 0.80, below the flattening threshold. Third, displacement cannot change topology. Figure 20 shows the consequence on the glenoid fossa, a shell of approximately 0.19 mm thickness: a requested depth greater than the shell thickness perforates and self-intersects rather than opening a tunnel. New holes require the volumetric module.

Texture displacement carries no representation of shell thickness. Once the requested depth exceeds the bone wall, Pass 2 continues to accumulate displacement, Pass 3 continues to build a Sobel tessellation map, and the hull and domain shaders continue to subdivide and displace vertices. Because topology is fixed, the front surface is driven through the back rather than opening a hole, and adaptive tessellation compounds the artifact by adding polygons to the over displaced, self-intersecting geometry: greater depth yields more triangles rather than a clean tunnel.

Table 9. Capability comparison between texture-based visual displacement and volumetric AMR/Marching Cubes approaches, used as a selection rule between the two ViRCAST bone-deformation modules. This comparison is qualitative: it is not a controlled experimental benchmark under matched assets, cut depths, hardware, or timing protocols. Measured results in this paper are confined to the shader pipeline (Sections 4.1–4.8); entries for volumetric AMR summarize representational capability rather than head-to-head runtime or error measurements. The two representations are complementary rather than competing.

Capability or Operation	Texture Displacement (This Method)	Volumetric AMR/Marching Cubes
Graphics update rate	Sub-millisecond at 4K (0.397 ms end-to-end, Table 3)	Slower; scales with grid resolution and re-extraction cost
Shallow cortical beds, decortication, acromioplasty	Well suited; wall gradient bounded, triangles near-isotropic	Capable but costlier for the same visual result
Deep cylindrical anchor sockets	Wall triangles stretch as $\sqrt{1 + g^2}$; not isotropic (Figure 19)	Natural representation
Through-holes and genus change	Not possible; the surface stays a fixed-topology disk	Native
Thin shells (glenoid, thin cortex)	Over-deep cuts perforate or self-intersect (Figure 21)	Represents thickness explicitly
UV atlas seams	Brush must stay inside one island (Section 3.14)	Independent of UV parameterization
Long stroke trains	Pass 2 renormalizes at unit length; the tool profile flattens	Boolean-stable under repeated removal
Internal structure and material-dependent forces	Not represented; a separate volumetric or SDF layer is required	Can share the volume with the haptic layer
Role in ViRCAST	Visual surface deformation at graphics and haptic rates	Complement for deep and topology-changing cuts

