

RADmesh: Remesh-Aware Mesh Deformation

Nam Anh Dinh¹, Itai Lang¹, Oded Stein^{2,3}, and Rana Hanocka¹

¹ University of Chicago, Chicago IL 60637, USA
{namanh, itailang, ranahanocka}@uchicago.edu

² University of Southern California, Los Angeles CA 90089, USA
ostein@usc.edu

³ Technion, Haifa, Israel
oded.stein@technion.ac.il

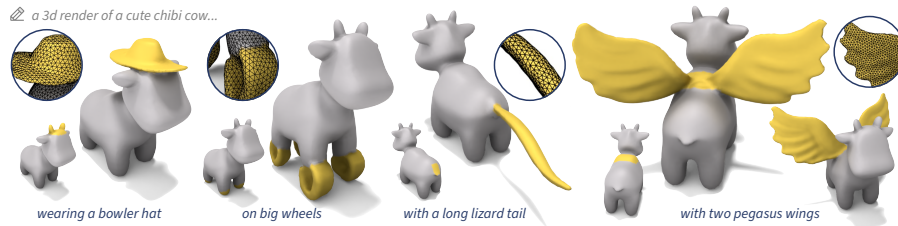


Fig. 1: By deforming together with remeshing in the loop, our method is capable of growing large, detailed parts and imparting drastic geometric changes on small selected regions (yellow) on the source shape (bottom left of each example), guided by text prompts. The insets show that the growths maintain isotropic triangulations with good element quality, indicating their suitability for the new geometry.

Abstract. We propose a remeshing-enhanced method for generatively deforming shapes with visual losses. It is intuitive that sufficiently drastic deformations of a mesh without changing its triangulation can easily compromise element quality, even if such large geometry changes may be semantically desired. Shape deformation methods could thus benefit from changing the triangulation; however, this is not done by most generative, text-based, visually-supervised mesh deformation methods. Remeshing is a discrete operation, proven to be especially challenging to couple with the notoriously noisy supervision signal provided by visual losses. We propose a vertex-based deformation optimization quantity capable of large deformations and robustness to such noise; we periodically remesh using an isotropic remeshing that interpolates and carries forward the deformation optimization state. This enables continuous, geometry-informed progress in coarse-to-fine addition of resolution. The resulting shapes’ triangulations fit their optimized geometry and have neat isotropic elements. Further, our method is localizable, able to grow new features on a base shape with expressive detail, leaving the rest unchanged. We showcase the effectiveness of our method on a variety of shapes and prompts, both local and global deformations, and demonstrate its superior visual quality and triangle efficiency. Our project page is at <https://threedle.github.io/radmesh>.

Keywords: Meshes · Deformation · Remeshing · Generative 3D

1 Introduction

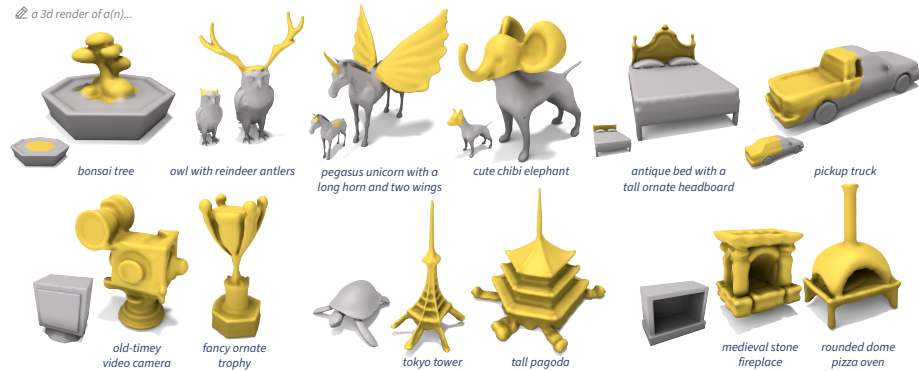


Fig. 2: Gallery of local growths and global detailing deformations. *Top row:* our method is capable of growing large parts out of small selected regions on diverse shapes, organic and manufactured. The results show symmetry, high detail even at extremities, and clean geometry at selection boundaries. *Bottom row:* we can also perform global “detailization” deformations, where the whole shape is allowed to change. This is useful on simple base shapes and demonstrates the ability to change both high-level structure and finer details.

Many recent methods optimize mesh deformations with visual losses as guidance [16, 20, 26]. These methods largely achieve their shape editing goals through finding vertex displacements and do not change mesh connectivity. While this approach is effective for a limited class of edits, there remains a wide class of shape deformations that are limited when keeping mesh connectivity constant. Growths, appendages, thin sheet-like volumes, and other large-scale structural reconfigurations generally cannot be achievable without excessive stretching or shrinking of triangles. Such edits may result in bad triangle quality, poor fidelity, and unsuitably small or large numbers of triangles forced to represent certain features of the shape.

We posit that such expressive deformations in the fully explicit setting would strongly benefit from the ability to modify mesh connectivity, as connectivity alone implies some information about the underlying geometry. However, deformation editing methods have rarely incorporated remeshing, being inherently discrete in the required modifications of indices/pointers, making it challenging to involve discrete triangulation choices in the optimization (*e.g.* [49]).

Existing visual supervision-based remeshers [6, 45] are meant for the inverse rendering task and are built with heuristics deeply tied to the use of *vertex positions* as the optimization variable. In contrast, the generative deformation task has been shown to greatly benefit from using differential deformation quantities as the optimization variable instead. These quantities, such as per-face Jacobians [20, 26] and per-vertex rotations [16], solve into smooth, globally-informed

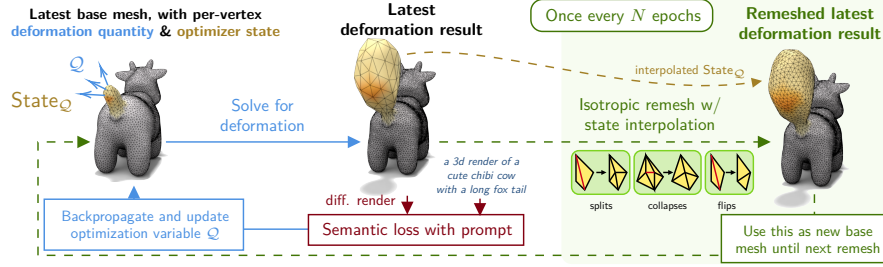


Fig. 3: Overview of our optimization loop. Geometry is optimized via gradient descent of our deformation quantity Q , with associated optimizer state, with respect to a base mesh. At epoch i , we solve for the deformation using Q_i (Sec. 3.2), render, take visual loss, and update the quantity. Once every N epochs, we remesh the last deformed shape, interpolate the optimizer state onto the result, and use it as the new source mesh until the next scheduled remesh. (Note that the deformation in this figure is exaggerated for illustrative purposes; in practice, the remeshes are frequent enough that the deformation between each remesh is more subtle.)

deformations in the face of the noisy loss signals (namely SDS) used in the text-based, visually-supervised generative setting.

In our method, *RADmesh*, we propose a differential deformation optimization pipeline enhanced by remeshing, bringing the added expressiveness of connectivity modifications to the generative deformation task and its preferred deformation representations.

We extend the vertex normal-based deformation quantity of Geometry in Style [16]. As a vertex-based quantity, it allows straightforward transfer and interpolation from old mesh elements to new mesh elements during remeshing. Augmented with optimizable local scale in addition to local rotation, in tandem with remeshing at regular intervals, it enables growing large, detailed geometries. Deformation and remeshing can be localized to specific regions by construction, or globally applied across the mesh. We employ an off-the-shelf optimizer (Adam [27]) to update the deformation quantity, and interpolate its tracked state to the new discretization upon remesh. We adopt a standard isotropic remeshing algorithm, modified to enable this interpolation.

We demonstrate that this simple integration of remeshing together with our deformation quantity lets us achieve flexible shape optimizations that work well in generative settings with text-guided SDS (part growth, abstract shape detailing). Other methods that perform similar generative 3D tasks execute generation on dedicated proxy representations, *e.g.* generating implicit shapes or multi-view image inpainting followed by 3D reconstructions. In contrast, we do not generate via alternative intermediate representations: we achieve desirable results by generative optimization *directly on meshes*, using an image diffusion model. Our results highlight the effectiveness and versatility of a purely mesh-based generative optimization approach.

