DiffSimp：用于基于梯度的低多边形建模的可微网格简化

Seonghun Oh
Holiday Robotics

seonghun.oh@holiday-机器人学.com

arXiv:2611.31018v1 [cs.GR] 23 Nov 2026

摘要

可微网格提取能够针对各种任务，对精细的三维几何进行基于梯度的优化。然而，紧凑网格通常通过后处理简化获得，使简化引入的误差被排除在上游优化之外。我们提出 DIFFSIMP，一种无需训练、即插即用的可微网格简化层。该层基于基于 *QEM* 的并行边折叠构建，使现有几何优化流水线能够优化其最终的简化网格。我们将固定折叠历史下的简化形式化为局部可微折叠映射，从而无需放宽离散折叠决策即可实现梯度传播。我们将快速且高质量的并行简化，与利用可微顶点二次误差和二次误差归约森林实现高效梯度传播的因子化反向公式相结合。在低多边形网格重建和逆向渲染任务上的实验表明，与后处理简化相比，该方法具有更高的保真度；随着顶点预算更加紧张，梯度传播带来的额外收益也会增加。DIFFSIMP 支持多种几何优化前端，在所评估的流水线中，其端到端运行时间为未简化基线的 *1.00–1.25*× 倍。

1. 引言

三角网格仍然是计算机图形学、三维视觉和机器人学中的基础曲面表示。近年来，可微网格提取 [1, 44, 46, 47, 56] 通过将基于网格目标函数的梯度向上游传播，实现了基于梯度的几何优化，支持逆向渲染 [24, 35, 59],场景重建 [10, 19], 以及生成式三维建模 [2, 13, 57]。然而，这些流水线通常会生成稠密网格以捕捉精细的几何结构，而下游应用可能需要紧凑网格。

获得紧凑网格的一种直接方法是降低可微网格提取的分辨率，但这会限制优化过程中对精细几何细节的恢复。相反，一种常见策略是重建或生成足够详细的网格，然后通过后处理简化来满足下游的复杂度要求。在网格简化方法中，
在这些方法中，二次误差度量（QEM）边折叠 [16, 17] 已成为一种实用的现成选择 [27, 31]。它能够自然地根据局部几何调整网格复杂度，在限制几何近似误差的同时实现激进简化，而无需像变分方法 [9, 58]或预训练简化模型 [42]那样进行单独的迭代拟合。这些优势使 QEM 成为现代三维处理流程中的常见后处理步骤 [7, 15, 52, 55]。

然而，后处理网格简化使几何优化与最终简化网格相脱节。尽管可微网格提取能够将基于网格的目标函数的梯度向上游传播，但这些目标函数通常是在详细的中间网格上计算，而非最终部署的简化网格上计算。在激进简化的情况下，这种不匹配会造成尤为严重的影响：随着精细结构被逐渐更粗糙的网格近似，即使是二次误差度量（QEM）也可能引入显著的几何失真。这些由简化引起的误差无法为上游优化器提供梯度信号，因而无法让它直接补偿这些误差，以改善最终部署的网格。

为解决这种优化–部署不匹配问题，我们希望将网格简化作为可微层整合到基于梯度的几何优化流程中。为实现实际集成，我们确定了三项设计优先事项。首先，该层应当是无需训练的且原生支持网格的，不依赖于学习得到的简化先验和特定的上游表示，从而能够应用于各种流水线和下游目标。其次，必须保持较高的前向简化质量。在限制较少的预算下，简化通常会引入较小的几何误差，为基于梯度的补偿留下的空间有限，因此前向质量尤为重要。在更严格的预算下，高质量的前向简化与基于梯度的补偿将变得相辅相成，共同维持几何保真度。第三，高效的前向和反向执行至关重要，因为简化必须在每次优化迭代中执行。

，共同维持几何保真度。第三，高效的前向和反向执行至关重要，因为简化必须在每次优化迭代中执行。

为满足这些需求，我们提出了 DIFFSIMP，这是一种无需训练、可直接插入的可微网格简化层，基于二次误差度量的并行边折叠构建，使现有的基于梯度的几何优化流程能够通过简化后的网格进行优化。尽管离散的边折叠决策使全局可微性难以实现，我们仍利用记录的折叠历史，将简化表示为局部可微折叠映射，从而无需放宽这些决策即可实现梯度传播。为实现快速且高质量的前向简化，我们扩展了 PaMO [38], 的并行边折叠方案，在保持执行效率的同时提升简化质量。为实现高效的反向传播，我们提出一种因子化微分方案，将顶点二次型视为可微中间变量，从而实现并行的、局部于折叠的向量-雅可比积（VJP），并利用二次误差约简森林结构，避免冗余的面级微分。我们在 CUDA 中实现了这两种操作，并提出渐进式顶点预算调度，以便将其集成到迭代优化中。

在低多边形网格重建和逆向渲染任务上的实验表明，与后处理相比，通过 DIFF-SIMP 进行优化能提升两项任务中最终简化网格的保真度。我们还表明，梯度传播能带来额外收益，且顶点预算越紧张，收益越大，因为上游优化能够补偿简化引入的误差。我们进一步证明，该方法兼容多种现有的基于梯度的几何优化流水线。实现这些改进仅需相应未简化流水线端到端优化时间的 1.00–1.25×

2. 相关工作

2.1. 基于梯度的网格优化

可微网格提取 [1, 20, 21, 44, 46, 47, 56]和可微网格表示 [43, 49, 50] 能够从基于网格的目标向上游传播梯度，支持多种下游应用 [10,13, 19, 35, 37, 57, 59]。然而，提取网格的复杂度仍受底层提取结构的限制。即使是自适应提取方法 [1, 47]，调整的也是空间采样和提取结构，而非执行激进的几何自适应网格简化。基于梯度的低多边形网格优化也已有探索，包括几何和外观细化 [7, 23], 、联合基线网格与位移优化 [11], ，以及自适应重新网格化或拓扑更新 [12, 14, 39]。然而，这些方法要么在给定或重建的几何上优化紧凑网格，却不通过简化向上游表示传播梯度；要么在专门的优化流水线中调整网格连接关系。相比之下，我们可复用的可微网格简化层能够将来自激进简化网格上目标的梯度传播至上游表示，而无需重新设计现有优化流水线。

2.2. 网格简化

网格简化已通过顶点聚类 [45], 变分近似 [9, 58],重新网格化 [6, 53, 54], 和迭代边折叠 [25, 30]得到了广泛研究。这些无需训练、适用于网格的方法被广泛用作现代三维处理流程中的后处理步骤。其中，二次误差度量（QEM）边折叠 [16, 17] 已成为网格简化的标准方法。为克服顺序贪心边折叠的计算开销，并行方法 [38, 40] 在保持具有竞争力的几何质量的同时加速了简化，但其效果仍略逊于顺序 QEM。然而，这些方法无法通过坍缩操作传播梯度，因此简化在基于梯度的流水线中仍主要作为后处理步骤。基于 PaMO [38], , 我们将 QEM 的几何自适应性与并行执行的效率同固定历史微分相结合，使其能够应用于基于梯度的优化。我们进一步提升了并行简化的质量，以便在大幅简化时更好地保留几何保真度，同时维持高效执行。

2.3. 基于学习的低多边形建模

基于学习的网格简化和低多边形建模方法包括任务驱动的网格池化 [22], 学习型简化算子 [42], 学习型几何表示 [33], 渐进式网格表示 [3, 5, 28], 以及生成式网格先验 [4, 29, 32, 48, 51]。虽然这些方法支持网格简化和低多边形建模，但它们也依赖于学习参数和训练分布。值得注意的是，Neural Mesh Simplification [42]通过学习型顶点采样和连接性预测实现了可微分简化，但仍依赖于预训练网络。相比之下，我们的无需训练的可微分层保留了二次误差度量的几何自适应性，而无需依赖学习型简化先验。

3. 预备知识

3.1. 边折叠的二次误差度量

二次误差度量（QEM）简化通过连续执行边折叠，迭代地简化三角网格。设 $\mathcal{M}^{(0)} = (V_{(0)}\, F^{(0)})$ 表示一个输入三角网格，其顶点为 $V^{(0)} = \{v_i^{(0)}\}_{i=1}^{N}$、$v_i^{(0)} \in \mathbb{R}^3$，面为 $F^{(0)}$。简化过程如下： $\mathcal{M}^{(0)} \xrightarrow{\mathcal{E}_1} \ldots \xrightarrow{\mathcal{E}_R} \mathcal{M}^{(R)}$ ，其中 $\mathcal{E}_r$ 表示已坍缩的边集合

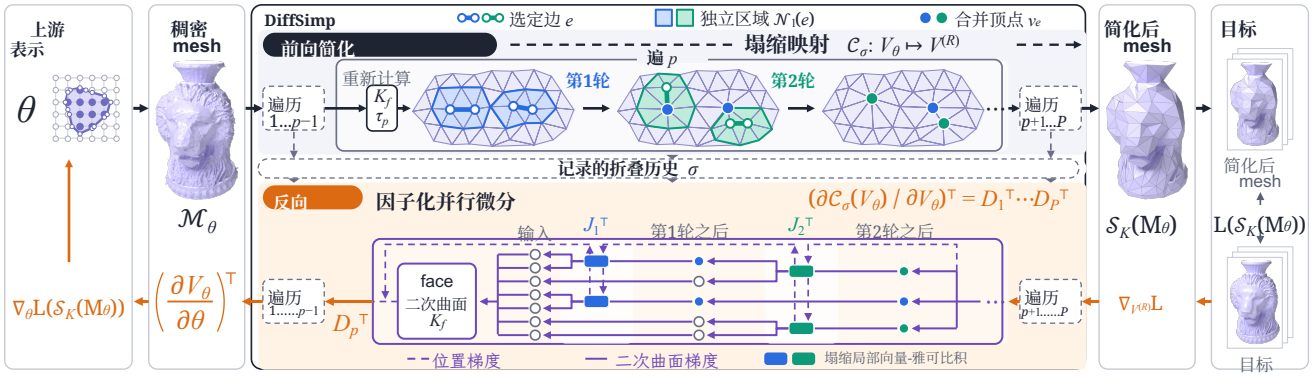


图1. DIFFSIMP 概览。给定稠密网格 $\mathcal{M}_\theta = (V_\theta, F_\theta)$，DIFFSIMP 在基于梯度的几何优化流程中，将简化网格 $\mathcal{S}_K(\mathcal{M}_\theta)$ 上目标函数的梯度传播至上游参数 θ。前向（上）：并行边折叠轮次、渐进式代价阈值（τ_p）以及逐轮二次误差矩阵重计算生成简化网格，同时记录折叠历史 σ（第 4.1 节）。反向（下）：在固定 σ 的情况下，我们对塌缩映射 $\mathcal{C}_\sigma$ 求微分（第 4.2 节）。对于每一遍，$D_p^\top$ 通过因子化并行微分（第 4.3 节）计算得出。该方法将每轮中的并行塌缩局部向量-雅可比积（形成 $J_r^\top$）与通过归约森林进行的二次曲面梯度传播相结合，使面二次型 K_f 在每一遍中只需微分一次。