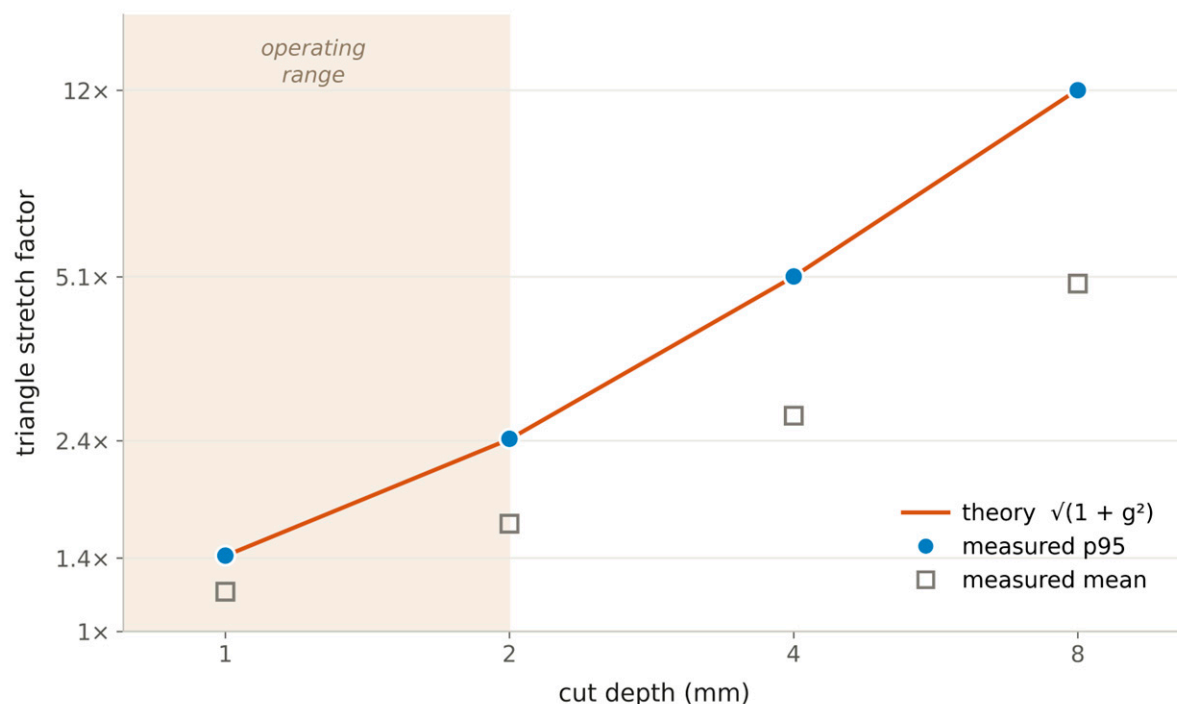


Figure 19. Triangle stretch against cut depth on R_Humerus. Displacing the undeformed base surface stretches wall triangles by approximately $\sqrt{1 + g^2}$ on a wall of slope g , shown as the orange line evaluated at the measured slopes $g = 1.0, 2.2, 5.0$ and 12.0 . Filled circles are the measured 95th percentile stretch, which agrees with the prediction to within 0.5 percent at every depth; open squares are the measured mean, which is lower because most wall triangles sit below the steepest gradient. Measured maxima are within one percent of p95 throughout, reaching $1.419\times$, $2.431\times$, $5.135\times$ and $12.122\times$. Both axes are logarithmic. The shaded band marks the operating range used in this paper, a 6 mm burr cutting a 2.0 mm bone bed, where $g \leq 2.2$ and stretch $\leq 2.4\times$. Figure 21 shows the corresponding wireframe views.