2 Related Work

Remeshing and triangle refinement. Classical geometry processing has a large body of work on remeshing and triangle refinement. We refer to [3, 25] for general surveys of remeshing methods for triangle meshes. An overarching goal is uniformly-sized triangles, with no excessively small or large angles, numerically desirable for algorithms that are finite element discretizations, *e.g.* those that use cotangent edge weights (which our method and others [1] fall under).

Remeshers belong to a few categories. One family of methods performs mesh parameterization followed by point sampling and Delaunay (or similar) triangulation on the parameter domain [2, 4, 47, 49, 58]. Another family uses only local topology-preserving connectivity edits (edge collapses, splits, flips) to iteratively improve triangles across the mesh. Of these, particularly relevant are the works of Botsch & Kobbelt *et al.* [10] and Dunyach *et al.* [17], iterative methods for mesh improvement given a target edge length (either global or adaptive based on curvature). Edges are collapsed and split to maintain the target length range; edge flips and tangential smoothing improve vertex valence and triangle quality. This yields isotropic triangulations with near-equilateral triangle aspect ratios. Using these on-the-fly remeshing operations, editing [17, 59] and correspondence [41] paradigms have also been proposed. Similar operations have been built to simplify intrinsic triangulations [36] and non-manifold meshes [38].

Vertex optimizers with remeshing. Recent applications of on-the-fly remeshing have included the inverse rendering and mesh reconstruction task, where remeshing is implemented directly in optimizers whose optimization variable is vertex positions. In particular, Palfinger [45] proposed a vertex optimizer and remesher that adapts Adam [27] to take gradient statistics to determine and apply local remeshing operations (collapses, splits, flips). Connectivity is added in areas experiencing large change, making available more triangles for the geometry to evolve towards the inverse rendering target (*e.g.* multiview images). Building on this work, Barda *et al.* [6] improved heuristics and incorporated a raycaster; this has since been used for reconstructing meshes from other representations (*e.g.* implicit fields [8] and multiview 2D inpainting [7]). In contrast, our use of remeshing resembles that of [43, 66], applying an isotropic remesher coarse-to-fine in a geometry optimization loop. Our vertex-based deformation quantity lets us remesh with interpolation of the geometry optimizer state, allowing more expressive and stable deformations than optimizing raw vertex displacements.

Generative mesh deformation. There is a sizable body of recent work in mesh deformation-based generative modeling of shapes. The seminal Neural Jacobian Fields method [1] has inspired a range of generative applications that use deformation for either primary generation or refinement of shapes. Methods with deformations as the main generative medium include TextDeformer [20], MeshUp [26], and Geometry in Style [16]. TextDeformer and MeshUp use Jacobians; Geometry in Style uses normals with differentiable As-Rigid-As-Possible (dARAP) solves, though in a similar pipeline. In these methods, including our work, deformations (guided by a semantic visual loss) are the *only* means of ge-

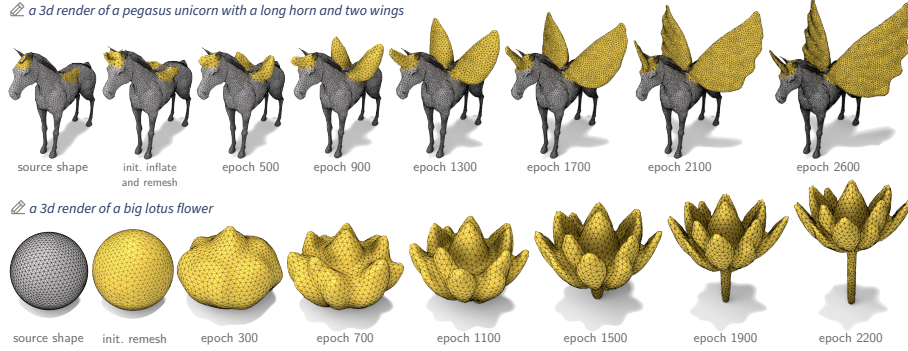


Fig. 4: Evolution. We visualize the geometric evolution through the course of two example optimizations, one localized and the other a global deformation. The local growth example involves disjoint selected components (the wings, the horn), which all grow simultaneously. The global deformation shows an ability to form complex geometric detail (starting from a shape as basic as a sphere) and appendages while maintaining clean isotropic triangulation throughout.

ometry production. Other work has used Jacobian deformation to create coarse geometry that is then textured [70], constrained creation of head avatars [62], improving a shape generated upstream [72], handle-based deformations [5, 71], and shape correspondence [40, 57].

These studies are part of the wider literature on shape generation, which has seen a proliferation of methods, particularly implicits, images, or latent representations followed by mesh reconstruction or improvement [11, 18, 23, 24, 28, 32, 33, 44, 54, 56, 65, 67–69, 74, 76, 78], and autoregressive sequence models of discrete elements [19, 21, 30, 53, 60, 64, 73, 77]. Many of these methods have been enabled by advances in lifting 2D visual priors to 3D in a variety of generative and analysis tasks and representations [12–15, 22, 31, 34, 39, 42, 46, 48, 50, 61, 63, 63, 75].

3 Method

3.1 Overview

Given an original source mesh $\mathcal{M}_0$ with vertices and faces $(\mathcal{V}_0, \mathcal{F}_0)$, we wish to *deform and remesh* $\mathcal{M}_0$ to find a new mesh $\mathcal{M}^* = (\mathcal{V}^*, \mathcal{F}^*)$ that meets a target text prompt $\mathbf{x}$. A selection region of vertices can optionally be specified, represented as a binary selection mask ($\text{sel}(k) = 1$ denotes vertex k is part of the selection, and $\text{sel}(k) = 0$ denotes k is not selected), where deformation and remeshing are meant to be restricted to selected vertices only.

Fig. 3 presents an overview of the optimization loop and our use of remeshing. The desired mesh $\mathcal{M}^*$ is constructed via gradient descent optimization; at any given iteration or epoch i , this loop’s optimization variable is a *deformation quantity* $\mathcal{Q}_i$ defined with respect to the loop’s *current base mesh* $\mathcal{M}^{r_i} = (\mathcal{V}^{r_i}, \mathcal{F}^{r_i})$ where r_i denotes the current number of remeshes done thus far at epoch i .

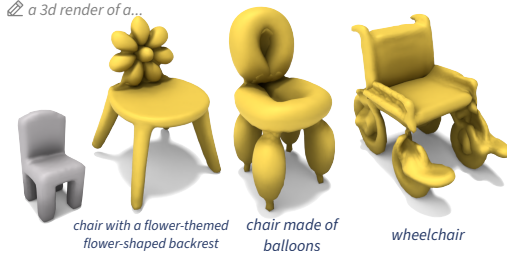


Fig. 5: Global detailization. Our method is able to take basic shapes and globally deform & remesh them expressively towards prompts. The results keep high-level structure (*e.g.* four similar legs) while exhibiting expressive geometric detail: the folds on the *flower-themed flower-shaped* backrest, the tapered lobes of the *chair made of balloons*, the *wheelchair’s* wheels.

is then rendered with a differentiable renderer, and the images fed to a cascaded score distillation (CSD) loss providing the semantic supervision to be backpropagated to update the deformation quantity. This gives $\mathcal{Q}_{i+1}$ for the next epoch (epoch $i + 1$); see Sec. 3.6.

At regular intervals in this optimization loop, we interrupt the geometry optimization and perform a remesh using the remeshing function R . The remesh will *replace* the current base mesh with a remesh of the latest deformation result. More precisely, let $\text{do_remesh}(i) = 1$ mean that a remesh is scheduled for optimization epoch i . For each i such that $\text{do_remesh}(i) = 1$, we compute the *next base mesh* $\mathcal{M}^{r_{i+1}} := R(\mathcal{V}'_i) = R(D(\mathcal{M}^{r_i}, \mathcal{Q}_i))$. We describe the remeshing method R in Sec. 3.3. This leads to $\mathcal{Q}_{i+1}$ becoming reinitialized, since the base mesh $\mathcal{M}^{r_i}$ that $\mathcal{Q}_i$ was associated with has now been switched out for $\mathcal{M}^{r_{i+1}}$. Crucially, we also interpolate and carry forward optimization states in order to then resume optimizing geometry after the remesh; we describe this in Sec. 3.4.