在第 r 轮中。在经典串行二次误差度量中，每轮都会贪心地坍缩最小代价边 $e = (i,\ j)$，将其端点 v_i 和 v_j 合并为 v_e，更新网格连通性和相邻边的代价。

在第一轮之前，每个面 $f \in F^{(0)}$ 定义一个面二次误差 $K_f = w_f p_f p_f^\top \in \mathbb{R}^{4\times4}$，其中 $p_f = (n_f^\top, d_f)^\top$表示面 f的支撑平面，其单位法向量为n_f，偏移量为 d_f，而 w_f 是其权重。每个顶点 v_i 根据其关联面被赋予一个顶点二次误差 $Q_i = \sum_{f\ni i} K_f$。为评估边 $e = (i, j)$的坍缩误差，我们首先将其两个端点的顶点二次误差合并：

$$Q_e = Q_i + Q_j = \begin{bmatrix} A_e & b_e \\ b_e^\top & c_e \end{bmatrix}, \tag{1}$$

其中 $A_e \in \mathbb{R}^{3\times3}$、$b_e \in \mathbb{R}^3$和 $c_e \in \mathbb{R}$。位置 v 处的坍缩误差为 $E_e(v) = \tilde{v}^\top Q_e \tilde{v}$，其中 $\tilde{v} = (v^\top, 1)^\top$。当 A_e 的条件足够良好时，最优位置为 $v_e = \mathrm{argmin}_{v\in\mathbb{R}^3} E_e(v) = -A_e^{-1} b_e$；否则使用边线段极小值点或中点回退。坍缩代价为 $C_e = E_e(v_e)$。坍缩边e随后会更新其端点位置和顶点二次误差，即 $(v_i, Q_i), (v_j, Q_j) \mapsto (v_e, Q_e)$。

3.2. 并行边折叠

并行边折叠会在单个轮次中折叠多个互不冲突的边。PaMO [38] 通过独立区域选择实现了这一点。对于每条边 $e = (i,\ j)$，令$\mathcal{N}_1(e) = \mathcal{N}_1(v_i) \cup \mathcal{N}_1(v_j)$表示其一环面邻域，其中 $\mathcal{N}_1(v_i)$包含与 v_i相邻的面。每条边都会将其折叠代价 C_e 传播到$\mathcal{N}_1(e)$中的所有面，每个面选择一个最小代价候选项。如果某条边 e 在 $\mathcal{N}_1(e)$中的每个面上都是最小代价候选项，则该边会被选中。由于折叠 e仅修改 $\mathcal{N}_1(e)$内的网格几何和拓扑，因此该邻域定义了其独立区域。由于每个面只选择一个候选项，各独立区域在面层面互不相交，从而能够进行无冲突的并行折叠。在每个轮次中，满足 $C_e \le \tau$ 的选定边会同时进行折叠，其中 τ表示代价阈值。虽然并行执行提高了效率，但与顺序 QEM 简化相比，可能会牺牲近似质量。

4. 方法

我们引入 DIFFSIMP （图1），这是一种可微网格简化层，基于基于 QEM 的并行边折叠构建。给定一个参数化为 θ 的上游网格生成器，令$\mathcal{M}_\theta = (V_\theta,\ F_\theta)$ 表示其输出网格。我们将优化目标表述为

$$\min_\theta \ \mathcal{L}\left(\mathcal{S}_K(\mathcal{M}_\theta)\right), \tag{2}$$

其中，$\mathcal{S}_K$ 表示 DIFFSIMP 的前向简化算子，其目标顶点预算为 K。

4.1. 前向简化

对于前向简化，我们扩展第 3.2 节中的并行边坍缩方案，采用第 3.1 节中的 QEM 放置、坍缩代价和二次误差度量更新规则。为在保持并行执行效率的同时提高简化质量，我们在各次简化遍次中逐步增加代价阈值，直到达到目标顶点预算 K。每一遍 p都会在代价阈值 τ_p 下执行并行坍缩轮次，直到无法再坍缩任何边。如果未达到目标预算 K，则将阈值增加至 τ_{p+1}。按照无记忆简化的方式，[26, 30], 我们会重新计算面和顶点

二次曲面，以反映更新后的网格几何。我们还使用面积加权面二次曲面$K_f = w_f p_f p_f^\top$，其中 $w_f = a_f/\bar{a}$，其中 a_f表示面面积，而 $\bar{a}$表示每一遍开始时的平均面面积。[1] 为保证数值稳定性，对于条件良好的 $v_e = -(A_e+\lambda_e I)^{-1} b_e$，我们采用正则化 QEM 放置 A_e，其中 $\lambda_e = \eta$最大值$_k |(A_e)_{kk}|$，且 η 为一个较小的系数。否则，将使用边线段或中点回退，从而产生三种可能的放置分支。

4.2. 固定历史微分

边折叠涉及离散的边选择和连接关系更新，因此总体简化过程并非全局可微。然而，在给定固定的折叠历史后，输出连通性便已确定，而输出顶点位置则是通过局部可微映射由输入顶点位置得到的。我们利用这一差异，在前向简化过程中保留离散的边选择和连接关系更新，同时在反向过程中对得到的几何进行微分。

令 $\mathcal{M}^{(0)} = \mathcal{M}_\theta$，并令 $\sigma(V^{(0)} F^{(0)})$ 表示离散的折叠历史，记录已折叠边集$\{\mathcal{E}_r\}_{r=1}^{R}$、遍次边界、保留和移除的端点索引，以及每次折叠的放置分支。在σ 固定的情况下，顶点放置、二次曲面加法和逐轮二次误差矩阵重计算共同构成一个局部可微的塌缩映射 $\mathcal{C}_\sigma$： $V^{(0)} \mapsto V^{(R)}$。因此，我们可以将前向简化算子表示为

$$\mathcal{S}_K(\mathcal{M}^{(0)}) = \mathcal{M}^{(R)} = \left(\mathcal{C}_\sigma(V^{(0)}), F^{(R)}\right). \tag{3}$$

将式（3）代入式（2）后，损失变为$\mathcal{L}(\mathcal{C}_\sigma(V_\theta)\ F^{(R)})$。其中 σ保持不变， $F^{(R)}$ 固定，对损失通过输出顶点位置进行微分可得

$$\nabla_\theta \mathcal{L}(\mathcal{S}_K(\mathcal{M}_\theta)) = \left(\frac{\partial V_\theta}{\partial \theta}\right)^\top \left(\frac{\partial \mathcal{C}_\sigma(V_\theta)}{\partial V_\theta}\right)^\top \nabla_{V^{(R)}} \mathcal{L}. \tag{4}$$

塌缩映射雅可比矩阵 $\frac{\partial \mathcal{C}_\sigma(V_\theta)}{\partial V_\theta}$ 捕捉了连续塌缩之间的依赖关系，使梯度能够传递到在简化过程中被移除的输入顶点。

4.3. 因子化并行微分

高效计算向量-雅可比积 $\left(\frac{\partial \mathcal{C}_\sigma(V_\theta)}{\partial V_\theta}\right)^\top \nabla_V)^{(R)} \mathcal{L}$ 在公式（4）中需要处理塌缩映射雅可比矩阵中的几何依赖关系，包括每个轮次内部以及连续轮次之间的依赖关系。在每个轮次内部，独立区域$\{\mathcal{N}_1(e)\}_{e\in\mathcal{E}_r}$ 彼此面不相交，但可能共享相邻顶点，从而通过面二次型产生重叠依赖关系。在不同轮次之间，坍缩依赖于此前更新后的顶点位置和累积的顶点二次型。我们通过将顶点二次型视为可微中间变量，并将每次坍缩与其底层面几何分离，来处理这些依赖关系。这样即可并行计算一个轮次内多次坍缩的梯度。在不同轮次之间，顶点二次曲面梯度沿归约森林中的加性二次曲面合并进行传播，从而避免冗余微分。

设 P表示简化遍次的数量， V_p 表示第 p 遍的输出顶点位置，其中 $V_0 = V^{(0)} = V_\theta$，并设 $\mathbf{Q}$ 为顶点二次型的集合 Q_i。在第 p 遍开始时，根据输入几何 V_{p-1}重新计算顶点二次型 $\mathbf{Q}^{p-1}$ 。对于具有 R_p 个坍缩轮次的第 p 遍，我们使用局部轮次索引，并省略遍次索引。令 $X^{(r)} = (V^{(r)}, \mathbf{Q}^{(r)})$ 表示第 r 轮后的状态，其初始化为 $X^{(0)} = (V_{p-1}, \mathbf{Q}^{p-1})$，并令 $J_r = \partial X^{(r)}/\partial X^{(r-1)}$ 表示其逐轮雅可比矩阵。第 p 遍的雅可比矩阵可因子化为

$$D_p := \frac{\partial V_p}{\partial V_{p-1}} = \begin{bmatrix} I & 0 \end{bmatrix} J_{R_p} \cdots J_1 \begin{bmatrix} I \\ \dfrac{\partial \mathbf{Q}_{p-1}}{\partial V_{p-1}} \end{bmatrix}, \tag{5}$$

其中，最左侧因子用于选取输出顶点位置，最右侧因子则对遍次初始化进行微分，包括二次型重计算。于是，完整的塌缩映射雅可比矩阵为 $\frac{\partial \mathcal{C}_\sigma(V_\theta)}{\partial V_\theta} = D_P \cdots D_1$。

并行雅可比结构。 对于遍次 p 中的 $e = (i, j) \in \mathcal{E}_r$，令 $J_e^{(r)}$ 表示塌缩更新的雅可比矩阵 $(v_i^{(r-1)}, Q_i^{(r-1)}, v_j^{(r-1)}, Q_j^{(r-1)}) \mapsto (v_e^{(r)}, Q_e^{(r)})$ 。由于每次塌缩所涉及的由面导出的几何贡献会累积到其端点的二次曲面中，将这些二次曲面视为可微中间变量后， $J_e^{(r)}$ 仅依赖于其两个端点状态。由于已塌缩边所在区域面不相交 $\mathcal{N}_1(e)$ ，因此端点也互不相交，逐轮雅可比矩阵由彼此独立的塌缩局部块构成：

$$J_r = \left(\bigoplus_{e\in\mathcal{E}_r} J_e^{(r)}\right) \oplus I, \tag{6}$$

在置换意义下，其中 $\oplus$表示矩阵直和， I 表示未改变顶点状态的单位块。因此，逐轮向量-雅可比乘积可分解为相互独立的塌缩局部向量-雅可比积，并行计算时不会对端点梯度产生写入冲突。