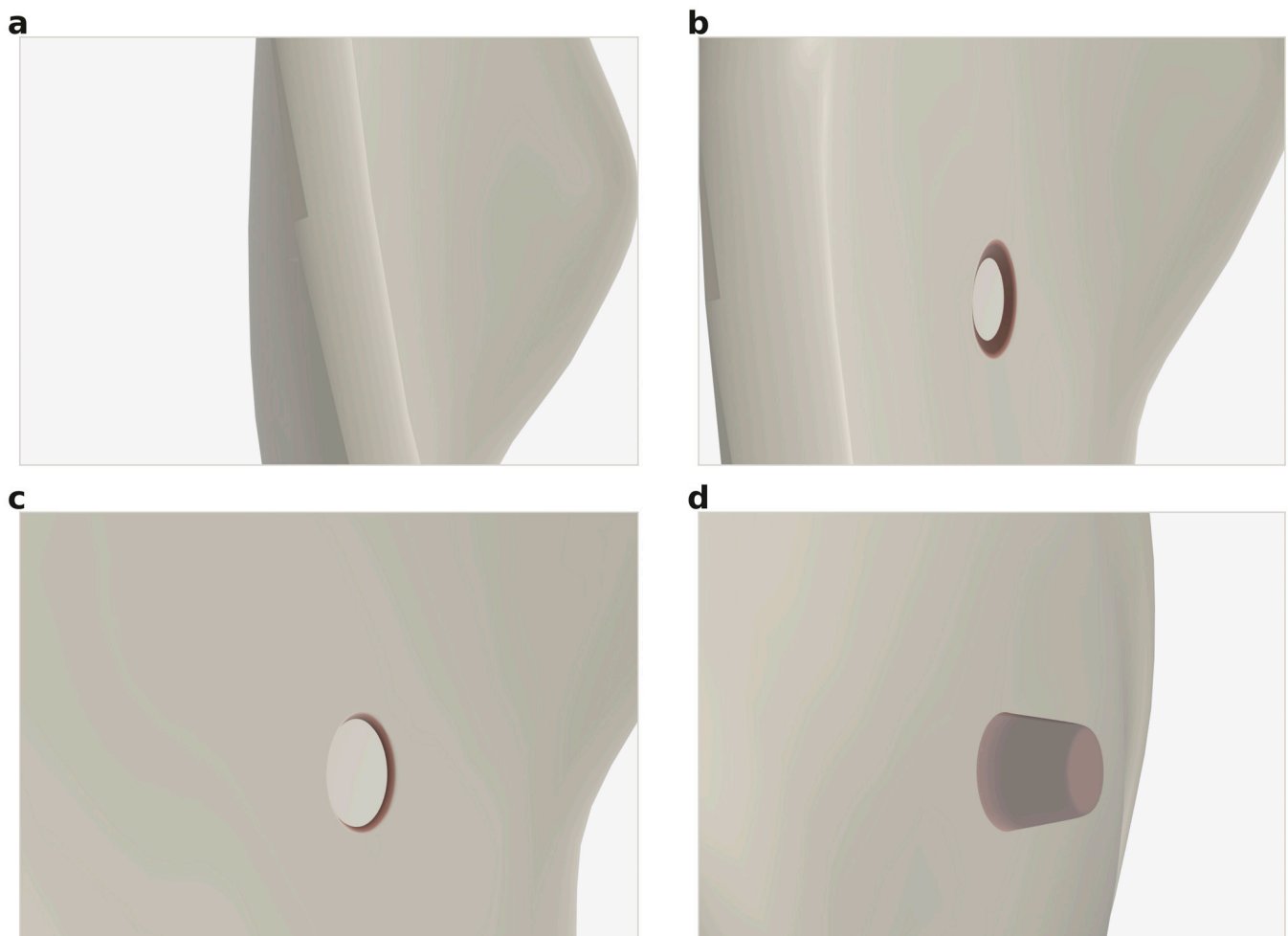


Figure 20. Fixed topology limit of texture displacement, on the ViRCAST Glenoid_Fossa asset with a median face to back thickness of approximately 0.19 mm. The pipeline never changes mesh connectivity, so once the requested depth exceeds the shell the front surface is forced through the back rather than a tunnel being opened, because displacement cannot change genus. (a) A shallow dent, 0.03 mm requested, well under shell thickness and visually valid. (b) At the shell limit, 0.20 mm requested, approximately equal to thickness. (c) Perforation, 0.58 mm requested, viewed from the front: the surface perforates and self-intersects, and the light disk is the back surface showing through rather than a drilled tunnel. (d) Severe overcut, 2.31 mm requested, viewed from the back: displacement and adaptive tessellation continue past thickness, extruding a dense nonphysical membrane. Operations of this kind belong to the volumetric module (Table 9).

The pipeline as described assumes a single drill contact per frame (Section 3.7). Extending it to concurrent instruments is straightforward in principle and untested in practice. Pass 2 would iterate the blend of Equation (8) once per contact into the same displacement texture, so the cost of Passes 2 and 3 would be expected to grow linearly with the number of contacts while Passes 4–6, which read the finished texture, would not grow at all. Overlapping brush footprints would compose through the same α -weighted interpolation, which is order-dependent and would require a defined resolution order to remain frame-rate independent. Neither the cost scaling nor the overlap behavior has been measured here, and both are prerequisites for any claim about multi-instrument scenarios.

The method depends on hardware tessellation, which is available under Direct3D 11 (Shader Model 5.0) and later, OpenGL 4.0 and later, Vulkan, and Metal, but not on platforms that expose no tessellation stage, including WebGL 2.0 and several standalone mobile VR runtimes. This dependency is inherited rather than chosen: the pipeline was developed as a

module of the existing ViRCAST simulator, whose rendering, haptic, and scene subsystems already target Unity with a Direct3D 11 backend, and a bone deformation module must remain interoperable with that installed base rather than require the surrounding system to be rebuilt.

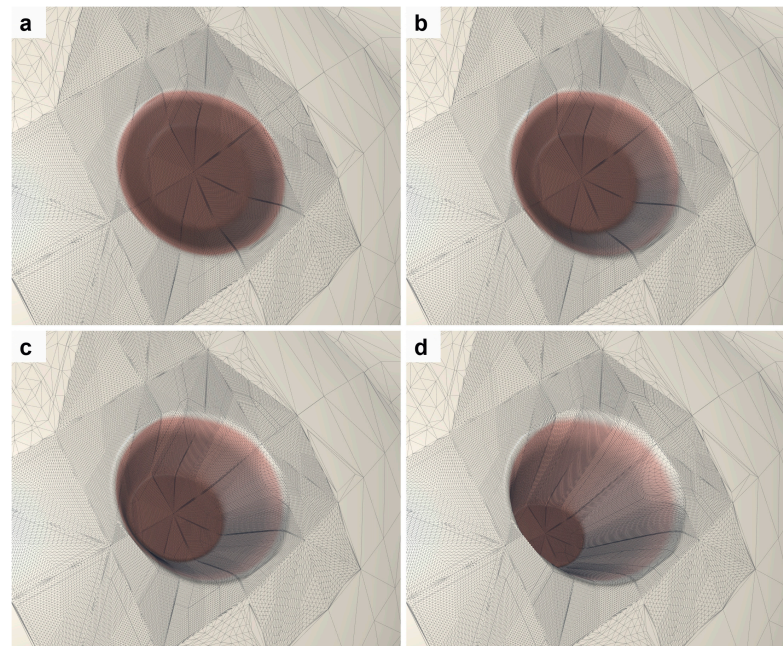


Figure 21. Limitation: deep cuts stretch wall triangles. Wireframe views of the same cavity on R_Humerus at increasing cut depth, read row by row. Because displacement is parameterized on the undeformed base surface, a wall of slope g stretches faces by approximately $\sqrt{1 + g^2}$. Top row: (a) 1 mm deep, $g = 1.0$, theory $1.4\times$, measured p95 $1.4\times$; (b) 2 mm deep, $g = 2.2$, theory $2.4\times$, measured p95 $2.4\times$, the operating configuration used throughout this paper. Bottom row: (c) 4 mm deep, $g = 5.0$, theory $5.1\times$, measured p95 $5.1\times$; (d) 8 mm deep, $g = 12.0$, theory $12.0\times$, measured p95 $12.0\times$, maximum $12.1\times$. Wall faces remain near isotropic through the top row and are visibly drawn out along the cut direction in the bottom row. Figure 19 plots the same measurements against the prediction. Deep cylindrical sockets are deferred to the volumetric AMR module.