3.2 Deformation Quantity and Method

The deformation quantity $\mathcal{Q}$ we use is a 6-component vector for each vertex, *i.e.* $\mathbf{q}_k \in \mathbb{R}^6$ for each vertex k , and so $\mathcal{Q} = \{\mathbf{q}_k \in \mathbb{R}^6 \mid k \in \{1 \dots |\mathcal{V}|\}\}$. The first three features, which we denote by $\mathbf{q}_k^{\text{dir}} \in \mathbb{R}^3$, describe a direction vector, and the last three features, denoted by $\mathbf{q}_k^{\text{scale}} \in \mathbb{R}^3$, describe a scaling vector for the three world axes.

This deformation quantity is an extension of the per-vertex $\mathbb{R}^3$ normal vector quantity of Geometry in Style [16]. Originally proposed for its inherent regularization using differentiable ARAP, its vertex-based deformation quantity is also advantageous in a remeshing setting. Our remesh operations operate with vertex attributes; thus, a vertex-based optimization quantity makes it straightforward to interpolate optimizer state onto new mesh elements during a remesh. This

The *current base mesh* is initialized to $\mathcal{M}^0$, which is the original source mesh $\mathcal{M}_0$ with preprocessing applied (described in Sec. 3.5).

At optimization epoch i , a deformation function D finds deformed vertices $\mathcal{V}'_i$ using the latest value of the deformation quantity $\mathcal{Q}_i$ together with the associated base mesh $\mathcal{M}^{r_i}$ for which it is defined, *i.e.* $\mathcal{V}'_i := D(\mathcal{M}^{r_i}, \mathcal{Q}_i)$. This deformation gives the shape to be rendered for visual losses, and keeps the current base triangulation $\mathcal{F}^{r_i}$ fixed. The deformation formulation is described in Sec. 3.2.

The deformed mesh $(\mathcal{V}'_i, \mathcal{F}^{r_i})$

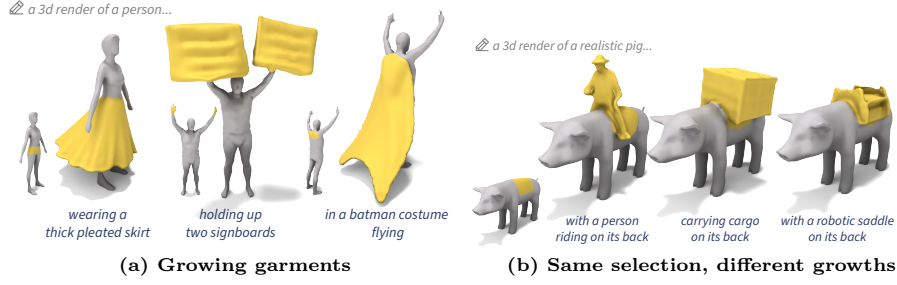


Fig. 6: More local growth examples. (a) **Growing garments:** Our method is able to generate thin parts and garments from small selection regions on human shapes, extending considerably outwards from the original extents with high detail at the generated extremities. (b) **Same selection, different growths:** On the same selection and mesh, our method can deform & remesh to produce varied growths, such as the *person* riding (including a detailed hat), the *cargo* with geometric texture on the surface, and the *robotic saddle* with sharp features.

contrasts with the more conventional per-face Jacobians [1], a face-based quantity that would be more lossy to cast onto vertices and interpolate during a remesh. In order to support expressive deformations and growing large parts beyond the extents of the original selection (see ablation in Fig. 9a), we append features to Geometry in Style’s representation that encode scale in addition to rotation. Accordingly, the deformation function D is

$$D(\mathcal{M}, \mathcal{Q}) = \text{GlobalStep}(\mathcal{M}, \text{LocalStep}(\mathcal{M}, \mathcal{Q})) \quad (1)$$

The local step first finds a local transform T_k for each vertex k , composed of a rotation matrix R_k computed using $\mathbf{q}_k^{\text{dir}}$ and a diagonal scaling matrix S_k formed from $\mathbf{q}_k^{\text{scale}}$. The global step then solves for global vertex positions that best satisfy those local transformations.

The rotation is found by Eq. (2), a Procrustes problem, which finds a rotation matrix R_k taking the set of vectors $\mathcal{N}_k \cup \{\mathbf{u}_k\}$ to the set of vectors $\mathcal{N}_k \cup \{\mathbf{q}_k^{\text{dir}}\}$, where $\mathbf{u}_k$ is the normal of vertex k in the current base mesh and $\mathcal{N}_k$ is the set of edge vectors in the spokes-and-rims neighborhood of vertex k . Here, $\mathbf{p}_i, \mathbf{p}_j$ are the position vectors of vertices i and j ; w_{ij} is the cotangent weight of edge (i, j) ; a_k is the Voronoi area of vertex k ; and the hyperparameter λ controls the strength of the rotation towards the goal $\mathbf{q}_k^{\text{dir}}$, as described in [16] and [37].

$$R_k = \underset{\tilde{R}_k \in \text{SO}(3)}{\text{argmin}} \sum_{(i,j) \in \mathcal{N}_k} w_{ij} \|\tilde{R}_k(\mathbf{p}_j - \mathbf{p}_i) - (\mathbf{p}_j - \mathbf{p}_i)\|_2^2 + \lambda a_k \|\tilde{R}_k \mathbf{u}_k - \mathbf{q}_k^{\text{dir}}\|_2^2 \quad (2)$$

$$S_k = \text{diag}(\mathbf{q}_k^{\text{scale}}) \quad (3)$$

$$T_k = S_k R_k \quad (4)$$

$$\text{LocalStep}(\mathcal{M}, \mathcal{Q}) = \{T_k \mid k \in \{1 \dots |\mathcal{V}|\}\} \quad (5)$$

The global step is the differentiable ARAP global solve from Geometry in Style except for the use of our local step’s composed transformations $\mathcal{T} = \{\mathbf{T}_k \mid k \in \{1 \dots |\mathcal{V}|\}\}$ rather than just the rotations $\{\mathbf{R}_k \mid k \in \{1 \dots |\mathcal{V}|\}\}$.

$$\begin{aligned} & \text{GlobalStep}(\mathcal{M}, \mathcal{T}) \\ &= \underset{\mathcal{V}'}{\operatorname{argmin}} \sum_{k \in \{1 \dots |\mathcal{V}|\}} \sum_{(i,j) \in \mathcal{N}_k} w_{ij} \|\mathbf{T}_k(\mathbf{p}_j - \mathbf{p}_i) - (\mathbf{p}'_j - \mathbf{p}'_i)\|_2^2 \end{aligned} \quad (6)$$

Restricting deformation to a selected region. To restrict the deformation to the selected region, we 1) set $\mathbf{T}_k$ to be identity for k where $\text{sel}(k) = 0$ and 2) substitute the original positions of these vertices into Eq. (6) before solving for the unknowns. The solutions are thus the deformed positions for only the selected vertices. We describe the matrix form of this solve in the supplementary material.

Designing constraints on deformations. Having separate $\mathbf{q}_k^{\text{scale}}$ and $\mathbf{q}_k^{\text{dir}}$ components (scale & direction) on the deformation quantity lets us constrain the possible deformations *by construction* (rather than by losses). For global deformations, we apply a soft clamp (leaky ReLU) on the $\mathbf{q}_k^{\text{scale}}$ input to Eq. (3) with a leaky ReLU floor of 0.98. This combats the supervision model’s tendency towards shrinkage when the whole mesh is optimizable; see supplementary material for ablation.

Initialization. Upon a new base mesh, for vertex i , $\mathbf{q}_i^{\text{dir}}$ is initialized to the area-weighted vertex normal, and $\mathbf{q}_i^{\text{scale}}$ is initialized to all 1.

3.3 Remeshing

We use the remeshing method of Botsch & Kobbelt [10]. As an isotropic remesher that aims to produce uniform, near-equilateral triangles given a target edge length, one iteration of the method involves 1) an edge collapse step, 2) an edge split step, 3) a valence-improving edge flip step, 4) a tangential smoothing step, and 5) a projection step to project the result of these steps to the original surface. These steps make up one iteration; many such iterations can be done for better isotropic triangulations. In practice, we perform 2 such iterations once every 100 deformation epochs.

In propagating the vertex selection, we incorporate the selection flag $\text{sel}(k)$ as a real-valued vertex attribute $\text{sel}_{\mathbb{R}}(k)$. All remeshing operations (edge collapse, edge flip, edge split) only occur on edges (u, v) where $\text{sel}_{\mathbb{R}}(k) \geq 0.5$ for both $k \in \{u, v\}$. Edge collapses and splits grant the resulting vertex w with $\text{sel}_{\mathbb{R}}(w) = \frac{1}{2}(\text{sel}_{\mathbb{R}}(u) + \text{sel}_{\mathbb{R}}(v))$. After a complete remesh iteration (*i.e.* after reprojection), every $\text{sel}_{\mathbb{R}}(k)$ is rounded to 1 if at least 0.5, and 0 otherwise.