二次型归约森林反向传播。 尽管逐轮雅可比矩阵支持并行局部微分，但累积的顶点二次型保留了连续塌缩之间的依赖关系。由于顶点二次型

[1] 微分时省略了 $\bar{a}$ 的归一化，因为均匀的二次曲面缩放不会影响顶点放置。

是以加法方式合并的，记录的合并历史形成了一片归约森林，其叶节点是初始顶点二次型。我们通过这片森林反向传播二次曲面梯度，直到抵达叶节点后才进行面级微分。这样可以避免在每次塌缩时反复对构成该二次型的面几何进行微分。

令 $g^{(r)} = \nabla_{X^{(r)}}\mathcal{L}$ 表示相对于第 r 轮之后状态的梯度。由于损失仅通过 V_p 依赖于终止状态 $X^{(R_p)}$，我们初始化 $g^{(R_p)} = (\nabla_{V_p}\mathcal{L}, 0)$，并按轮次的逆序应用 $g^{(r-1)} = J_r^\top g^{(r)}$。放置导数会生成二次曲面梯度，并通过更早的合并进行传播。对于 $e = (i, j) \in \mathcal{E}_r$，令 $g_{v_e}^{(r)}$ 和 $g_{Q_e}^{(r)}$ 表示其在合并位置和二次曲面处接收的梯度。由于 $Q_e^{(r)} = Q_i^{(r-1)} + Q_j^{(r-1)}$，两个输入顶点二次型接收到相同的梯度：

$$g_{Q_i}^{(r-1)} = g_{Q_j}^{(r-1)} = g_{Q_e}^{(r)} + \left(\frac{\partial v_e^{(r)}}{\partial Q_e^{(r)}}\right)^\top g_{v_e}^{(r)}. \tag{7}$$

第一项携带来自后续坍缩的二次曲面梯度，而第二项则加入当前顶点放置所产生的贡献。

通过合并历史传播梯度可得到 $g^{(0)}$，即将 V_{p-1} 和$\mathbf{Q}^{p-1}$ 视为独立状态输入时的梯度。随后，我们对本遍的二次型重计算求导，以获得相对于输入几何的完整梯度。由于 $Q_i^{(0)} = \sum_{f\ni i} K_f$，

$$\begin{aligned} g_{K_f} &= \sum_{i\in f} g_{Q_i}^{(0)}, \\ \nabla_{v_i^{(0)}}\mathcal{L} &= g_{v_i}^{(0)} + \sum_{f\ni i}\left(\frac{\partial K_f}{\partial v_i^{(0)}}\right)^\top g_{K_f}. \end{aligned} \tag{8}$$

这将直接位置梯度与延迟的二次曲面诱导梯度相结合，得到逐遍的向量-雅可比积$D_p^\top \nabla_{V_p}\mathcal{L} = \nabla_{V_{p-1}}\mathcal{L}$。按照遍次的逆序应用这些逐遍向量-雅可比积，即可得到公式（4）中的完整坍缩映射向量-雅可比积，其中包括对每个遍次边界处二次曲面重新计算过程的微分。

在每一遍中，每次坍缩都需要进行一个常数规模的局部反向更新，而每个初始面只需进行一次微分。因此，这些反向更新所需的工作量为$O(\sum_{r=1}^{R_p}|\mathcal{E}_r| + |F_{p-1}|)$，其中 F_{p-1} 是第 p遍的输入面集。即使仅保留输入顶点的 0.5%，简化平均也需要约四遍。我们的 CUDA 实现执行这些更新时，无需构建逐次坍缩的 PyTorch 自动微分图 [41]。用于传播位置梯度的放置导数见附录 B。

4.4. 集成到基于梯度的优化中

在底层几何尚未得到充分优化之前应用激进的简化，或突然施加目标顶点预算，可能会导致显著的几何变化并使优化过程不稳定。因此，我们提出了一种渐进式顶点预算调度方案，用于将简化整合到基于梯度的几何优化中。令 t 表示优化迭代的索引，并令 $N_t = |V_\theta| = |V^{(0)}|$ 表示第 t 次迭代时的输入顶点数量。给定预热时长 t_w 和预算缩减时长 t_r，我们定义

$$s_t = \begin{cases} 0, & t < t_w, \\ \dfrac{t - t_w}{t_r}, & t_w \le t < t_w + t_r, \\ 1, & t \ge t_w + t_r. \end{cases} \tag{9}$$

迭代 t 时的顶点预算为

$$K_t = \mathrm{round}\left(N_t^{1-s_t} K^{s_t}\right), \tag{10}$$

其中 round(·) 表示四舍五入到最接近的整数。预热后，顶点预算在对数空间中从 N_t 插值至 K，随后在目标预算下进行优化。在每次迭代中，我们将 $\mathcal{S}_{K_t}$ 应用于当前网格。

尽管折叠历史可能在优化迭代之间发生变化，但我们通过实验证明，在每个实际产生的历史下对塌缩映射求导，能够支持有效的优化（第 5.3节）。

5. 实验

我们在低多边形网格重建和逆向渲染任务上评估 DIFFSIMP 。在这两项任务中，我们遵循第 4.4 节中的渐进式顶点预算调度方案，其中预热时长为 $t_w = 0.2T$，预算缩减时长为 $t_r = 0.45T$，其中 T 表示优化迭代的总次数。所有实验均在单张 RTX 5090 上进行。有关详细的实验设置，请参见附录 A。

5.1. 低多边形网格重建

实验设置。 我们使用 Myles 等人提供的 79 个形状 [36], , 排除非水密形状和非常薄的形状 [47]。我们的基础重建流程在归一化至适合 $[-0.9, 0.9]^3$的形状上，以网格分辨率 $n = 128$ 优化 FlexiCubes [47]，迭代 5,000 次，并采用多视图轮廓、深度和法线监督。基线方法包括低分辨率 FlexiCubes（Low-res）：使用每个形状最精细的网格，顶点数最多为 K；还包括应用于基础重建结果的代表性网格简化方法：二次误差度量（QEM） [17]（MeshLab [8]）、PaMO [38], 和顶点聚类 [45], , 以及重新网格化方法 ACVD [53]。我们还与 CWF [58], 进行比较，后者优化简化几何体；以及 TriFlow [29], , 一种基于学习的方法。不同于后处理，我们将 DIFFSIMP 集成到基础流程中，以通过简化网格进行优化。

我们评估最大连通分量，以排除基于图像的监督无法观察到的非连通内部几何体 [1]。我们报告平方倒角距离（CD）

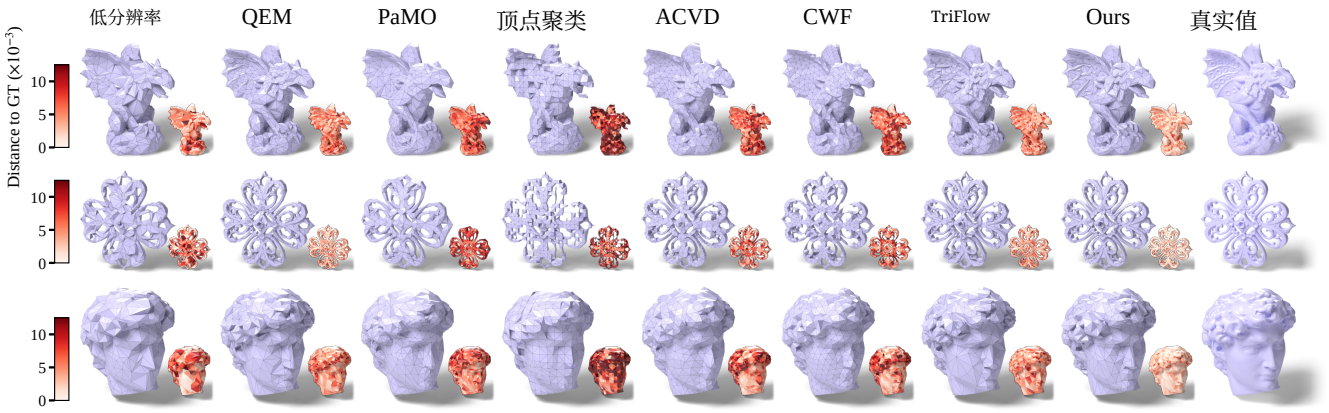


图 2. 在 $K = 1000$ 处对应表 1 的定性比较。

和 F1 分数（10^{-3} 阈值），使用 10^6 个表面采样点进行计算，同时报告法线一致性（NC）、自相交率（SI）和瘦长三角形率（SR）。CD 按 10^5, 缩放，而 SR 是最小角低于 1°的三角形所占百分比。

与低多边形建模方法的比较。 表 1 和图 2比较了低多边形建模方法在$K = 1000$处的表现。我们的方法在倒角距离、F1 分数和法向一致性方面均优于所有基线方法，同时保持了与二次误差度量相当的简化率和简化比。这些结果证明，与直接进行低分辨率重建和后处理简化相比，通过简化进行优化是有效的。我们进一步在顶点预算 $K \in \{2000, 1500, 1000, 500, 250\}$ 下、$n = 128$处评估重建效果，同时保持其余设置不变。为区分前向简化与微分的贡献，我们还将我们的前向算子作为后处理进行评估（我们的方法，仅前向）。如表2 和图 3所示，通过简化进行微分能够持续提升重建保真度，相较于仅前向变体表现更佳。在较大的预算下，相比二次误差度量，大部分改进来自前向简化器，因为简化引入的误差较小，留给梯度校正的空间很少。在更紧的预算下，这些误差会增大，梯度能够补偿其中更多误差，因此微分带来的绝对倒角距离增益从0.05 处的 $K = 2000$ 提升至 1.34 处的 $K = 250$。

几何优化流程之间的集成。我们通过将 FlexiCubes 替换为 DiffMC [56], , 将 DIFFSIMP 集成到 DMTet [46], 的独立四面体重建管线中，并将其应用于 Large Steps [37], 来评估 DIFFSIMP 在 FlexiCubes 之外的表现。Large Steps 通过微分坐标优化显式网格，无需进行等值面提取。三者均使用与基础重建流程相同的多视图监督。我们使用 $K = 1000$（DMTet 和 DiffMC 使用$n = 128$）以及这 42 个零亏格

表 1。使用 $n = 128$、 $K = 1000$进行低多边形重建。Reference 表示未简化的基础重建。CD_{mean} 和 CD_{med} 分别表示平均 CD 和中位数 CD（$\times 10^{-5}$）。

方法	倒角距离$_{\text{mean}}$ ↓	倒角距离$_{\text{med}}$ ↓	F1↑	法向一致性↑	简化率(%)↓	简化比(%)↓	$\lvert V\rvert$
参考	0.47	0.39	0.45	0.99	0.70	0.65	42 ,195
低分辨率 [47]	338.22	9.02	0.20	0.93	6.41	0.93	924
二次误差度量 [17]	1.92	1.11	0.21	0.96	0.56	0.02	989
PaMO [38]	4.96	2.90	0.12	0.95	**0.00**	**0.00**	988
顶点聚类 [45]	42.99	17.76	0.06	0.89	10.02	0.01	995
ACVD [53]	4.35	2.62	0.17	0.95	0.64	0.14	1,047
CWF [58]	7.10	5.15	0.11	0.94	0.02	0.02	1,000
TriFlow [29]	2.53	1.20	0.19	0.96	2.67	0.37	989
Ours	**1.26**	**0.76**	**0.26**	**0.97**	0.62	0.05	1 ,000