Passes 2 and 3 and the mipmap chain are ordinary full screen shader work and port unchanged; Pass 1 is CPU side and is unaffected. Passes 4 to 6 require a fallback. The most direct is to pre subdivide the drillable region once at load time to the maximum factor the method would have requested and to apply Equations (6) and (7) in the vertex shader; this preserves the visual result at a fixed and higher memory and vertex cost and gives up exactly the adaptivity that Table 4 measures. Where compute shaders are available but tessellation is not, a computation of the Tessellator in the manner of [42] is the alternative. Neither fallback has been implemented or measured.

The evaluation in this paper is entirely computational. No orthopedic surgeon has rated the visual realism of the simulated drilling, and no comparison against intra-operative arthroscopic video has been performed. High rendering fidelity does not by itself establish skill transfer, so the claims made here are limited to rendering behavior and to geometric agreement with the displacement field. An expert-rated realism study and a comparison against recorded arthroscopic footage are the natural next steps; both require surgeon-participant recruitment and institutional review board approval that fall outside the scope of a computational rendering study, and both are planned alongside the haptic-loop characterization.

Surgical technique training is fundamental for orthopedic surgeons to ensure they reach a minimum level of proficiency before independent practice. Specifically, humeral

head preparation for suture anchors in rotator cuff surgery is necessary for both fixation strength and biological healing. This is achieved by placing pilot holes perpendicular to the native footprint to maximize cortical purchase without allowing intra-articular breach. The interactive virtual reality arthroscopy platforms increase early skill acquisition by exposing participants to repeated feedback-rich practice on core principles such as triangulation, camera navigation, and anchor drilling.

While construct validity of arthroscopic simulators is well established, recent studies [63–65] have demonstrated transfer validity to cadaveric and simulated surgical environments; however, evidence demonstrating consistent transfer to live operating-room performance remains limited. A future study is needed to determine whether skills acquired through VR training translate into improved surgical outcomes and repair success in the operating room. This study may become increasingly important as cadaver availability for surgical training declines, potentially increasing the reliance on VR simulation [66]. Lohre et al. [19] also reported that senior residents in immersive VR training experience a superior learning environment compared to traditional instructional video in reverse shoulder arthroplasty, with a reported Transfer Effectiveness Ratio (TER) of 0.79 (47–48 min of OR-equivalent learning per 60 min of VR). Overall, the current literature [67,68] reports that VR training can improve surgical skills acquisition compared to traditional training methods. For ViRCAST to deliver this level of training benefit, it must run smoothly, especially during the drilling phase, which is the most technically demanding aspect. Frame drops or visual interruptions during the drilling step would decrease the reliability of the user's performance and diminish the value of the simulation experience.

6. Conclusions

We have presented a fully GPU-resident shader-based bone-drilling pipeline for ViRCAST that modifies dynamic textures in place of the underlying mesh. The six passes, comprising drawing-vector computation, displacement-map update via patterned drawing, Sobel-driven tessellation-map generation, mipmap-based tessellation sampling, vertex displacement, and fragment normal correction, together execute in 0.397 ms per frame on average at 4K texture resolution (p95 = 0.424 ms, p99 = 0.433 ms, N = 1000) on a desktop workstation GPU, and yield 120 fps in representative multi-object scenes. Both the texture-update path (0.292 ms mean) and the full end-to-end update (0.397 ms mean) remain far below the 16.7 ms graphics-frame budget of a 60 Hz display across every resolution tested. All performance claims in this paper are limited to rendering. The fact that the rendering update remains far below the 1.4 ms budget of a 700 Hz haptic loop is a necessary but not sufficient condition for haptic-loop stability and should not be read as a claim that the ViRCAST haptic loop itself achieves 700 Hz. Characterization of the haptic loop is deferred to a follow-up study (see Section 5). All performance measurements in this study were obtained on a single high-end desktop GPU (NVIDIA GeForce RTX 5090); scalability to lower-tier and typical VR hardware is discussed under Hardware scope in Section 5 and remains unmeasured here.

A patterned-drawing scheme suppresses stretched-triangle artifacts at drill-site edges, and an edge-symmetric UV sampling rule assigns identical tessellation factors to both sides of every shared edge: on the production humeral asset all 4271 shared edges agree exactly; whereas, a whole-triangle level-of-detail rule leaves 36 edges disagreeing by up to 31. On production anatomical meshes the adaptive result stays within 0.002–0.008 mm RMS of the continuous displacement field, at 67–304× fewer triangles than a quality-matched uniform reference (depth SSIM ≥ 0.9998). By sustaining high graphics rates during active drilling, the method supports continuous visual feedback associated with transferable surgical skill

acquisition, and it complements the volumetric AMR module in the bone-deformation capability of ViRCAST.