The Botsch & Kobbelt method requires a target length for each edge. We use a global target length for all edges involved in remeshing. This length is on a coarse-to-fine schedule, critical for substantial growth of geometry as shown in Fig. 10a. We describe our schedule for the local and global tasks in Sec. 4.

Despite the connectivity change, we retain correspondences with the original mesh. We keep perfect face and vertex correspondence outside modified regions (see Fig. 12). Within modified regions, vertex attributes are interpolated through; the optimizer state is one such attribute (Sec. 3.4).

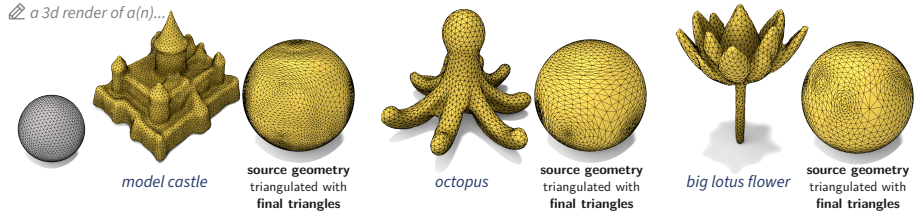


Fig. 7: Generated triangulation tailored for the generated geometry. Taking global deformation of a sphere as an example, we show that each final triangulation, applied to the original geometry, encodes the structures and appendages requiring high vertex density in the result shape. This shows the appropriateness and adaptivity of the generated triangulation for the generated geometry, not merely a uniform upsample of the original triangulation.

3.4 State Interpolation for Resuming Optimization

We note that the reprojection step involves computing barycentric coordinates on the pre-remesh triangles for placing new vertices. We take advantage of this to interpolate the *optimizer state*. In most optimizers, including Adam [27] which we use, the state is an accumulator of the gradient with respect to the variable. Since our deformation quantity is vertex-based, the optimizer state is also vertex-based and can be easily interpolated with barycentric coordinates. In the supplementary and Fig. 9b, we show that this interpolation is important for stable, symmetric growths. In global deformations, it also prevents unchecked scale-ups leading to excess resolution without better geometry.

3.5 Preprocessing

We normalize all starting meshes such that their axis-aligned bounding box fits within the standardized cube $[-1, 1]^3$ in world coordinates. After normalization, for the local growth task in particular, we apply *initial inflation* to provide a starting geometry, which SDS supervision is sensitive to. To each initial selected vertex we add a displacement of length ℓ along its vertex normal direction. We find ℓ by a heuristic formula based on the sum volume of the selected patches, computed after closing holes making them watertight surfaces: $\ell = a^{-b} \sqrt[3]{V}$ where V is the sum volume of all hole-closed selection patches. For positive a, b , this formula grants larger inflation to smaller selections, the relation decaying exponentially. (See supplementary material for an ablation on this initial inflation step.) Preprocessing concludes with a first remesh of the base geometry before the optimization loop begins.

3.6 Differentiable Rendering & Supervision

To obtain supervision for the deformation quantity, we follow the common protocol of differentiable rendering into images followed by a 2D image diffusion

model providing score distillation sampling (SDS) loss. We use Cascaded Score Distillation [14], an SDS variant making use of the multiple diffusion stages of DeepFloyd IF [55]. We use `nvdiffrast` [29] as our differentiable renderer.

4 Experiments

Hyperparameters. Each remesh consists of 2 Botsch & Kobbelt iterations. A remesh happens every $N = 100$ epochs. Local growth runs last at most 2600 epochs, and global deformations 2200 epochs. The initial inflation (Sec. 3.5), used only for local growths, has a, b empirically set to $a = 4.3$, $b = 5.2$. We use $\lambda = 8$ for Eq. (2). Each epoch uses 8 views sampled uniform-randomly from fixed ranges for azimuth, elevation, and FOV (settings are shared across all runs within each task: local growths, global deforms). See supplementary material.

Target length schedule. For local growths, the coarse-to-fine target length schedule begins at $1.7\times$ the average edge length of the mesh after all preprocessing, linearly ramping from $1.7\times$ to $1.3\times$ over the next 10 remeshes, then from $1.3\times$ to $1.0\times$ over the next 5 remeshes, then $1.0\times$ for the remainder of remeshes. Global deformations begin at $1.4\times$ the average edge length of the preprocessed mesh, ramp to $1.0\times$ over the next 18 remeshes, and set to $1.0\times$ for the remainder. We show the significance of this coarse-to-fine schedule as opposed to remeshing finely once or remeshing curvature-adaptively in Figs. 10a and 10b.

4.1 Qualitative Evaluation

Local growths: large extents & clean detail. We demonstrate our method’s ability to impart large-scale yet detailed growth of whole parts to even small selections on a variety of shapes, organic and manufactured. The growths extend far beyond the source selection’s spatial extent, such as the *long lizard tail*, the *two pegasus wings* (Fig. 1), the large *cargo* on the pig’s back (Fig. 6b), the large *reindeer antlers* (Fig. 2), and the skirt and cape growths (Fig. 6a).

They also exhibit expressive geometry even at the far ends of growths: the antlers’ branching tines (Figs. 2 and 8), the curled trunk of the *elephant* (Fig. 2), the feather texture on the *wings* examples in Figs. 1 and 2, the “text” on the signboards in Fig. 6a, and a hat on the growth of a person riding on a pig’s back in Fig. 6b. Noteworthy is also the ability to “remove” and reconfigure structures: in the *bowler hat* example (Fig. 1) the horns dissolve into a hat; in Fig. 2 the back of the car shrinks to become the *pickup truck*’s bed.

Expressive global detailization. When the selection is the entire shape, our method generates highly detailed geometry out of plain, simple shapes, closely matching the text prompt. Fig. 5 demonstrates the deformation of a chair subject to different prompts. While the source chair has simple geometry, it is successfully deformed and remeshed into significantly detailed, distinct outputs. For example, the backrest turns into a crisp flower geometry, handles and wheels emerge, or balloon-like elements form. Even with drastic changes, the chair’s semantic structure (backrest, seat, four legs) is preserved.

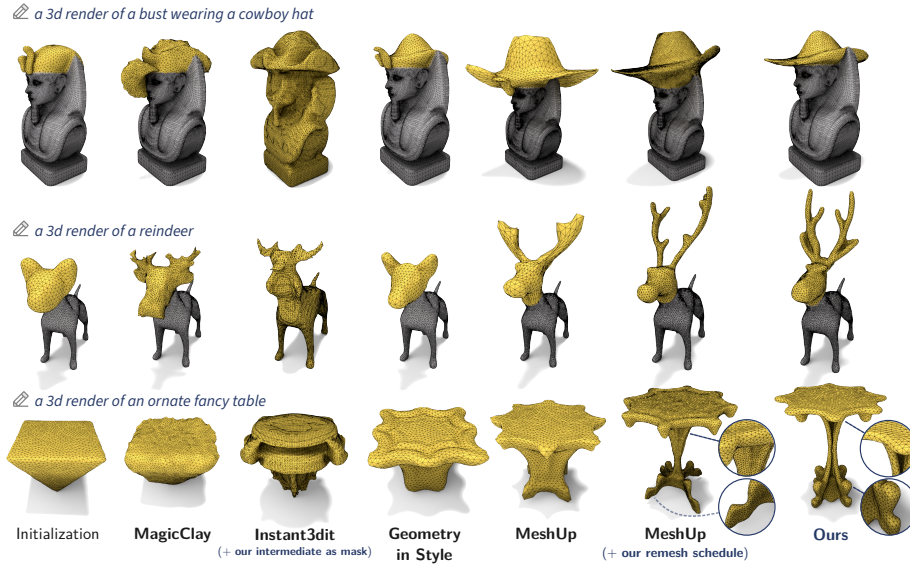


Fig. 8: Qualitative comparison. We compare our method with recent mesh deformation techniques [7, 8, 16, 26] for both local growths and global detailization. The baselines produce noisy results (MagicClay and Instant3dit) or are limited in the generated geometry expressivity (Geometry in Style and MeshUp). Our remeshing schedule also helps MeshUp, though our method still produces cleaner, symmetric deformations, with more efficient use of resolution (see insets), aligning better with the source mesh. Note that Instant3dit requires a 3D mask to inpaint; we use an early epoch from our run as this mask, plus 0.25 dilation, since the selection mask alone restricts growth-like inpainting. Instant3dit’s public code does not include the adaptive remesher or closed-source mesh LRMs from the paper.