表 2。在 $n = 128$（倒角距离 $\times 10^{-5}$）下不同顶点预算的重建结果。我们的方法（仅前向）将我们的简化器作为后处理应用。 Δ_{grad} 表示相对于该变体的绝对增益，正值表示有所改进。

K	指标	二次误差度量	ACVD	CWF	TriFlow	Ours（仅前向）	我们的方法	Δ_{grad} ↑
2000	倒角距离↓	0.90	1.96	3.25	0.95	0.75	**0.70**	+0.05
	F1↑	0.30	0.25	0.18	0.27	0.31	**0.34**	+0.03
1500	倒角距离↓	1.18	2.69	4.42	1.29	0.95	**0.87**	+0.08
	F1↑	0.26	0.22	0.15	0.23	0.28	**0.31**	+0.03
1000	倒角距离↓	1.92	4.35	7.10	2.53	1.53	**1.26**	+0.27
	F1↑	0.21	0.17	0.11	0.19	0.23	**0.26**	+0.03
500	倒角距离↓	5.10	10.72	26.41	6.78	3.97	**2.89**	+1.08
	F1↑	0.14	0.10	0.06	0.12	0.16	**0.18**	+0.03
250	倒角距离↓	14.88	36.59	111.05	22.87	12.10	**10.76**	+1.34
	F1↑	0.09	0.06	0.04	0.07	0.10	**0.13**	+0.03

Large Steps 的形状。表 3 显示，在这三种流水线中，DIFFSIMP 的倒角距离和 F1 分数均优于所有基线方法。这些结果表明，通过简化进行优化，能够持续提升低多边形重

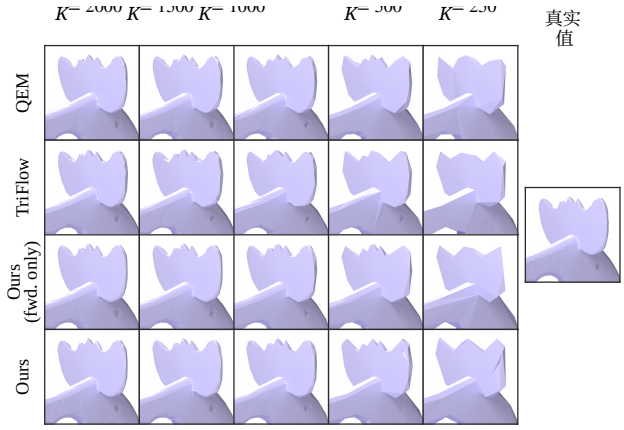


图 3。在不同顶点预算下的定性比较（表 2）。

表 3。在 $K = 1000$（倒角距离 $\times 10^{-5}$）下，不同几何优化流程中的低多边形重建。DMTet 和 DiffMC 使用 $n = 128$，而 Large Steps 使用固定拓扑二十面体球。

前端	指标	二次误差度量 ACVD	CWF	TriFlow	Ours（仅前向）	我们的方法
DMTet [46]	CD↓	1.82	4.42	7.01	2.85	1.56	**1.23**
	F1↑	0.21	0.16	0.12	0.19	0.23	**0.26**
DiffMC [56]	CD↓	2.07	4.27	7.46	1.98	1.44	**1.03**
	F1↑	0.21	0.17	0.11	0.19	0.23	**0.29**
Large Steps [37] 倒角距离↓		7.35	11.05	9.41	7.31	6.81	**6.47**
	F1↑	0.14	0.10	0.09	0.13	0.15	**0.21**

建效果，且这一优势适用于不同的基于梯度的几何优化流水线，不仅限于等值面提取，也适用于显式网格优化。

5.2. 逆向渲染

实验设置。我们采用 nvdiffrec [35]作为基线逆向渲染流程，并使用 FlexiCubes 作为几何前端。我们在八个 NeRF合成场景 [34]上进行评估。阶段 1 对几何、材质和光照进行优化，随后进行 UV参数化和纹理烘焙。阶段2 在固定连接关系下联合精化顶点位置和 UV 纹理，迭代 5,000 次。我们将 DIFFSIMP 集成到阶段 1 的几何优化中，同时保留相同的阶段 2。在 $K = 2000$，我们将方法与低分辨率 FlexiCubes（低分辨率）、QEM [17]，以及我们的仅前向变体进行比较。低分辨率 FlexiCubes 直接优化粗网格，而 QEM 和我们的方法（仅前向）则在阶段 1 优化之后、纹理烘焙之前简化几何。三种方法都采用与我们的方法相同的阶段 2 精化。我们还将方法与纹理感知QEM（TQEM） [18]和外观驱动简化（ADS） [23]，进行比较，后两者均作用于最终的带纹理参考结果。ADS 还会进一步精化由 TQEM 初始化的网格，使其与参考结果相匹配。我们报告留出视图上的峰值信噪比、结构相似性指数和学习感知图像块相似度。

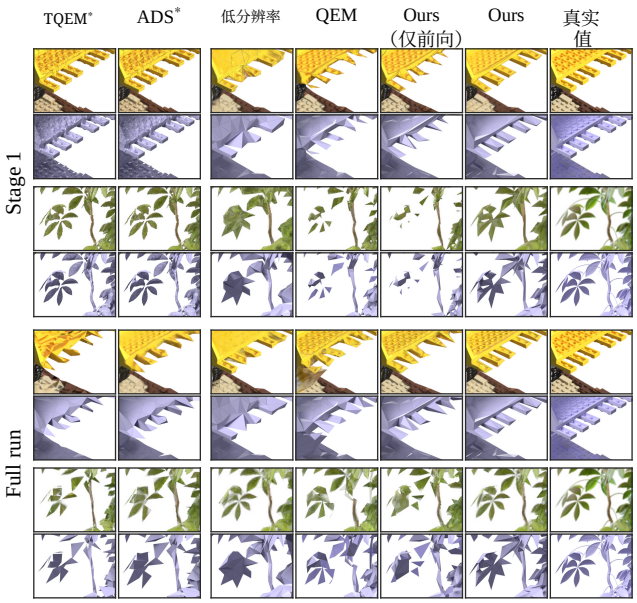


图 4. 定性逆向渲染结果（表 4）。标记为 * 的方法，其阶段 1 结果取自参考结果。

表 4. 八个 NeRF合成场景上的新视角合成（$K = 2000$）。完整运行包括阶段 2 精化。标记为 * 的方法会简化最终带纹理参考结果；其阶段 1 结果取自参考结果，故未列出。

	阶段 1				完整运行			
方法	峰值信噪比↑	结构相似性指数↑	学习感知图像块相似度↓	$\lvert V\rvert$	峰值信噪比↑	结构相似性指数↑	学习感知图像块相似度↓	$\lvert V\rvert$
参考	26.88	0.907	0.114	51,512	29.02	0.937	0.082	51,512
TQEM* [18]	–	–	–	–	22.47	0.853	0.139	2,000
ADS* [23]	–	–	–	–	25.71	0.907	0.110	2,000
低分辨率	23.73	0.880	0.132	1,883	25.56	0.907	0.106	1,883
二次误差度量 [17]	23.73	0.885	0.123	2,000	25.43	0.907	0.100	2,000
我们的方法（仅前向）	24.18	0.889	0.120	2,000	26.03	0.912	0.095	2,000
Ours	**25.56**	**0.906**	**0.109**	2,000	**27.29**	**0.925**	**0.085**	2,000

简化后的渲染保真度。如表 4所示，阶段 1 结束后，DIFFSIMP 仅比稠密参考低 1.32 dB，同时仅保留了其 3.9% 的顶点。如图 4所示，我们的方法在阶段 1 结束后和完整运行后都能更好地保留细微结构。它优于后处理 QEM 和我们的仅前向变体，表明这些优势不仅来自前向简化器本身的改进。在经过共享的阶段 2 精化后，这些优势依然存在。我们的方法在所有渲染指标上也都优于 TQEM 和 ADS，这表明，外观感知后处理无法达到通过可微分简化优化上游几何所实现的保真度。这些结果表明，通过简化过程进行优化能够使上游几何补偿简化引入的误差，从而改善最终的带纹理网格。

表5. 反向微分对79种形状的影响。CD（$\times 10^{-5}$）和 F1 分数用于评估最终的简化网格。Raw 和部署后分别表示在未简化网格和 $K = 500$ 网格上评估的渲染目标。我们报告了预算削减（递增阶段，迭代次数1000–3250）、目标预算优化（保持，3250–5000）以及整个区间（总体，1000–5000）内的平均相对目标变化（$\Delta L/L$，%）。

			原始 $\Delta L/L$	部署后 $\Delta L/L$		
变体	CD↓	F1↑	Hold	Ramp	Hold	总体
仅前向	3.971	0.156	-3.4	$+0.6$	$+0.4$	$+1.2$
幸存者梯度	3.652	0.152	$+6.4$	-5.5	-2.4	-6.9
簇梯度	3.533	0.177	$+19.3$	$\mathbf{-29.7}$	-2.1	-30.2
无二次曲面梯度	3.941	0.141	$+3.4$	$+20.6$	$+0.5$	$+21.6$
Ours	**2.891**	**0.182**	$+12.3$	-26.3	$\mathbf{-9.7}$	$\mathbf{-33.2}$

5.3. 方法分析

简化梯度的有效性。 我们使用第 5.1 节的基础重建流程 $K = 500$来评估我们的梯度所产生的影响。我们将其与仅前向 后处理以及三个采用我们前向简化器的循环内变体进行比较： *幸存者梯度* 仅将每个输出梯度传播给其幸存的输入顶点；*簇梯度* 将其等量分配给合并到该输出顶点中的所有输入顶点；而*无二次曲面梯度* 则将二次曲面视为常量，同时保留直接的顶点位置梯度。按照渐进式顶点预算调度，我们在三个区间内衡量优化进展：递增阶段（迭代 1000–3250，预算削减）、保持阶段（3250–5000，固定目标预算）和总体阶段（1000–5000）。在迭代 1000、3250 和 5000 次时，我们使用未简化网格（原始）及其在 $K = 500$ （已部署）下的简化版本，针对真实渲染结果评估渲染损失，而不考虑当前顶点预算。我们报告相对变化 $\Delta L/L = （L_{\text{end}} - L_{\text{start}}）/L_{\text{start}}$。