This paper addresses only the visual half of bone drilling. The pipeline described here does not simulate volumetric bone removal, represent internal cortical or trabecular structure, represent spatially varying bone density, or produce material-dependent drilling forces. Tasks that require these physical properties, including deep cortical-breach modeling, density-dependent haptic feedback, and planning-grade volumetric material removal, require a complementary volumetric or physics-based drilling module such as the AMR-based approach of prior work [22]; Table 9 sets out which representation is appropriate for which class of operation. The shader pipeline described here should therefore be understood as providing real-time visual surface deformation and rendering for the drilled cavity, not as a complete physical bone-drilling model.

Author Contributions: S.D.K. contributed to conceptualization, investigation, supervision, software (revision-cycle implementation updates), visualization (additional figures), and writing—review and editing. F.D. contributed to software, validation, and data curation. T.H. contributed to methodology, formal analysis, and writing—review and editing. H.R. and C.M. contributed to clinical validation and writing—review and editing. All authors have read and agreed to the published version of the manuscript.

Funding: This research is partially supported by the National Institutes of Health award NIAMS R44AR075481.

Institutional Review Board Statement: Not applicable. This study is a computational graphics and rendering evaluation that did not involve human participants or animals.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest. Tansel Halic is currently employed by Intuitive Surgical. However, all work contributed by Halic and presented in this manuscript was conducted prior to his employment with Intuitive Surgical. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as potential conflicts of interest.

References

1. Colvin, A.C.; Egorova, N.; Harrison, A.K.; Moskowitz, A.; Flatow, E.L. National trends in rotator cuff repair. *J. Bone Jt. Surg. Am.* **2012**, *94*, 227–233. [\[CrossRef\]](#)
2. Stew, B.; Ooi, E. The role of simulation in endoscopic sinus surgery training. In *Paranasal Sinuses*; IntechOpen: London, UK, 2017; p. 93.
3. Iannotti, J.P. Full-thickness rotator cuff tears: Factors affecting surgical outcome. *J. Am. Acad. Orthop. Surg.* **1994**, *2*, 87–95. [\[CrossRef\]](#)
4. Farmer, J.; Tunc, M.; Ahmadi, D.; Demirel, D.; Halic, T.; Arikatla, S.; Kockara, S.; Ahmadi, S. Virtual Rotator Cuff Arthroscopic Skill Trainer: Results and analysis of a preliminary subject study. In Proceedings of the 4th International Conference on Information System and Data Mining (ICISDM 2020), Hilo, HI, USA, 15–17 May 2020.
5. Longo, U.G.; Carnevale, A.; Piergentili, I.; Berton, A.; Candela, V.; Schena, E.; Denaro, V. Retear rates after rotator cuff surgery: A systematic review and meta-analysis. *BMC Musculoskelet. Disord.* **2021**, *22*, 749. [\[CrossRef\]](#)
6. Wang, E.E.; Vozenilek, J.A.; Flaherty, J.; Kharasch, M.; Aitchison, P.; Berg, A. An innovative and inexpensive model for teaching cricothyrotomy. *Simul. Healthc.* **2007**, *2*, 25–29. [\[CrossRef\]](#)
7. Pettineo, C.M.; Vozenilek, J.A.; Wang, E.; Flaherty, J.; Kharasch, M.; Aitchison, P. Simulated emergency department procedures with minimal monetary investment: Cricothyrotomy simulator. *Simul. Healthc.* **2009**, *4*, 60–64. [\[CrossRef\]](#)
8. Cho, J.; Kang, G.H.; Kim, E.C.; Oh, Y.M.; Choi, H.J.; Im, T.H.; Yang, J.H.; Cho, Y.S.; Chung, H.S. Comparison of manikin versus porcine models in cricothyrotomy procedure training. *Emerg. Med. J.* **2008**, *25*, 732–734. [\[CrossRef\]](#)