Taking global detailization further, in Fig. 7 we showcase the deformation of a primitive source shape as simple as a sphere. Beyond the expressive outputs, this experiment stresses the specificity of our remeshing process: rather than a mere subdivide or upsample of the source geometry, triangles appear *as needed* to form the desired outcome. This corresponds to an efficient, *non-uniform* triangulation when viewed on the original geometry: triangle density corresponds to the *model castle*’s towers, the *octopus* tentacles, and the *lotus flower*’s receptacle and petals, and is sparser elsewhere. We attribute this to the frequent coarse-to-fine remesh schedule, accompanying the geometry optimization as it forms new parts and details. Our optimizer state interpolation also helps: the [10] remesher aims to get all edges to a target length, and when growth momentum is carried through remeshing via our optimizer state interpolation, actively growing regions (edges becoming longer) also get more triangles as needed.

4.2 Comparison

We contrast RADmesh with some recent generative deformation baselines [7, 8, 16, 26]. Fig. 8 presents visuals; Tab. 1 reports numeric results on 64 shape-prompt

Table 1: Quantitative comparison: CLIP and VQA scores ($\times 100$, higher is better), average face quality (FQ) ($\times 100$, higher is better), average face counts (#F). We evaluate on $n = 64$ shape-prompt pairs (see supplementary material for details). Our method achieves superior CLIP and VQA score, suggesting better perceptual quality and adherence to prompts, as well as superior triangle quality while being efficient with face budget among the methods that remesh. (*Geometry in Style and MeshUp do not remesh; their average face counts thus reflect the resolutions of the source meshes.)

Method	CLIP $\uparrow$	VQA $\uparrow$	FQ $\uparrow$	#F
MagicClay [8]	29.821	57.150	86.784	17203.55
Instant3dit [7]	28.399	52.111	80.132	40034.69
Geometry in Style [16]	29.279	52.120	94.146	11308.06*
MeshUp [26]	30.725	66.093	84.831	11308.06*
MeshUp + our remesh	31.148	69.360	96.310	29461.34
RADmesh (ours)	31.416	72.930	96.516	16963.75

pairs. Rather than defining the deformation objective on the mesh surface, MagicClay and Instant3dit generate on intermediate proxy representations: implicit SDFs and multiview image inpainting, respectively. These representations are lossy, more detached from the formation of a mesh, resulting in artifacts. In contrast, Geometry in Style and MeshUp deform the mesh directly. However, both lack remeshing and are not expressive enough. Our method enjoys both worlds: it deforms and remeshes the mesh itself, generating highly detailed and clean geometries. Notably, adding our remesh schedule to MeshUp boosts expressivity, showing the benefit of remeshing in explicit mesh deformation.

Quantitatively we evaluate via four metrics: CLIP score [48] measuring CLIP similarity of result renders to the text prompt; VQA score [35] as a perceptual score for text-to-visual generative models; average triangle face quality (FQ) defined as $4\sqrt{3}\frac{A}{\ell_1^2 + \ell_2^2 + \ell_3^2}$ [9, 52] (face area A , edge i length ℓ_i) where equilateral triangles achieve the ideal score of 1; and face count, measuring triangulation efficiency. Among compared methods, ours achieves the best CLIP and VQA scores, indicating perceptual quality and fidelity to the text prompt. Since we regularly run isotropic remeshing, we maintain higher quality triangles than competing methods, which either only distort triangles without remeshing, or reconstruct towards proxy representations, not ensuring good elements. Our quality is also more efficiently achieved, with a lower face count than the baselines that remesh.

4.3 Ablation Study

Central to our method is the addition of the optimizable scale parameter, the optimizer state interpolation through the remeshing, and the coarse-to-fine remesh schedule. We validate these design choices with ablation experiments. In Fig. 9a, we show that the scale parameter, $\mathbf{q}_k^{\text{scale}}$ (Sec. 3.2) is critical for growing large parts and appendages, with detail and sharpness even at extremities. In Fig. 9b,

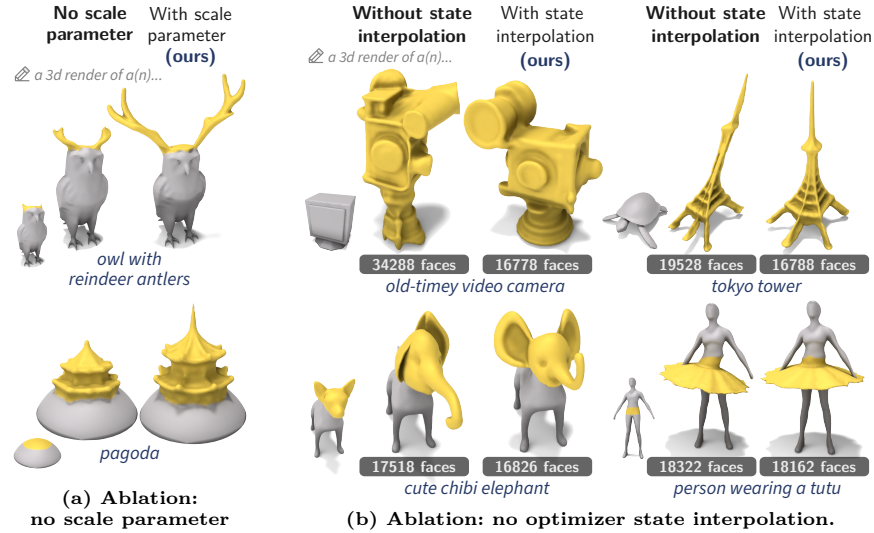


Fig. 9: Ablations. (a) **No scale parameter:** Without the optimizable scale parameter, growths fail to attain large extents and have blunted, less sharp detail at the extremities. (b) **No optimizer state interpolation:** Without optimizer state interpolation, the optimization loses all gradient history upon each remesh. This leads to less stable deformations, asymmetries, exaggerated view-dependent artifacts, and, in some cases, excessive resolution or scale-up without better geometry.

we show that when optimizer state interpolation is disabled, the deformation becomes unstable, as the optimization history is wiped after each remesh. This setting leads to two undesired effects: uneven results with geometric artifacts, as well as increased face counts. In Fig. 10a, we stress the importance of our coarse-to-fine remeshing schedule: remeshing to high resolution once complicates the optimization landscape and compromises expressivity. Similarly, Fig. 10b shows that using adaptive remeshing [17] (*i.e.* giving more resolution to higher-curvature regions), also fails to support large growths. Committing to dense triangles early, especially in areas undergoing rapid shape change whose curvature is not yet finalized, compromises the coarse-to-fine property. These results indicate that uniform, gradual remeshing allows better exploration of the deformation space, allowing efficient evolution of large and detailed geometry, as also visualized in Fig. 4.

4.4 Applications and Workflows

We demonstrate some extended workflows using our method. Fig. 11 showcases a workflow comprising multiple prompts: we can iteratively add parts (arms, head, and a pineapple on the head) to the torso with high detail, including on top of previously grown parts. Fig. 12 shows a workflow on a textured mesh. Our method propagates vertex data within the selection region, and keeps all

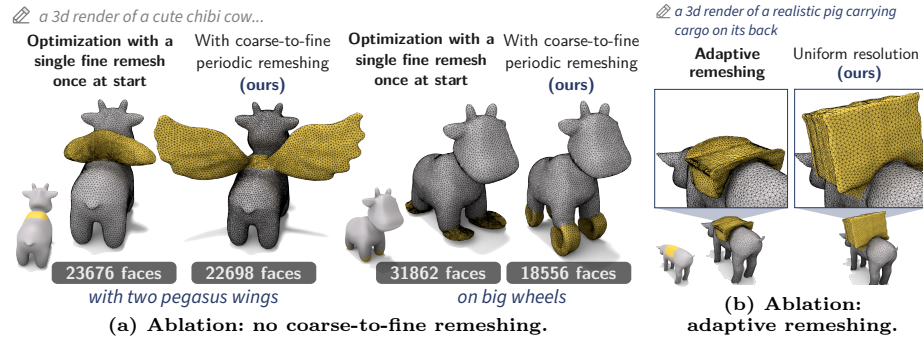


Fig. 10: Ablations on remeshing. (a) **No coarse-to-fine remeshing:** Simply finely remeshing the region *only once at the start*, even to higher resolutions than the final resolution of our coarse-to-fine schedule, fails to produce quality growths. This shows the significance of a coarse-to-fine remeshing schedule. (b) **Adaptive remeshing:** Using adaptive remeshing (giving more resolution to high-curvature areas) rather than uniform isotropic remeshing also breaks the coarse-to-fine property, compromising the ability to grow large geometries.

correspondence, corner UVs, and data outside the selection region. The new geometry can be easily re-unwrapped and retextured.