表 5显示，所有循环内变体相较于仅前向都改善了倒角距离，而簇梯度和我们的方法还改善了 F1 分数，这表明通过简化过程进行优化可以提升几何保真度。簇梯度在递增阶段取得了实质性进展，但在保持阶段基本停滞。幸存者梯度带来的收益有限，而移除二次曲面梯度会使部署目标恶化。相比之下，我们的方法持续降低部署目标，取得了总体上最大的改进和最佳的几何保真度。这些结果凸显了对简化过程的几何依赖关系进行求导的重要性，而不是依赖代理梯度传播。此外，即使每次迭代都会重新选择坍缩历史，基于梯度的优化仍然有效。

原始目标和部署后目标进一步揭示了优化–部署不匹配。仅前向会降低原始目标，却不会改善部署目标；而我们的方法即使在其原始目标上升的情况下，也能降低部署目标，

表 6。在 $K = 1000$ 下高斯随机场网格上的算子运行时间，以毫秒为单位。加速比均相对于我们的方法，Launch× 表示反向CUDA内核启动比（自动求导/我们的方法）。二次误差度量在 CPU 上运行，其余方法在图形处理器上运行。

	前向				反向			
$\lvert V_{\text{in}}\rvert$	Ours	二次误差度量	加速	我们的方法	Autograd	加速	启动×
50,145	8.65	530.32	61.3×	1.99	74.26	37.3×	81.5×
502,141	36.37	8,198.28	225.4×	2.34	144.53	61.8×	107.0×
5,045,762	322.21	102,830.06	319.1×	6.75	509.20	75.4×	122.5×

这种分歧表明，优化未简化的几何并不一定能改善其受预算约束的输出。通过简化进行优化，则能够让上游几何补偿最终复杂度降低所引入的误差。

计算效率。 我们在高斯随机场网格上，将 DIFFSIMP与 QEM 以及 PyTorch 自动微分重放进行了比较。如表6所示，我们的前向传播速度快 61.3–319.1×倍于 QEM，展现了图形处理器并行执行的效率。在坍缩历史相同的情况下，反向传播达到数量级的加速比且 CUDA 内核启动次数更少，这体现了我们的因子化反向传播：它能够并行计算塌缩局部向量-雅可比积，并通过二次曲面约简森林，避免跨轮次进行冗余的面级微分。在表 1中针对低多边形重建，DIFFSIMP 的运行时间为 1.25× 其基线流水线的 205 秒，而基线为 164 秒。在表4的逆向渲染中，其运行时间仍与基线流水线相当（3,262 秒对 3,255 秒）。图形处理器峰值内存使用量分别增加了 11 和 59 兆二进制字节。

6. 结论

我们提出 DIFFSIMP，将基于 QEM 的网格简化作为可微层集成到基于梯度的几何优化中。通过在固定记录的折叠历史的情况下对塌缩映射求导，我们的方法使上游优化能够补偿简化引入的误差。结合改进的前向简化器与因子化反向方案，DIFFSIMP 在低多边形重建和逆向渲染任务中优于所评估的后处理简化基线，同时仅带来适度的运行时间和内存开销。我们进一步表明，在更严格的顶点预算下，梯度传播带来的收益会不断增加，因为简化会引入更大的几何误差。这些改进适用于多种几何优化前端，证明了通过 DIFFSIMP 进行优化，而非将简化作为后处理步骤的有效性。

致谢

我们感谢 Xinyue Wei 的深入讨论和宝贵反馈，这些帮助塑造并完善了这项工作。

参考

[1] Alexandre Binninger、Ruben Wiersma、Philipp Herholz 和 Olga Sorkine-Hornung。TetWeave：利用动态生成的 Delaunay 四面体网格进行等值面提取，以实现基于梯度的网格优化。ACM 图形学汇刊，44(4)：146:1–146:19，2025。1, 2, 5[2] Xingyu Chen、Fu-Jen Chu、Pierre Gleize、Kevin J. Liang、Alexander Sax、Hao Tang、Weiyao Wang、Michelle Guo、Thibaut Hardin、Xiang Li、Aohan Lin、Jia-Wei Liu、Ziqi Ma、Anushka Sagar、Bowen Song、Xiaodong Wang、Jianing Yang、Bowen Zhang、Piotr Doll´ar、Georgia Gkioxari、Matt Feiszli 和 Jitendra Malik。SAM 3D：对图像中的任何内容进行 3Dfy。在 IEEE/CVF 计算机视觉与模式识别会议（CVPR）论文集，第 7220–7232 页，2026。1[3] Yihua Chen。基于神经隐式表示的渐进式网格简化算法。Computational Intelligence，40(1)：e12605，2024。2[4] Yiwen Chen、Tong He、Di Huang、Weicai Ye、Sijin Chen、Jiaxiang Tang、Zhongang Cai、Lei Yang、Gang Yu、Guosheng Lin 和 Chi Zhang。MeshAnything：利用自回归 Transformer 进行艺术家创建的网格生成。在国际学习表征会议（ICLR）论文集，2025。2[5] Yun-Chun Chen、Vladimir G. Kim、Noam Aigerman 和 Alec Jacobson。神经渐进式网格。在 ACM SIGGRAPH 2023 会议论文集，第 84:1–84:2 页，2023。[6] Zhen Chen、Zherong Pan、Kui Wu、Etienne Vouga 和 Xifeng Gao。面向通用 3D 模型的稳健低多边形网格化。ACM 图形学汇刊，42(4)：119:1–119:20，2023。[7] Jaehoon Choi、Rajvi Shah、Qinbo Li、Yipeng Wang、Ayush Saraf、Changil Kim、Jia-Bin Huang、Dinesh Manocha、Suhib Alsisan 和 Johannes Kopf。LTM：面向大型无界场景的轻量级带纹理网格提取与细化，以实现高效存储和实时渲染。在 IEEE/CVF 计算机视觉与模式识别会议（CVPR）论文集，第 5053–5063 页，2024。1, 2[8] Paolo Cignoni、Marco Callieri、Massimiliano Corsini、Matteo Dellepiane、Fabio Ganovelli 和 Guido Ranzuglia。MeshLab：开源网格处理工具。在 Eurographics 意大利分会会议论文集，第 129–136 页，2008。5[9] David Cohen-Steiner、Pierre Alliez 和 Mathieu Desbrun。变分形状近似。ACM 图形学汇刊，23(3)：905–914，2004。1, 2[10] Jinkai Cui、Kaiwen Song、Chumeng Niu 和 Juyong Zhang。用于端到端网格重建的距离场光栅化。在 ACM SIGGRAPH 2026 会议论文集，第 116:1–116:11 页，2026。1, 2

[11] Yishun Dou、Zhong Zheng、Qiaoqiao Jin、Rui Shi、Yuhan Li 和 Bingbing Ni。可微微网格构建。载于 IEEE/CVF 计算机视觉与模式识别会议（CVPR）论文集，第 4294 4303 页，2024 年 2 – 。 [12] Chuanjin Fan、Lifan Wu、Wenjie Chang、Hanzhi Chang、Wenfei Yang 和 Tianzhu Zhang。ExMesh：具有拓扑适应的显式网格重建。载于 IEEE/CVF 计算机视觉与模式识别会议（CVPR）论文集，第 27136–27145 页，2026 年。 2[13] Jun Gao、Tianchang Shen、Zian Wang、Wenzheng Chen、Kangxue Yin、Daiqing Li、Or Litany、Zan Gojcic 和 Sanja Fidler。GET3D：从图像中学习高质量三维纹理形状的生成模型。载于神经信息处理系统进展（NeurIPS），第 31841–31854 页，2022 年。 1, 2[14] Xiang Gao、Xin mu Wang、Xiaolong Wu、Jiazhi Li、Jingyu Shi、Yu Guo、Yuanpeng Liu、Xiyun Song、Heather Yu、Zo ng- fang Lin 和 Xianfeng David Gu。基于多视图图像的高亏格表面网格逆向渲染。载于欧洲计算机视觉会议（ DV），第 3 373 页，2026 年 3 363 2 D Vision（ ） 。 [15] Yaoan Gao、Jiamin Xu、James Tompkin、Qi Wang、Zheng Dong、Hujun Bao、Yujun Shen、Huamin Wang、Changqing Zou 和 Weiwei Xu。采用可微面积光源着色的高效物体重建。载于 SIGGRAPH Asia 2025 会议论文集，第 65:1–65:12 页，2025 年。 1[16] Michael Garland。基于二次曲面的多边形表面简化。博士学位论文，卡内基梅隆大学，1999 年。技术报告 CMU-CS-99-105。 1, 2[17] Michael Garland 和 Paul S. Heckbert。使用二次误差度量进行表面简化。载于第 24 届计算机图形学与交互技术年度会议（SIGGRAPH ’97）论文集，第 209–216 页，1997 年。 1, 2, 5, 6, 7[18] Michael Garland 和 Paul S. Heckbert。使用二次误差度量结合颜色与纹理进行表面简化。载于 IEEE Visualization ’98 论文集，第 263–269 页，1998 年。 7[19] Antoine Gu´edon、Diego Gomez、Nissim Maruani、Bingchen Gong、George Drettakis 和 Maks Ovsjanikov。MILo：用于详细且高效表面重建的网格内循环高斯泼溅。ACM 图形学汇刊，44(6):223:1–223:15，2025 年。 1, 2[20] Benoˆıt Guillard、Federico Stella 和 Pascal Fua。MeshUDF：无符号距离场网络的快速可微网格化。载于欧洲计算机视觉会议（ECCV），第 576–592 页，2022 年。 2[21] Benoˆıt Guillard、Edoardo Remelli、Artem Lukoianov、Pierre Yvernay、Ste phan R. Richter、Timur M. Bagautdinov、Pierre Baqu´e 和 Pascal Fua。DeepMesh：可微等值面提取。IEEE模式分析与机器智能汇刊，46(11):7072–7087，2024 年。 2[22] Rana Hanocka、Amir Hertz、Noa Fish、Raja Giryes、Shachar Fleishman 和 Daniel Cohen-Or。MeshCNN：一种带有边的网络。ACM 图形学汇刊，38(4):90:1–90:12，2019 年。 2