9. Aggarwal, R.; Ward, J.; Balasundaram, I.; Sains, P.; Athanasiou, T.; Darzi, A. Proving the effectiveness of virtual reality simulation for training in laparoscopic surgery. *Ann. Surg.* **2007**, *246*, 771–779. [\[CrossRef\]](#)
10. Jamal, M.H.; Rousseau, M.C.; Hanna, W.C.; Doi, S.A.; Meterissian, S.; Snell, L. Effect of the ACGME duty hours restrictions on surgical residents and faculty: A systematic review. *Acad. Med.* **2011**, *86*, 34–42. [\[CrossRef\]](#)
11. Angelo, R.L.; Ryu, R.K.; Pedowitz, R.A.; Beach, W.; Burns, J.; Dodds, J.; Field, L.; Getelman, M.; Hobgood, R.; McIntyre, L.; et al. A proficiency-based progression training curriculum coupled with a model simulator results in the acquisition of a superior arthroscopic Bankart skill set. *Arthroscopy* **2015**, *31*, 1854–1871. [\[CrossRef\]](#)
12. Albani, J.M.; Lee, D.I. Virtual reality-assisted robotic surgery simulation. *J. Endourol.* **2007**, *21*, 285–287. [\[CrossRef\]](#)
13. Wang, P.; Becker, A.; Jones, I.; Glover, A.; Benford, S.; Greenhalgh, C.; Vloeberghs, M. A virtual reality surgery simulation of cutting and retraction in neurosurgery with force-feedback. *Comput. Methods Programs Biomed.* **2006**, *84*, 11–18. [\[CrossRef\]](#)
14. Parham, G.; Bing, E.G.; Cuevas, A.; Fisher, B.; Skinner, J.; Mwanahamuntu, M.; Sullivan, R. Creating a low-cost virtual reality surgical simulation to increase surgical oncology capacity and capability. *Ecancermedicalscience* **2019**, *13*, 910. [\[CrossRef\]](#)
15. Dakson, A.; Hong, M.; Clarke, D.B. Virtual reality surgical simulation: Implications for resection of intracranial gliomas. *Prog. Neurol. Surg.* **2017**, *30*, 106–116. [\[CrossRef\]](#)
16. Frank, R.M.; Erickson, B.; Frank, J.M.; Bush-Joseph, C.A.; Bach, B.R.; Cole, B.J.; Romeo, A.A.; Provencher, M.T.; Verma, N.N. Utility of modern arthroscopic simulator training models. *Arthroscopy* **2014**, *30*, 121–133. [\[CrossRef\]](#)
17. Modi, C.S.; Morris, G.; Mukherjee, R. Computer-simulation training for knee and shoulder arthroscopic surgery. *Arthroscopy* **2010**, *26*, 832–840. [\[CrossRef\]](#)
18. Hodgins, J.L.; Veillette, C. Arthroscopic proficiency: Methods in evaluating competency. *BMC Med. Educ.* **2013**, *13*, 61. [\[CrossRef\]](#)
19. Lohre, R.; Bois, A.J.; Pollock, J.W.; Lapner, P.; McIlquham, K.; Athwal, G.S.; Goel, D.P. Effectiveness of immersive virtual reality on orthopedic surgical skills and knowledge acquisition among senior surgical residents. *JAMA Netw. Open* **2020**, *3*, e2031217. [\[CrossRef\]](#)
20. Waterman, B.R.; Martin, K.D.; Cameron, K.L.; Owens, B.D.; Belmont, P.J. Simulation training improves surgical proficiency and safety during diagnostic shoulder arthroscopy performed by residents. *Orthopedics* **2016**, *39*, 479–485. [\[CrossRef\]](#)
21. Gomoll, A.H.; O'Toole, R.V.; Czarnecki, J.; Warner, J.J.P. Surgical experience correlates with performance on a virtual reality simulator for shoulder arthroscopy. *Am. J. Sports Med.* **2007**, *35*, 883–888. [\[CrossRef\]](#)
22. Dinc, F.; Oumimoun, K.; Kwabla, W.; Kockara, S.; Halic, T.; Arikatla, S.; Ahmadi, S. Towards real-time bone drilling simulation for anchor placement in VR-based arthroscopic rotator cuff surgery simulation. *AMIA Summits Transl. Sci. Proc.* **2022**, *2022*, 178–185.
23. Dendukuri, A.; Tunc, M.; Demirel, D.; Kockara, S.; Halic, T. Unified framework for real-time fluid simulation in Virtual Rotator Cuff Arthroscopic Skill Trainer (ViRCast). In Proceedings of the 2023 IEEE 23rd International Conference on Bioinformatics and Bioengineering (BIBE), Dayton, OH, USA, 4–6 December 2023.
24. Kwabla, W.; Dinc, F.; Oumimoun, K.; Kockara, S.; Halic, T.; Demirel, D.; Arikatla, S.; Ahmadi, S. Evaluation of WebRTC in the cloud for surgical simulations: A case study on Virtual Rotator Cuff Arthroscopic Skill Trainer (ViRCast). In *Learning and Collaboration Technologies (HCII 2023); Lecture Notes in Computer Science*; Springer: Cham, Switzerland, 2023; Volume 14041, pp. 127–143.
25. Delorme, S.; Laroche, D.; DiRaddo, R.; Del Maestro, R.F. NeuroTouch: A physics-based virtual simulator for cranial microneurosurgery training. *Neurosurgery* **2012**, *71*, 32–42.
26. Weber, G.H.; Kreylos, O.; Ligocki, T.J.; Shalf, J.M.; Hagen, H.; Hamann, B.; Joy, K.I. Extraction of crack-free isosurfaces from Adaptive Mesh Refinement data. In Proceedings of the Joint Eurographics/IEEE TCVG Symposium on Visualization (VisSym 2001), Ascona, Switzerland, 28–30 May 2001.
27. Tunc, M.; Demirel, D.; Kockara, S.; Halic, T. Crack-free multi-resolution dynamic volume refinement for physics-based bone drilling and shaving in arthroscopic procedures. In Proceedings of the 2025 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW), Saint-Malo, France, 8–12 March 2025.
28. Lorensen, W.E.; Cline, H.E. Marching cubes: A high resolution 3D surface construction algorithm. *SIGGRAPH Comput. Graph.* **1987**, *21*, 163–169. [\[CrossRef\]](#)
29. van Gelder, A.; Wilhelms, J. Topological considerations in isosurface generation. *ACM Trans. Graph.* **1994**, *13*, 337–375. [\[CrossRef\]](#)
30. Berger, M.J.; Colella, P. Local adaptive mesh refinement for shock hydrodynamics. *J. Comput. Phys.* **1989**, *82*, 64–84. [\[CrossRef\]](#)
31. Berger, M.J.; Olinger, J. Adaptive mesh refinement for hyperbolic partial differential equations. *J. Comput. Phys.* **1984**, *53*, 484–512. [\[CrossRef\]](#)
32. Wald, I. A simple, general, and GPU-friendly method for computing dual mesh and iso-surfaces of Adaptive Mesh Refinement (AMR) data. *arXiv* **2020**, arXiv:2004.08475.
33. Schäfer, H.; Keinert, B.; Nießner, M.; Stamminger, M. Local painting and deformation of meshes on the GPU. *Comput. Graph. Forum* **2015**, *34*, 26–35. [\[CrossRef\]](#)
34. Schäfer, H.; Keinert, B.; Stamminger, M. Real-time local displacement using dynamic GPU memory management. In Proceedings of the 5th High-Performance Graphics Conference (HPG '13), Anaheim, CA, USA, 19–21 July 2013.