4.5 Limitations

Running time is dominated by the slow supervision, being SDS-based: on a single L40S GPU, a local growth run (2600 epochs) takes 85-90 minutes, and a global run (2200 epochs) 70-75 minutes. However, epochs earlier than this may already sufficiently satisfy the prompt. The use of a Poisson solve also requires the mesh to be manifold, a restriction shared by all other such Poisson solve-based methods (Geometry in Style [16], MeshUp [26], other NJF [1]-based systems.) However, integrating non-manifold mesh Laplacians [51] and non-manifold connectivity edit operations [38] is a promising direction for future work.

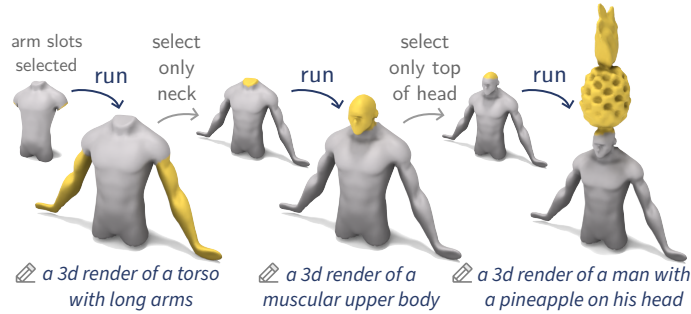


Fig. 11: Iterative editing. We progressively add parts to the torso mesh by iterative selections (including on top of previously grown parts) and prompts.

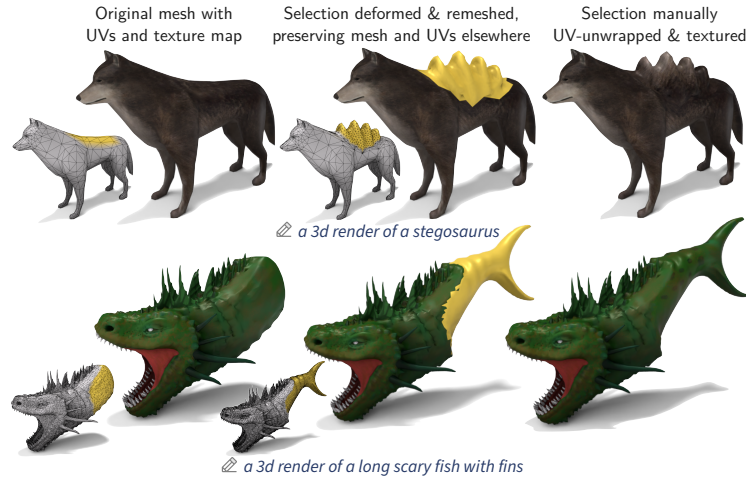


Fig. 12: Workflow on textured meshes. Despite changing connectivity, our method interpolates vertex attributes inside, and fully preserves correspondence, geometry, and data of mesh elements outside the region selected for deforming and remeshing. This includes *e.g.* corner UVs. We demonstrate the workflow of deforming and remeshing a region on a mesh with existing UVs and texture: the rest of the mesh is unchanged, and the user can choose to unwrap and texture the newly grown geometry.

5 Conclusion

We presented a technique for learning to deform and change the triangulation of a mesh using noisy visual losses. Our framework optimizes a controllable, robust, smooth per-vertex deformation quantity instead of raw vertex positions. Our remesher propagates this quantity’s optimization state across discrete remeshing operations, allowing optimization remain coherent despite the inherently non-differentiable nature of connectivity changes. As a result, our method smoothly molds meshes to have new extended parts and detail with stability, fidelity, triangle quality and efficiency.

In the future, we are interested in applying this framework for learning desirable triangulations and for quad meshing. In our current formulation, all the remeshing operations (edge collapse, splits, flips) are genus-preserving. Looking ahead, we aim to incorporate mesh tearing and merging operations that would allow the optimization to introduce topological holes and genus modifications.

Acknowledgments

This work was supported by NSF grants #2304481, #2335493, #2402894, #2542757, BSF grant #2022363, gifts from Adobe, Snap, and Google, and the Bennett Family AI + Science Collaborative Research Program. We thank the members of the 3DL lab for valuable discussions and feedback; we are grateful for the computing resources, services, and staff expertise at the University of Chicago.

References

1. Aigerman, N., Gupta, K., Kim, V.G., Chaudhuri, S., Saito, J., Groueix, T.: Neural jacobian fields: learning intrinsic mappings of arbitrary meshes. *ACM Trans. Graph.* **41**(4) (Jul 2022). <https://doi.org/10.1145/3528223.3530141>, <https://doi.org/10.1145/3528223.3530141>
2. Alliez, P., Meyer, M., Desbrun, M.: Interactive geometry remeshing. *ACM Trans. Graph.* **21**(3), 347–354 (Jul 2002). <https://doi.org/10.1145/566654.566588>, <https://doi.org/10.1145/566654.566588>
3. Alliez, P., Ucelli, G., Gotsman, C., Attene, M.: Recent advances in remeshing of surfaces. *Shape analysis and structuring* pp. 53–82 (2008)
4. Alliez, P., Verdière, E.C.d., Devillers, O., Isenburg, M.: Isotropic surface remeshing. In: *Proceedings of the Shape Modeling International 2003*. p. 49. SMI '03, IEEE Computer Society, USA (2003)
5. Baieri, D., Maggioli, F., Rodolà, E., Melzi, S., Löhner, Z.: Implicit-arap: Efficient handle-guided neural field deformation via local patch meshing. *arXiv preprint arXiv:2405.12895* (2024)
6. Barda, A., Erel, Y., Kasten, Y., Bermano, A.H.: Roar: robust adaptive reconstruction of shapes using planar projections. *arXiv preprint arXiv:2307.00690* (2023)
7. Barda, A., Gadelha, M., Kim, V.G., Aigerman, N., Bermano, A.H., Groueix, T.: Instant3dit: Multiview inpainting for fast editing of 3d objects. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. pp. 16273–16282 (June 2025)
8. Barda, A., Kim, V., Aigerman, N., Bermano, A.H., Groueix, T.: Magicclay: Sculpting meshes with generative neural fields. In: *SIGGRAPH Asia 2024 Conference Papers*. SA '24, Association for Computing Machinery, New York, NY, USA (2024). <https://doi.org/10.1145/3680528.3687627>, <https://doi.org/10.1145/3680528.3687627>
9. Bhatia, R., Lawrence, K.L.: Two-dimensional finite element mesh generation based on stripwise automatic triangulation. *Computers & Structures* **36**(2), 309–319 (1990)
10. Botsch, M., Kobbelt, L.: A remeshing approach to multiresolution modeling. In: *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*. p. 185–192. SGP '04, Association for Computing Machinery, New York, NY, USA (2004). <https://doi.org/10.1145/1057432.1057457>, <https://doi.org/10.1145/1057432.1057457>
11. Chen, Q., Yang, Y., Wang, Y., Kim, V., Chaudhuri, S., Zhang, H., Chen, Z.: Art-deco: Arbitrary text guidance for 3d detailizer construction. In: *Proceedings of the SIGGRAPH Asia 2025 Conference Papers*. SA Conference Papers '25, Association for Computing Machinery, New York, NY, USA (2025). <https://doi.org/10.1145/3757377.3763877>, <https://doi.org/10.1145/3757377.3763877>
12. Chen, R., Chen, Y., Jiao, N., Jia, K.: Fantasia3d: Disentangling geometry and appearance for high-quality text-to-3d content creation. *arXiv preprint arXiv:2303.13873* (2023)
13. Chen, Z., Wang, F., Wang, Y., Liu, H.: Text-to-3d using gaussian splatting. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. pp. 21401–21412 (June 2024)
14. Decatur, D., Lang, I., Aberman, K., Hanocka, R.: 3d paintbrush: Local stylization of 3d shapes with cascaded score distillation. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. pp. 4473–4483 (June 2024)