[23] Jon Hasselgren、Jacob Munkberg、Jaakko Lehtinen、Miika Aittala 和 Samuli Laine。外观驱动的自动三维模型简化。载于 Eurographics 渲染专题研讨会——仅 DL 赛道，第 85–97 页，2021 年。 2, 7[24] Jon Hasselgren、Nikolai Hofmann 和 Jacob Munkberg。使用蒙特卡洛渲染与去噪从图像中分解形状、光照与材质。载于神经信息处理系统进展（NeurIPS），第 22856–22869 页，2022 年。 1[25] Hugues Hoppe。渐进式网格。载于第 23 届计算机图形学与交互技术年度会议（SIGGRAPH '96）论文集，第 99–108 页，1996 年。 2[26] Hugues Hoppe。用于简化具有外观属性网格的新二次度量。载于 IEEE Visualization '99 论文集，第 59–66 页，1999 年。 3[27] Julian Knodt。单边折叠四边形主导网格简化。ACM 图形学汇刊，44(4):145:1–145:23，2025 年。 1[28] Jiabao Lei、Kewei Shi、Zhihao Liang 和 Kui Jia。AR Mesh：通过下一级细节预测实现的自回归网格生成。载于神经信息处理系统进展（NeurIPS），第 179454–179474 页，2025 年。 2[29] Haoxuan Li、Ziya Erkoc¸、Daniele Sirigatti、Vladislav Rosov、Lei Li、Angela Dai 和 Matthias Nießner。TriFlow：通过最近顶点向量场生成艺术家风格的三维网格拓扑。载于欧洲计算机视觉会议（ECCV），第 407–425 页，2026 年。 2, 5, 6[30] Peter Lindstrom 和 Greg Turk。快速且节省内存的多边形简化。载于 IEEE Visualization '98 论文集，第 279–286 页，1998 年。 2, 3[31] Hsueh-Ti Derek Liu、Xiaoting Zhang 和 Cem Yuksel。对自然场景中的纹理三角形网格进行简化。ACM 图形学汇刊，44(6):181:1–181:16，2025 年。 1[32]Hang Long、Tianhao Zhao、Junkai Lin、Youjia Zhang、Huipeng Guo、Rendong Liang、Jiale Xu、Jozef Hladk´y、Matthias Nießner、Yuanming Hu 和 Wei Yang。LATO.2：结合顶点流与拓扑流的因子化三维网格生成。arXiv 预印本 arXiv:2607.10623，2026 年。 2[33] Nissim Maruani、Maks Ovsjanikov、Pierre Alliez 和 Mathieu Desbrun。PoNQ：基于神经二次误差度量的网格表示。载于 IEEE/CVF 计算机视觉与模式识别会议（CVPR）论文集，第 3647–3657 页，2024 年。 2[34]Ben Mildenhall、Pratul P. Srinivasan、Matthew Tancik、Jonathan T. Barron、Ravi Ramamoorthi 和 Ren Ng。NeRF：将场景表示为用于视图合成的神经辐射场。载于欧洲计算机视觉会议（ECCV），第 405–421 页，2020 年。 7[35] Jacob Munkberg、Jon Hasselgren、Tianchang Shen、Jun Gao、Wenzheng Chen、Alex Evans、Thomas M¨uller 和 Sanja Fidler。从图像中提取三角形三维模型、材质与光照。载于 IEEE/CVF 计算机视觉与模式识别会议（CVPR）论文集，第 8280–8290 页，2022 年。 1, 2, 7[36] Ashish Myles、Nico Pietroni 和 Denis Zorin。稳健的场对齐全局参数化。ACM 图形学汇刊，33(4):135:1–135:14，2014 年。 5

[37] Baptiste Nicolet、Alec Jacobson 和 Wenzel Jakob。几何逆向渲染中的大步长。ACM 图形学汇刊，40(6):248:1–248:13，2021。2、6、7[38] Seonghun Oh、Xiaodi Yuan、Xinyue Wei、Ruoxi Shi、Fanbo Xiang、Minghua Liu 和 Hao Su。PaMO：基于图形处理器的无相交低多边形建模并行网格优化。《计算机图形学论坛》，44(7):e70267，2025。2、3、5、6 [39] Werner Palfinger。用于逆向渲染的连续重网格化。《计算机动画与虚拟世界》，33 2101，2022 5 2 ( ):e 。[40] Alexandros Papageorgiou 和 Nikos Platis。图形处理器上的三角网格简化。The Visual Computer，31(2):235–244，2015。2[41] Adam Paszke、Sam Gross、Francisco Massa、Adam Lerer、James Bradbury、Gregory Chanan、Trevor Killeen、Zem- ing Lin、Natalia Gimelshein、Luca Antiga、Alban Desmai- son、Andreas K¨opf、Edward Yang、Zachary DeVito、Mar- tin Raison、Alykhan Tejani、Sasank Chilamkurthy、Benoit Steiner、Lu Fang、Junjie Bai 和 Soumith Chintala。Py- Torch：命令式风格的高性能深度学习库。载于《神经信息处理系统进展》（NeurIPS），第 8024–8035页，2019。5[42] Rolandos Alexandros Potamias、Stylianos Ploumpis 和 Stefanos Zafeiriou。神经网格简化。载于 IEEE/CVF计算机视觉与模式识别会议（CVPR）论文集，第18583–18592页，2022。1、2[43] Marie-Julie Rakotosaona、Noam Aigerman、Niloy J. Mitra、Maks Ovsjanikov 和 Paul Guerrero。可微表面三角剖分。ACM 图形学汇刊，40(6):267:1–267:13，2021。2[44] Edoardo Remelli、Artem Lukoianov、Stephan R. Richter、Beno ̂ıt Guillard、Timur M. Bagautdinov、Pierre Baqu´e 和 Pascal Fua。MeshSDF：可微等值面提取。载于《神经信息处理系统进展》（NeurIPS），第22468–22478页，2020。1、2 [45] Jarek Rossignac 和 Paul Borrel。用于渲染复杂场景的多分辨率三维近似。载于《计算机图形学建模：方法与应用》，第 455–465页。施普林格，1993。2、5、6[46] Tianchang Shen、Jun Gao、Kangxue Yin、Ming-Yu Liu 和 Sanja Fidler。深度行进四面体：用于高分辨率三维形状合成的混合表示。载于《神经信息处理系统进展》（NeurIPS），第6087–6101页，2021。1、2、6、7[47] Tianchang Shen、Jacob Munkberg、Jon Hasselgren、Kangxue Yin、Zian Wang、Wenzheng Chen、Zan Gojcic、Sanja Fidler、Nicholas Sharp 和 Jun Gao。用于基于梯度的网格优化的灵活等值面提取。ACM 图形学汇刊，42(4):37:1–37:16，2023。1、2、5、6[48] Yawar Siddiqui、Antonio Alliegro、Alexey Artemov、Ta- tiana Tommasi、Daniele Sirigatti、Vladislav Rosov、Angela Dai 和 Matthias Nießner。MeshGPT：使用仅解码器Transformer生成三角网格。载于 IEEE/CVF计算机视觉与模式识别会议（CVPR）论文集，第19615–19625页，2024。2

[49] Sanghyun Son、Matheus Gadelha、Yang Zhou、Zexiang Xu、Ming C. Lin 和 Yi Zhou。DMesh：可微网格表示。载于《神经信息处理系统进展》（NeurIPS），第12035–12077页，2024。2[50] Sanghyun Son、Matheus Gadelha、Yang Zhou、Matthew Fisher、Zexiang Xu、Yi-Ling Qiao、Ming C. Lin 和 Yi Zhou。DMesh++：用于复杂形状的高效可微网格。载于 IEEE/CVF国际计算机视觉会议（ICCV）论文集，第26590–26599页，2025。2[51] Kaiyu Song、Hanjiang Lai、Yaqing Zhang、Chuangjian Cai、Yan Pan、Kun Yue 和 Jian Yin。拓扑雕刻器、形状细化器：用于高保真三维网格生成的离散扩散模型。arXiv预印本 arXiv:2510.21264，2025。2[52] 腾讯混元3D团队。混元3D 2.1：从图像到具有生产级基于物理的渲染材质的高保真三维资产。arXiv预印本 arXiv:2506.15442，2025。1[53] S´ebastien Valette 和 Jean-Marc Chassery。用于均匀多边形网格粗化的近似质心沃罗诺伊图。《计算机图形学论坛》，23 389，2004 3 381 2、5、6 ( ): – 。[54] S´ebastien Valette、Jean-Marc Chassery 和 R´emy Prost。基于度量的离散沃罗诺伊图的三维三角网格通用重新网格化。《IEEE可视化与计算机图形学汇刊》，14(2):369–381，2008。2[55] Dilin Wang、Xiaoyu Xiang、Kihyuk Sohn、Tom Monnier、Yu- Ying Yeh、Thu Nguyen-Phuoc、Jiawen Zhang、Yuchen Fan、Antoine Toisoul、Hyunyoung Jung、Prithviraj Dhar、Michael Bunnell、Nikolaos Sarafianos、Chuhang Zou、Roman Shapovalov、Andrea Vedaldi 和 Rakesh Ranjan。AssetGen：以交互速度部署三维资产生成。arXiv预印本 arXiv:2605.26137，2026。1[56] Xinyue Wei、Fanbo Xiang、Sai Bi、Anpei Chen、Kalyan Sunkavalli、Zexiang Xu 和 Hao Su。NeuManifold：支持高效高质量渲染的神经网络水密流形重建。载于 IEEE/CVF冬季计算机视觉应用会议（WACV）论文集，第731–741页，2025。1、2、6、7[57] Jianfeng Xiang、Zelong Lv、Sicheng Xu、Yu Deng、Ruicheng Wang、Bowen Zhang、Dong Chen、Xin Tong 和 Jiaolong Yang。用于可扩展多用途三维生成的结构化三维潜变量。载于 IEEE/CVF计算机视觉与模式识别会议（CVPR）论文集，第21469–21480页，2025。1、2[58] Rui Xu、Longdu Liu、Ningna Wang、Shuangmin Chen、Shiqing Xin、Xiaohu Guo、Zichun Zhong、Taku Komura、Wenping Wang 和 Changhe Tu。CWF：在高质量网格简化中整合弱特征。ACM 图形学汇刊，43(4):80:1–80:14，2024。1、2、5、6[59] Kai Ye、Chong Gao、Guanbin Li、Wenzheng Chen 和 Bao- quan Chen。GeoSplatting：面向几何引导的基于高斯的物理逆向渲染。载于 IEEE/CVF 国际计算机视觉会议（ ICCV ）论文集，第 28991–29000页，2025。1、2

DiffSimp：用于基于梯度的低多边形建模的可微网格简化补充材料

目录

A. 实验细节.12
A.1. 低多边形网格重建.12 A.2. 逆向渲染.13

B. 实现细节.13
B.1. 代价阈值与预算控制.13 B.2. 放置分支与回退机制.13 B.3. 顶点放置的梯度.14 B.4. 面二次曲面导数.14

C. 补充方法分析.15C

.1. 前向简化器.15 C.1.1. 组件消融.15 C.1.2. 与其他简化算子的比较.15

C.2. 反向.16 C.2.1. 遍数与二次误差度量简化森林效率.16 C.2.2. 反向传播工作量.16 C.2.3. 梯度正确性.17 C.3. 迭代优化.17 C.3.1. 训练损失曲线.17 C.3.2. 简化后微调.17 C.3.3. 渐进式预算调度消融.17**D. 其他逆向渲染结果.19**D.1. 详细的 nvdiffrec 结果.19 D.2. 使用 ExMesh 进行场景重建.19**E. 局限性与未来工作.23**