35. Valdetaro, A.; Nunes, G.; Raposo, A.; Feijó, B.; De Toledo, R. Understanding Shader Model 5.0 with DirectX11. In Proceedings of the IX Brazilian Symposium on Computer Games and Digital Entertainment, Florianópolis, Brazil, 8–10 November 2010.
36. Catmull, E.; Clark, J. Recursively generated B-spline surfaces on arbitrary topological meshes. *Comput.-Aided Des.* **1978**, *10*, 350–355. [\[CrossRef\]](#)
37. Moreton, H. Watertight tessellation using forward differencing. In Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Workshop on Graphics Hardware, Los Angeles, CA, USA, 12–13 August 2001.
38. Boubekur, T.; Schlick, C. A flexible kernel for adaptive mesh refinement on GPU. *Comput. Graph. Forum* **2008**, *27*, 102–113. [\[CrossRef\]](#)
39. Boubekur, T.; Schlick, C. Generic mesh refinement on GPU. In Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Conference on Graphics Hardware, Los Angeles, CA, USA, 30–31 July 2005.
40. Boubekur, T.; Alexa, M. Phong tessellation. *ACM Trans. Graph.* **2008**, *27*, 141. [\[CrossRef\]](#)
41. Schwarz, M.; Stamminger, M. Fast GPU-based adaptive tessellation with CUDA. *Comput. Graph. Forum* **2009**, *28*, 365–374. [\[CrossRef\]](#)
42. Khoury, J.-N.; Dupuy, J.; Riccio, C. Adaptive GPU tessellation with compute shaders. In *GPU Zen 2: Advanced Rendering Techniques*; Black Cat Publishing: Encinitas, CA, USA, 2018; pp. 1–14.
43. Brainerd, W.; Foley, T.; Kraemer, M.; Moreton, H.; Nießner, M. Efficient GPU rendering of subdivision surfaces using adaptive quadrees. *ACM Trans. Graph.* **2016**, *35*, 113. [\[CrossRef\]](#)
44. Patney, A.; Ebeida, M.S.; Owens, J.D. Parallel view-dependent tessellation of Catmull–Clark subdivision surfaces. In Proceedings of the Conference on High Performance Graphics, New Orleans, LA, USA, 1–3 August 2009.
45. Nießner, M.; Loop, C.; Meyer, M.; DeRose, T. Feature-adaptive GPU rendering of Catmull–Clark subdivision surfaces. *ACM Trans. Graph.* **2012**, *31*, 6. [\[CrossRef\]](#)
46. Nießner, M.; Keinert, B.; Fisher, M.; Stamminger, M.; Loop, C.; Schäfer, H. Real-time rendering techniques with hardware tessellation. *Comput. Graph. Forum* **2016**, *35*, 113–137. [\[CrossRef\]](#)
47. Cook, R.L. Shade trees. *SIGGRAPH Comput. Graph.* **1984**, *18*, 223–231. [\[CrossRef\]](#)
48. Doggett, M.; Hirsch, J. Adaptive view dependent tessellation of displacement maps. In Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Workshop on Graphics Hardware, Interlaken, Switzerland, 21–22 August 2000.
49. Fisher, M.; Fatahalian, K.; Boulos, S.; Akeley, K.; Mark, W.R.; Hanrahan, P. DiagSplit: Parallel, crack-free, adaptive tessellation for micropolygon rendering. *ACM Trans. Graph.* **2009**, *28*, 150.
50. Nießner, M.; Loop, C. Analytic displacement mapping using hardware tessellation. *ACM Trans. Graph.* **2013**, *32*, 26. [\[CrossRef\]](#)
51. Jang, H.; Han, J. GPU-optimized indirect scalar displacement mapping. *Comput.-Aided Des.* **2013**, *45*, 517–522. [\[CrossRef\]](#)
52. Jang, H.; Han, J. Feature-preserving displacement mapping with Graphics Processing Unit (GPU) tessellation. *Comput. Graph. Forum* **2012**, *31*, 1880–1894. [\[CrossRef\]](#)
53. Moule, K.; McCool, M.D. Efficient bounded adaptive tessellation of displacement maps. In Proceedings of the Graphics Interface 2002, Calgary, AB, Canada, 27–29 May 2002.
54. Yusov, E.; Shevtsov, M. High-performance terrain rendering using hardware tessellation. In Proceedings of the 19th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG 2011), Plzeň, Czech Republic, 31 January–3 February 2011.
55. Ripolles, O.; Ramos, F.; Puig-Centelles, A.; Chover, M. Real-time tessellation of terrain on graphics hardware. *Comput. Geosci.* **2012**, *41*, 147–155. [\[CrossRef\]](#)
56. Mikhaylyuk, M.; Timokhin, P.; Maltsev, A. A method of Earth terrain tessellation on the GPU for space simulators. *Program. Comput. Softw.* **2017**, *43*, 243–249. [\[CrossRef\]](#)
57. Lambert, T.; Bénard, P.; Guennebaud, G. Multi-resolution meshes for feature-aware hardware tessellation. *Comput. Graph. Forum* **2016**, *35*, 253–262. [\[CrossRef\]](#)
58. Tiwary, A.; Ramanathan, M.; Kosinka, J.; Wilkie, A.; Banterle, F. Accelerated foveated rendering based on adaptive tessellation. In Proceedings of the Eurographics 2020—Short Papers, Norrköping, Sweden, 25–29 May 2020.
59. Sobel, I.; Feldman, G. A 3×3 isotropic gradient operator for image processing. In *Pattern Classification and Scene Analysis*; Duda, R.O., Hart, P.E., Eds.; Wiley: New York, NY, USA, 1973.
60. Wang, Z.; Bovik, A.C.; Sheikh, H.R.; Simoncelli, E.P. Image quality assessment: From error visibility to structural similarity. *IEEE Trans. Image Process.* **2004**, *13*, 600–612. [\[CrossRef\]](#)
61. Ruder, J.A.; Dickinson, E.Y.; Peindl, R.D.; Habet, N.A.; Fleischli, J.E. Greater tuberosity decortication decreases load to failure of all-suture anchor constructs in rotator cuff repair. *Arthroscopy* **2018**, *34*, 2777–2781. [\[CrossRef\]](#)
62. Cobaleda Aristizabal, A.F.; Barber, F.A. Rotator cuff repair: Technical considerations influencing optimum anchor choice in rotator cuff repair. *Ann. Jt.* **2021**, *6*, 18. [\[CrossRef\]](#)