15. Decatur, D., Lang, I., Hanocka, R.: 3d highlighter: Localizing regions on 3d shapes via text descriptions. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 20930–20939 (June 2023)
16. Dinh, N.A., Lang, I., Kim, H., Stein, O., Hanocka, R.: Geometry in style: 3d stylization via surface normal deformation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 28456–28467 (June 2025)
17. Donyach, M., Vanderhaeghe, D., Barthe, L., Botsch, M.: Adaptive remeshing for real-time mesh deformation. In: 34th Annual Conference of the European Association for Computer Graphics (Eurographics 2013). The Eurographics Association (2013)
18. Erkoç, Z., Gümeli, C., Wang, C., Nießner, M., Dai, A., Wonka, P., Lee, H.Y., Zhuang, P.: Predictor3d: Fast and precise 3d shape editing. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 640–649 (June 2025)
19. Gao, J., Liu, W., Sun, W., Wang, S., Song, X., Shang, T., Chen, S., Li, H., Yang, X., Yan, Y., et al.: Mars: Mesh autoregressive model for 3d shape detailization. arXiv preprint arXiv:2502.11390 (2025)
20. Gao, W., Aigerman, N., Groueix, T., Kim, V., Hanocka, R.: Textdeformer: Geometry manipulation using text guidance. In: ACM SIGGRAPH 2023 Conference Proceedings. SIGGRAPH '23, Association for Computing Machinery, New York, NY, USA (2023). <https://doi.org/10.1145/3588432.3591552>, <https://doi.org/10.1145/3588432.3591552>
21. Hao, Z., Romero, D.W., Lin, T.Y., Liu, M.Y.: Meshtron: High-fidelity, artist-like 3d mesh generation at scale. arXiv preprint arXiv:2412.09548 (2024)
22. Hu, Z., Zhao, M., Zhao, C., Liang, X., Li, L., Zhao, Z., Fan, C., Zhou, X., Yu, X.: Efficientdreamer: High-fidelity and robust 3d creation via orthogonal-view diffusion priors. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 4949–4958 (June 2024)
23. Jincheng, J., Cai, Y., Liu, L.: Craftmesh: High-fidelity generative mesh manipulation via poisson seamless fusion. arXiv preprint arXiv:2509.13688 (2025)
24. Jung, Y., Kim, H., Hwang, G., Baek, S.H., Lee, S.: Mesh density adaptation for template-based shape reconstruction. In: ACM SIGGRAPH 2023 Conference Proceedings. SIGGRAPH '23, Association for Computing Machinery, New York, NY, USA (2023). <https://doi.org/10.1145/3588432.3591498>, <https://doi.org/10.1145/3588432.3591498>
25. Khan, D., Plopski, A., Fujimoto, Y., Kanbara, M., Jabeen, G., Zhang, Y.J., Zhang, X., Kato, H.: Surface remeshing: A systematic literature review of methods and research directions. IEEE Transactions on Visualization and Computer Graphics **28**(3), 1680–1713 (2022). <https://doi.org/10.1109/TVCG.2020.3016645>
26. Kim, H., Lang, I., Groueix, T., Aigerman, N., Kim, V., Hanocka, R.: Meshup: Multi-target mesh deformation via blended score distillation. In: International Conference on 3D Vision 2025 (2025)
27. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
28. Lai, Z., Zhao, Y., Liu, H., Zhao, Z., Lin, Q., Shi, H., Yang, X., Yang, M., Yang, S., Feng, Y., et al.: Hunyuan3d 2.5: Towards high-fidelity 3d assets generation with ultimate details. arXiv preprint arXiv:2506.16504 (2025)
29. Laine, S., Hellsten, J., Karras, T., Seol, Y., Lehtinen, J., Aila, T.: Modular primitives for high-performance differentiable rendering. ACM Transactions on Graphics **39**(6) (2020)

30. Li, H., Erkoç, Z., Li, L., Sirigatti, D., Rosov, V., Dai, A., Nießner, M.: Meshpad: Interactive sketch-conditioned artist-reminiscent mesh generation and editing. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 16227–16237 (October 2025)
31. LI, W., Chen, R., Chen, X., Tan, P.: Sweetdreamer: Aligning geometric priors in 2d diffusion for consistent text-to-3d. In: Kim, B., Yue, Y., Chaudhuri, S., Fragkiadaki, K., Khan, M., Sun, Y. (eds.) International Conference on Learning Representations. vol. 2024, pp. 21567–21589 (2024), https://proceedings.iclr.cc/paper_files/paper/2024/file/5d5d2f63064a7da1336965bd027ea845-Paper-Conference.pdf
32. Li, W., Liu, J., Yan, H., Chen, R., Liang, Y., Chen, X., Tan, P., Long, X.: Craftsman3d: High-fidelity mesh generation with 3d native diffusion and interactive geometry refiner. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 5307–5317 (June 2025)
33. Liang, Y., Yang, X., Lin, J., Li, H., Xu, X., Chen, Y.: Luciddreamer: Towards high-fidelity text-to-3d generation via interval score matching. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 6517–6526 (June 2024)
34. Lin, C.H., Gao, J., Tang, L., Takikawa, T., Zeng, X., Huang, X., Kreis, K., Fidler, S., Liu, M.Y., Lin, T.Y.: Magic3d: High-resolution text-to-3d content creation (2023)
35. Lin, Z., Pathak, D., Li, B., Li, J., Xia, X., Neubig, G., Zhang, P., Ramanan, D.: Evaluating text-to-visual generation with image-to-text generation. arXiv preprint arXiv:2404.01291 (2024)
36. Liu, H.T.D., Gillespie, M., Chislett, B., Sharp, N., Jacobson, A., Crane, K.: Surface simplification using intrinsic error metrics. ACM Transactions on Graphics **42**(4) (2023)
37. Liu, H.T.D., Jacobson, A.: Normal-driven spherical shape analogies. In: Computer Graphics Forum. vol. 40, pp. 45–55. Wiley Online Library (2021)
38. Liu, H.T.D., Zhang, X., Yuksel, C.: Simplifying textured triangle meshes in the wild. ACM Transactions on Graphics (TOG) **44**(6), 1–16 (2025)
39. Liu, R., Wu, R., Van Hoorick, B., Tokmakov, P., Zakharov, S., Vondrick, C.: Zero-1-to-3: Zero-shot one image to 3d object. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 9298–9309 (October 2023)
40. Liu, T., Zhang, T., Chen, Y., Huang, Y., Wang, C.C.L.: Neural slicer for multi-axis 3d printing. ACM Trans. Graph. **43**(4) (Jul 2024). <https://doi.org/10.1145/3658212>, <https://doi.org/10.1145/3658212>
41. Maggioli, F., Baieri, D., Rodolà, E., Melzi, S.: Rematching: Low-resolution representations for scalable shape correspondence. In: Leonardis, A., Ricci, E., Roth, S., Russakovsky, O., Sattler, T., Varol, G. (eds.) Computer Vision – ECCV 2024. pp. 183–200. Springer Nature Switzerland, Cham (2025)
42. Michel, O., Bar-On, R., Liu, R., Benaim, S., Hanocka, R.: Text2mesh: Text-driven neural stylization for meshes. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 13492–13502 (2022)
43. Nicolet, B., Jacobson, A., Jakob, W.: Large steps in inverse rendering of geometry. ACM Trans. Graph. **40**(6) (Dec 2021). <https://doi.org/10.1145/3478513.3480501>, <https://doi.org/10.1145/3478513.3480501>
44. Palandra, F., Sanchietti, A., Baieri, D., Rodola, E.: Gsedit: Efficient text-guided editing of 3d objects via gaussian splatting. arXiv preprint arXiv:2403.05154 (2024)
45. Palfinger, W.: Continuous remeshing for inverse rendering. Computer Animation and Virtual Worlds **33**(5), e2101 (2022)