A. 实验细节

A.1. 低多边形网格重建

对于第 5.1 节中的低多边形网格重建实验，我们基于官方的 FlexiCubes 实现，在网格分辨率 $n = 128$ 下优化随机初始化的符号距离函数，进行 $T = 5000$轮次迭代，使用多视图轮廓、深度和法线监督，不使用真实符号距离函数。我们的重建目标为

$$\mathcal{L} = \lambda_{\text{mask}}\mathcal{L}_{\text{mask}} + \lambda_{\text{depth}}\mathcal{L}_{\text{depth}} + \lambda_{\text{normal}}\mathcal{L}_{\text{normal}} + \lambda_{\text{dev}}\mathcal{L}_{\text{dev}} + \lambda_{\text{sign}}\mathcal{L}_{\text{sign}}, \tag{S1}$$

其中 $\mathcal{L}_{\text{dev}}$ 和 $\mathcal{L}_{\text{sign}}$ 分别是 FlexiCubes 的变形正则项和符号正则化项。我们设置 $\lambda_{\text{mask}} = 5$， $\lambda_{\text{depth}} = 400$，$\lambda_{\text{normal}} = 1$，以及 $\lambda_{\text{dev}} = 1$。遵循 FlexiCubes 的官方实现，我们将 λ_{sign} 从 0.2 线性衰减至 0.01，历时前 $0.25T$次迭代。表 3 中的其他重建流水线采用相同的多视图监督方案，并针对各自的前端进行设置。DiffMC 遵循我们的 FlexiCubes 重建设置，但不使用 $\mathcal{L}_{\text{dev}}$。DMTet 也省略 $\mathcal{L}_{\text{dev}}$，但使用 $\lambda_{\text{mask}} = 10$，$\lambda_{\text{depth}} = 250$和 $\lambda_{\text{normal}} = 1$，同时保留相同的符号正则化调度方案。Large Steps 使用与 DMTet 相同的渲染损失权重，不使用变形正则化或符号正则化。在每种前端中，我们的方法和后处理基线采用相同的基础重建设置。

A.2. 逆向渲染

对于第 5.2 节中的逆向渲染，我们遵循 nvdiffrec 针对八个 NeRF合成场景的官方配置，并在 $n = 128$ 下使用 FlexiCubes。两个阶段均运行 5,000 次迭代，采用原始的逆向渲染目标函数，

$$\mathcal{L} = \mathcal{L}_{\text{image}} + \mathcal{L}_{\text{mask}} + \lambda \mathcal{L}_{\text{reg}}, \tag{S2}$$

其中正则化项遵循官方配置。对于后处理 ADS 基线，由于该方法需要一个初始简化纹理网格，我们对 nvdiffrec 两个阶段完成后得到的最终带纹理网格应用 TQEM。随后，以最终带纹理网格作为参考，在 2048^2 分辨率下对 TQEM 的输出进行 10,000 次迭代细化，迭代次数和分辨率均遵循原始 ADS 的设置。

B. 实现细节

我们在第 4.1 节提供前向简化的更多实现细节，以及第 4.3节中反向公式所依据的闭式导数。

B.1. 代价阈值与预算控制

我们将逐轮代价阈值初始化为 τ_p （第 4.1节），并将 $\tau_1 = 10^{-4}$ 设为 $\tau_{p+1} = \sqrt{10}\tau_p$。在每一遍中，并行坍缩轮次会持续进行，直到独立区域选择（第 3.2节）选出的所有边都不满足 $C_e \le \tau_p$。设 $|V|$ 为当前顶点数。如果某一轮选出的边数超过 $|V| - K$，我们只坍缩代价最低的 $|V| - K$ 个候选项，以确保不超出目标预算 K。

B.2. 放置分支与回退机制

考虑一遍中的一次坍缩 $e = (i, j) \in \mathcal{E}_r$。在没有歧义时，我们省略轮次上标： v_i, v_j 表示坍缩前的位置$v_i^{(r-1)}$，$v_j^{(r-1)}$，而 Q_e, v_e 表示 $Q_e^{(r)}, v_e^{(r)}$。与式（1）一样， $Q_e = Q_i^{(r-1)} + Q_j^{(r-1)}$ 包含以下分块 A_e, b_e, c_e。

正则化 QEM 放置。 当满足以下条件时，我们选择正则化 QEM 分支

$$\frac{\mu_{\min}(A_e)}{\mu_{\max}(A_e)} > \kappa, \qquad \kappa = 10^{-3}, \tag{S3}$$

其中， $\mu_{\min}$ 和 $\mu_{\max}$ 是 A_e的极端特征值；此外，我们还要求 $\mu_{\max}(A_e) > 0$。随后将合并顶点放置于

$$v_e = -(A_e + \lambda_e I)^{-1} b_e, \qquad \lambda_e = \eta \max_k |(A_e)_{kk}|, \quad \eta = 10^{-6}. \tag{S4}$$

边线段回退。 当未选择正则化 QEM 分支时，我们改为沿边最小化坍缩误差 E_e（第 3.1 节）。设 $\delta = v_j - v_i$，当满足以下条件时使用线段放置

$$\delta^\top A_e \delta > \epsilon_s, \qquad \epsilon_s = 10^{-7}, \tag{S5}$$

表示为

$$\begin{aligned} \alpha^\star &= -\frac{\delta^\top (A_e v_i + b_e)}{\delta^\top A_e \delta}, \\ \gamma_e &= \text{clamp}(\alpha^\star, 0, 1), \\ v_e &= v_i + \gamma_e \delta. \end{aligned} \tag{S6}$$

钳制会使 v_e 保持在边上；当 $\alpha^\star \notin [0, 1]$ 时，保持在端点处。

中点回退。 如果公式（S5）也不满足，我们使用 $v_e = \frac{1}{2}(v_i + v_j)$。

对于每次坍缩，前向传播还会记录反向传播所需的离散选择：放置分支、线段钳制状态（内部点或任一端点），以及 k_e 在公式（S4）中取得最大值的对角线索引。这些选择属于折叠历史 σ （第4.2节），并在微分过程中保持不变。在十个重建样本中，正则化二次误差度量、边-线段和中点放置分别占循环内折叠的 63.1%、35.6% 和 1.3%。

B.3. 顶点放置的梯度

我们在第 B.2节推导放置分支的塌缩局部向量-雅可比积，记 g_x 为损失相对于 x的梯度。给定输入位置梯度 $g_{v_e} = g_{v_e}^{(r)}$，每个分支都会产生直接位置梯度 g_{v_i}、g_{v_j} 以及二次曲面梯度 $G_e = (\partial v_e/\partial Q_e)^\top g_{v_e}$。记 G_A 和 G_b 为相对于独立块 A_e 和 b_e的梯度，我们将对称二次曲面梯度表示为

$$G_e = \begin{bmatrix} G_A & \frac{1}{2}G_b \\ \frac{1}{2}G_b^\top & 0 \end{bmatrix}. \tag{S7}$$

因子 12表示 Q_e 中 b_e 的两个对称出现项，而零块反映了 c_e 不会影响顶点放置。令 $\mathrm{sym}(M) = \frac{1}{2}(M + M^\top)$ 表示矩阵的对称部分。

正则化 QEM 放置。 在此分支中，式（S3）和 $\mu_{\max}(A_e) > 0$ 意味着 $A_e \succ 0$。因此，其对角元素为正， 中的绝对值可以去掉。对记录的 $(A_e + \lambda_e I)\, v_e + b_e = 0$ 求导，并将 k_e固定，可得

$$dv_e = -(A_e + \lambda_e I)^{-1}(dA_e\, v_e + d\lambda_e\, v_e + db_e), \qquad d\lambda_e = \eta\,(dA_e)_{k_e k_e}. \tag{S8}$$

对于 $u = (A_e + \lambda_e I)^{-1} g_{v_e}$，VJP 为

$$\begin{aligned} G_A &= -\,\mathrm{sym}(u v_e^\top) - \eta\,(u^\top v_e)\,\mathbf{e}_{k_e}\mathbf{e}_{k_e}^\top, \\ G_b &= -u, \qquad g_{v_i} = g_{v_j} = 0, \end{aligned} \tag{S9}$$

其中 $\mathbf{e}_{k_e}$ 是第 k_e 个单位向量。此分支完全通过二次曲面传播顶点放置梯度，包括 λ_e 对 A_e 的依赖。

边分段的放置。 对于内部解 $\gamma_e = \alpha^\star \in (0,\ 1)$，令 $\rho = \delta^\top A_e \delta$、$z_e = A_e v_e + b_e$，以及 d

$$\beta_e = -\frac{\delta^\top g_{v_e}}{\rho}, \qquad h_e = g_{v_e} + \beta_e A_e \delta. \tag{S10}$$

对公式（S6）求导可得

$$\begin{aligned} g_{v_i} &= (1 - \alpha^\star)\, h_e - \beta_e z_e, \quad g_{v_j} = \alpha^\star h_e + \beta_e z_e, \\ G_A &= \beta_e\, \mathrm{sym}(\delta v_e^\top), \qquad\qquad G_b = \beta_e \delta. \end{aligned} \tag{S11}$$

如果将解限制在某个端点，v_e 就与该端点重合，并且不依赖于 Q_e:

$$(g_{v_i}, g_{v_j}) = \begin{cases} (g_{v_e}, 0), & \gamma_e = 0, \\ (0, g_{v_e}), & \gamma_e = 1, \end{cases} \qquad G_e = 0. \tag{S12}$$

中点放置。对于 $v_e = \frac{1}{2}(v_i + v_j)$，

$$g_{v_i} = g_{v_j} = \tfrac{1}{2} g_{v_e}, \qquad G_e = 0. \tag{S13}$$

在所有分支中，G_e 是公式（7）中的放置项，并会加到两个端点的二次曲面梯度上。直接梯度 g_{v_i} 和 g_{v_j} 构成逐轮向量-雅可比乘积 $g^{(r-1)} = J_r^\top g^{(r)}$的位置部分。

B.4. 面二次曲面导数

在一遍开始时，设 $f = (i,\ j,\ k) \in F_{p-1}$为一个非退化面。我们将其遍历输入位置记为 v_i、v_j、v_k $v_i^{(0)}$、$v_j^{(0)}$、$v_k^{(0)}$ 并定义

$$\begin{aligned} s &= (v_j - v_i) \times (v_k - v_i), \qquad n_f = s/\|s\|, \\ d_f &= -n_f^\top v_i, \qquad p_f = \begin{bmatrix} n_f \\ d_f \end{bmatrix}, \qquad w_f = \frac{\|s\|}{2\bar{a}}, \qquad K_f = w_f p_f p_f^\top, \end{aligned} \tag{S14}$$

使得 $w_f = a_f/\bar{a}$ 如第 4.1 节所示。从公式（8）可知，传入的面二次误差矩阵梯度为 $g_{K_f} = g_{Q_i}^{(0)} + g_{Q_j}^{(0)} + g_{Q_k}^{(0)}$。对 K_f 求导可得

$$g_{p_f} = 2w_f \,\mathrm{sym}(g_{K_f})p_f, \qquad g_{w_f} = p_f^\top g_{K_f} p_f. \tag{S15}$$