63. Eng, E.M.; Childers, J.T.; Lack, B.T.; Haff, C.W.; Mouhawasse, E.; Jackson, G.R.; Sabesan, V.J. Arthroscopic shoulder simulation may improve short-term speed, accuracy, and efficiency of surgical movements in orthopaedic residents and fellows: A systematic review. *Arthroscopy* **2025**, *41*, 2623–2635. [[CrossRef](#)]
64. Crockatt, W.K.; Confino, J.; Kopydlowski, N.J.; Jobin, C.M.; Levine, W.N. Comparing skill acquisition and validity of immersive virtual reality with cadaver laboratory sessions in training for reverse total shoulder arthroplasty. *JBJS Open Access* **2023**, *8*, e22.00141. [[CrossRef](#)]
65. Hauschild, J.; Rivera, J.C.; Johnson, A.E.; Burns, T.C.; Roach, C.J. Shoulder arthroscopy simulator training improves surgical procedure performance: A controlled laboratory study. *Orthop. J. Sports Med.* **2021**, *9*, 23259671211003873. [[CrossRef](#)]
66. Rhodin, K.E.; Leraas, H.J.; Tracy, E.; Sudan, R.; Migaly, J. A role for live-animal models in undergraduate surgical education during the cadaver shortage. *Ann. Surg. Open* **2023**, *4*, e274. [[CrossRef](#)]
67. Walbron, P.; Common, H.; Thomazeau, H.; Hosseini, K.; Peduzzi, L.; Bulaid, Y.; Sirveaux, F. Virtual reality simulator improves the acquisition of basic arthroscopy skills in first-year orthopedic surgery residents. *Orthop. Traumatol. Surg. Res.* **2020**, *106*, 717–724. [[CrossRef](#)]
68. Mao, R.Q.; Lan, L.; Kay, J.; Lohre, R.; Ayeni, O.R.; Goel, D.P.; de Sa, D. Immersive virtual reality for surgical training: A systematic review. *J. Surg. Res.* **2021**, *268*, 40–58. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.