46. Poole, B., Jain, A., Barron, J.T., Mildenhall, B.: Dreamfusion: Text-to-3d using 2d diffusion. arXiv preprint arXiv:2209.14988 (2022), <https://arxiv.org/abs/2209.14988>
47. Praun, E., Hoppe, H.: Spherical parametrization and remeshing. *ACM Trans. Graph.* **22**(3), 340–349 (Jul 2003). <https://doi.org/10.1145/882262.882274>, <https://doi.org/10.1145/882262.882274>
48. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., Sutskever, I.: Learning transferable visual models from natural language supervision. arXiv preprint arXiv:2103.00020 (2021)
49. Rakotosaona, M.J., Aigerman, N., Mitra, N.J., Ovsjanikov, M., Guerrero, P.: Differentiable surface triangulation. *ACM Trans. Graph.* **40**(6) (Dec 2021). <https://doi.org/10.1145/3478513.3480554>, <https://doi.org/10.1145/3478513.3480554>
50. Seo, J., Jang, W., Kwak, M.S., Kim, I.H., Ko, J., Kim, J., Kim, J.H., Lee, J., Kim, S.: Let 2d diffusion model know 3d-consistency for robust text-to-3d generation. In: Kim, B., Yue, Y., Chaudhuri, S., Fragkiadaki, K., Khan, M., Sun, Y. (eds.) *International Conference on Learning Representations*. vol. 2024, pp. 32553–32572 (2024), https://proceedings.iclr.cc/paper_files/paper/2024/file/89ed365ef30f38efa088e48930238d0c-Paper-Conference.pdf
51. Sharp, N., Crane, K.: A laplacian for nonmanifold triangle meshes. *Computer Graphics Forum* **39**(5), 69–80 (2020). <https://doi.org/10.1111/cgf.14069>, <https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.14069>
52. Shewchuk, J.R.: What is a good linear finite element? interpolation, conditioning, anisotropy, and quality measures (preprint) (2002), <https://web.archive.org/web/20260217143453/https://people.eecs.berkeley.edu/~jrs/papers/elemj.pdf>, accessed February 17, 2026
53. Siddiqui, Y., Alliegro, A., Artemov, A., Tommasi, T., Sirigatti, D., Rosov, V., Dai, A., Nießner, M.: Meshgpt: Generating triangle meshes with decoder-only transformers. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. pp. 19615–19625 (June 2024)
54. Siddiqui, Y., Monnier, T., Kokkinos, F., Kariya, M., Kleiman, Y., Garreau, E., Gafni, O., Neverova, N., Vedaldi, A., Shapovalov, R., et al.: Meta 3d assetgen: Text-to-mesh generation with high-quality geometry, texture, and pbr materials. *Advances in Neural Information Processing Systems* **37**, 9532–9564 (2024)
55. Stability AI: Deepfloydif (2023), <https://github.com/deep-floyd/IF>, accessed September 19, 2025
56. Sun, J.M., Wu, T., Gao, L.: Recent advances in implicit representation-based 3d shape generation. *Visual Intelligence* **2**(1), 9 (2024)
57. Sundararaman, R., Donati, N., Melzi, S., Corman, E., Ovsjanikov, M.: Deformation recovery: Localized learning for detail-preserving deformations. *ACM Trans. Graph.* **43**(6) (Nov 2024). <https://doi.org/10.1145/3687968>, <https://doi.org/10.1145/3687968>
58. Surazhsky, V., Alliez, P., Gotsman, C.: Isotropic remeshing of surfaces: a local parameterization approach. Ph.D. thesis, INRIA (2003)
59. Suzuki, H., Sakurai, Y., Kanai, T., Kimura, F.: Interactive mesh dragging with adaptive remeshing technique. In: *Proceedings Pacific Graphics '98. Sixth Pacific Conference on Computer Graphics and Applications (Cat. No.98EX208)*. pp. 188–197 (1998). <https://doi.org/10.1109/PCCGA.1998.732112>

60. Tang, J., Li, M., Hao, Z., Liu, X., Zeng, G., Liu, M.Y., Zhang, Q.: Edgerunner: Auto-regressive auto-encoder for artistic mesh generation. In: Yue, Y., Garg, A., Peng, N., Sha, F., Yu, R. (eds.) International Conference on Learning Representations. vol. 2025, pp. 35913–35934 (2025), https://proceedings.iclr.cc/paper_files/paper/2025/file/58e6c003c9fb3992265005ff6aef1913-Paper-Conference.pdf
61. Wang, C., Chai, M., He, M., Chen, D., Liao, J.: Clip-nerf: Text-and-image driven manipulation of neural radiance fields. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 3835–3844 (June 2022)
62. Wang, D., Meng, H., Cai, Z., Shao, Z., Liu, Q., Wang, L., Fan, M., Zhan, X., Wang, Z.: Headevolver: Text to head avatars via expressive and attribute-preserving mesh deformation. In: 2025 International Conference on 3D Vision (3DV). pp. 211–221 (2025). <https://doi.org/10.1109/3DV66043.2025.00025>
63. Wang, H., Du, X., Li, J., Yeh, R.A., Shakhnarovich, G.: Score jacobian chaining: Lifting pretrained 2d diffusion models for 3d generation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 12619–12629 (June 2023)
64. Wang, Y., Yi, X., Weng, H., Xu, Q., Wei, X., Yang, X., Guo, C., Chen, L., Zhang, H.: Nautilus: Locality-aware autoencoder for scalable mesh generation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 10961–10970 (October 2025)
65. Wang, Z., Lu, C., Wang, Y., Bao, F., Li, C., Su, H., Zhu, J.: Prolificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation. *Advances in neural information processing systems* **36**, 8406–8441 (2023)
66. Worchel, M., Diaz, R., Hu, W., Schreer, O., Feldmann, I., Eisert, P.: Multi-view mesh reconstruction with neural deferred shading. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 6187–6197 (June 2022)
67. Xiang, J., Lv, Z., Xu, S., Deng, Y., Wang, R., Zhang, B., Chen, D., Tong, X., Yang, J.: Structured 3d latents for scalable and versatile 3d generation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 21469–21480 (June 2025)
68. Xu, J., Cheng, W., Gao, Y., Wang, X., Gao, S., Shan, Y.: Instantmesh: Efficient 3d mesh generation from a single image with sparse-view large reconstruction models. *arXiv preprint arXiv:2404.07191* (2024)
69. Yan, X., Lee, H.H., Wan, Z., Chang, A.X.: An object is worth 64×64 pixels: Generating 3d object via image diffusion. In: 2025 International Conference on 3D Vision (3DV). pp. 123–133. IEEE (2025)
70. Yang, H., Chen, Y., Pan, Y., Yao, T., Chen, Z., Wu, Z., Jiang, Y.G., Mei, T.: Dreammesh: Jointly manipulating and texturing triangle meshes for text-to-3d generation. In: European Conference on Computer Vision. pp. 162–178. Springer (2024)
71. Yoo, S., Kim, K., Kim, V.G., Sung, M.: As-plausible-as-possible: Plausibility-aware mesh deformation using 2d diffusion priors. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4315–4324 (2024)
72. Yu, Q., Li, X., Tang, Y., Han, X., Hu, L., Hao, Y., Chen, M.: Fancy123: One image to high-quality 3d mesh generation via plug-and-play deformation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 595–604 (June 2025)

73. Zhang, B., Ren, J., Wonka, P.: Geometry distributions. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 1495–1505 (October 2025)
74. Zhang, B., Tang, J., Nießner, M., Wonka, P.: 3dshape2vecset: A 3d shape representation for neural fields and generative diffusion models. *ACM Trans. Graph.* **42**(4) (Jul 2023). <https://doi.org/10.1145/3592442>, <https://doi.org/10.1145/3592442>
75. Zhang, J., Li, X., Wan, Z., Wang, C., Liao, J.: Text2nerf: Text-driven 3d scene generation with neural radiance fields. *IEEE Transactions on Visualization and Computer Graphics* **30**(12), 7749–7762 (2024). <https://doi.org/10.1109/TVCG.2024.3361502>
76. Zhang, L., Wang, Z., Zhang, Q., Qiu, Q., Pang, A., Jiang, H., Yang, W., Xu, L., Yu, J.: Clay: A controllable large-scale generative model for creating high-quality 3d assets. *ACM Trans. Graph.* **43**(4) (Jul 2024). <https://doi.org/10.1145/3658146>, <https://doi.org/10.1145/3658146>
77. Zhao, R., Ye, J., Wang, Z., Liu, G., Chen, Y., Wang, Y., Zhu, J.: Deepmesh: Autoregressive artist-mesh creation with reinforcement learning. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 10612–10623 (October 2025)
78. Zhu, J., Zhuang, P., Koyejo, S.: Hifa: High-fidelity text-to-3d generation with advanced diffusion guidance. In: Kim, B., Yue, Y., Chaudhuri, S., Fragkiadaki, K., Khan, M., Sun, Y. (eds.) *International Conference on Learning Representations*. vol. 2024, pp. 5762–5786 (2024), https://proceedings.iclr.cc/paper_files/paper/2024/file/178ae4ba29022eb7bf509c2e27bc8ab8-Paper-Conference.pdf