我们将 $g_{p_f} = (g_{n_f}^\top, g_{d_f})^\top$ 划分为法向量部分和偏移量部分。由于 d_f 还取决于 n_f，因此 n_f 上的总梯度为 $\widetilde{g}_{n_f} = g_{n_f} - g_{d_f} v_i$，而相对于 s 的梯度为

$$g_s = \frac{1}{\|s\|}(I - n_f n_f^\top)\widetilde{g}_{n_f} + \frac{g_{w_f}}{2\bar{a}} n_f. \tag{S16}$$

面对此顶点位置梯度的贡献可由叉积得到：

$$\begin{aligned} g_{v_j}^f &= (v_k - v_i) \times g_s, \\ g_{v_k}^f &= g_s \times (v_j - v_i), \\ g_{v_i}^f &= -g_{d_f} n_f - g_{v_j}^f - g_{v_k}^f. \end{aligned} \tag{S17}$$

将这些贡献在关联面上求和，并加上直接位置梯度 $g_{v_i}^{(0)}$ 即可得到公式（8）中的遍输入梯度。如第 4.1 节所述，$\bar{a}$ 在微分过程中保持不变。在固定折叠历史下，统一缩放所有二次曲面不会改变每个分支中的顶点放置。

C. 补充方法分析

我们进一步分析前向简化器、固定历史反向传播和迭代优化。除非另有说明，我们遵循第 5.1 节中的低多边形网格重建设置，并使用相同的 79 个形状。我们报告经过 10^5 缩放的平方倒角距离（CD）。

C.1. 前向简化器

C.1.1. 组件消融

通过从我们的前向简化器中分别移除面积加权面二次型、代价阈值调度和逐遍二次曲面重新计算，我们评估了这三个组件的贡献。所有变体都将相同 79 个形状的原始网格（平均 33,228 个顶点）简化至 $K = 1000$，且不进行优化。基线表示未采用这三项改进的并行基于 QEM 的边折叠简化器。如表 S1所示，与顺序 QEM 相比，基线的平均倒角距离更高（1.879 对 1.774），F1 分数更低（0.196 对 0.215），这体现了并行边折叠在质量与效率之间的权衡。我们的完整前向简化器将平均倒角距离降至 1.250，而移除任一组件都会导致倒角距离增加、F1 分数降低。图 S1 展示了这些组件在定性方面的影响。

表 S1。在 $K = 1000$且不进行优化的情况下，对 79 个原始网格进行前向简化器的组件消融实验。指标定义同表 1。倒角距离按 10^5 进行缩放。

变体	CD↓	F1↑	NC↑
QEM	1.774	0.215	0.964
Base	1.879	0.196	0.965
我们的方法（不使用面积加权面二次曲面）	1.601	0.196	**0.970**
我们的方法（不使用阈值调度）	1.698	0.216	0.965
我们的方法（不进行二次型重计算）	1.505	0.217	0.966
Ours	**1.250**	**0.230**	**0.970**

C.1.2. 与其他简化算子的比较

表 S2 比较了在相同的 79 个原始网格上，我们的方法这一前向简化器与 QEM、顶点聚类和 PaMO 的表现，测试条件为 K =1000 且未进行优化。这些方法呈现出不同的权衡：顶点聚类速度快，但由于空间量化会引入较大的几何误差；PaMO 则能确保输出无交，但计算成本较高。顺序 QEM 的几何保真度优于前两者，但耗时显著长于我们的方法。与所评估的方法相比，我们的前向简化器实现了最低的倒角距离、最高的 F1 分数和法向一致性，且简化时间最短（7.5 毫秒），同时其简化率和简化比与 QEM 相当。这种兼顾近似质量与效率的表现，使其适合在基于梯度的优化过程中反复应用。



在 $K = 1000$上对原始网格使用前向简化器进行组件消融，未进行优化（表 S1）。每个简化结果下方显示倒角距离（$\times 10^{-5}$），原始网格（真实值）下方显示 $|V|$。

表 S2。在 $K = 1000$ 条件下，79 个原始网格的简化算子对比，未进行优化。QEM 和顶点聚类在 CPU 上运行，而 PaMO 和我们的方法在图形处理器上运行。倒角距离按 10^5 缩放。

算子	CD↓	F1↑	NC↑	简化率(%)↓	简化比(%)↓	时间（毫秒）↓
QEM	1.774	0.215	0.964	0.42	0.02	368.3
顶点聚类	39.528	0.072	0.895	9.66	0.03	11.7
PaMO	5.241	0.114	0.950	**0.00**	**0.00**	817.4
Ours	**1.250**	**0.230**	**0.970**	0.47	0.06	**7.5**

C.2. 反向

C.2.1. 遍数与二次误差度量简化森林效率

我们的方法通过在归约森林中传播二次曲面梯度，大幅减少了每一遍中的计算，使面二次误差度量在多个坍缩轮次中每遍只需进行一次微分。如表 S3 所示，即使在 $K = 250$，简化平均也只需 4.3 遍，尽管包含 101.3 个坍缩轮次。相比之下，每轮重新计算二次曲面需要重复进行面二次误差度量微分。因此，我们的逐遍方法实现了 5.5–8.8× 倍的反向传播执行速度，展现了在坍缩轮次之间复用二次曲面的效率。需要注意的是，如第 4.3 节所述，每轮变体仍会并行计算塌缩局部向量-雅可比积。

表 S3。在 RTX 5090 上，不同顶点预算下的阈值遍数、坍缩轮次和算子运行时间。测量结果基于 79 个原始网格取平均，未进行优化，并且每个网格各进行一次前向和反向评估。遍数和轮次对应于我们的方法，其中轮次为所有遍的总和。每轮方法会在每个坍缩轮次后重新计算二次曲面。

			前向（毫秒）		反向（毫秒）	
K	遍历	轮次	Ours	每轮	Ours	每轮
2000	1.9	57.5	**6.40**	31.24	**0.84**	4.59
1000	2.6	74.2	**7.48**	42.34	**0.96**	6.16
250	4.3	101.3	**10.28**	65.20	**1.15**	10.17

C.2.2. 反向传播工作量

第 4.3 节中的二次型归约森林公式对每次坍缩执行固定规模的更新，并对每遍中的每个输入面仅微分一次，因此工作量与 $W = \sum_p \left(\sum_{r=1}^{R_p} |\mathcal{E}_r| + |F_{p-1}|\right)$ 成正比。这里，R_p 表示遍 p中的坍缩轮次数，为简洁起见，省略了 $\mathcal{E}_r$ 的遍次索引。如表 S4所示，在测试的各输入规模下，我们启动的 GPU 线程数与 W 之比保持在 2.05–2.25 附近。相比之下，对相同记录历史进行自动微分重放时，每单位 W会启动 501–772 个线程，而归一化的直接面导数计数 $\sum_e |\mathcal{F}_e|/W$则从 5.40 增至 21.95。这一增长反映了在连续塌缩过程中，同一面二次型贡献被反复微分并累积到顶点二次型中。这些结果验证了我们因子化反向传播的线性工作量结构，并展示了二次曲面归约森林如何避免在多个坍缩轮次中重复进行面级微分。

表 S4。 $K = 1000$处高斯随机场网格上的反向传播工作量。启动的 GPU 线程数和面导数计算次数均按 W 进行归一化。 $\mathcal{F}_e$ 表示对用于坍缩边 e 的顶点二次型作出贡献的面，求和范围为所有记录的坍缩操作。

				线程/W		
$\lvert V_{\text{in}}\rvert$	遍历	$\sum_p R_p$	W	Ours	Autograd	$\sum_e \lvert\mathcal{F}_e\rvert/W$
50,145	4	86	178,573	2.25	501	5.40
502,141	4	146	1,553,225	2.08	658	12.25
5,045,762	4	225	15,212,320	2.05	772	21.95

C.2.3. 梯度正确性

我们将 DIFFSIMP 的因子化 CUDA 反向传播与对相同记录折叠历史进行 PyTorch 自动微分重放的结果进行比较验证。我们从 79 个原始网格中随机选取十个，在 $K = 1000$ 下对每个网格各执行一次前向和反向评估，不进行优化。对于每个网格，我们在简化顶点位置上定义一个随机线性损失。令 G 和 G_{AD64} 分别表示通过我们的双精度 CUDA 反向传播和双精度 PyTorch 自动微分计算得到的输入顶点梯度。我们计算展平顶点梯度上的相对 L_2 误差 $\|G - G_{\text{AD64}}\|_2 / \|G_{\text{AD64}}\|_2$。相对 L_2 误差的平均值和最大值分别为 4.7×10^{-12} 和 2.5×10^{-11}，这表明我们的 CUDA 反向传播与自动微分参考之间在数值上几乎完全一致。

C.3. 迭代优化

C.3.1. 训练损失曲线

图 S2 比较了 DIFFSIMP 与低多边形网格重建（第 5.1节）和逆向渲染（第 5.2节）中未简化参考方法的训练损失。在 $T = 5000$下，预热（$t_w = 0.2T$）和预算缩减（$t_r = 0.45T$）调度方案将训练分为预热（0–1000）、递增阶段（1000–3250）和保持阶段（3250–5000）。两种方法在预热阶段都优化相同的稠密几何。在递增阶段，随着顶点预算逐步减少，我们的训练损失不断增加。达到目标预算后，在整个保持阶段，我们的训练损失继续下降；尽管优化过程中反复进行简化，仍呈现出明显的收敛趋势。

C.3.2. 简化后微调

另一种方法是先对稠密重建进行一次简化，然后在固定连接关系下优化简化网格。为构建这一基线，我们将前向简化器作为后处理应用，并使用第 A.1节所述的 Large Steps 及其多视图重建目标，对所得顶点位置额外细化 2,000 次迭代。表 S5显示，即使经过额外的 2,000 次迭代，简化后微调在所有测试的顶点预算下，其倒角距离和 F1 分数仍无法匹配 DIFFSIMP 的循环内输出。两项指标之间的差距在更严格的预算下均会增大，其中最大差异出现在 $K = 500$。

表 S5. 对 79 种形状进行简化后微调。Init 表示由我们的前向简化器作为后处理进行简化的基线 FlexiCubes 重建结果。Post-FT 表示使用 Large Steps 额外进行 2,000 次微调迭代。CD ($\times10^{-5}$) 和 F1 分数用于评估最终的简化网格。

	CD↓			F1↑		
K	Init	Post-FT	Ours	Init	Post-FT	Ours
2000	0.748	0.718	**0.704**	0.314	0.333	**0.339**
1000	1.528	1.351	**1.259**	0.230	0.254	**0.261**
500	3.971	3.664	**2.891**	0.156	0.172	**0.182**

C.3.3. 渐进式预算调度消融

我们对第4.4 节介绍的渐进式顶点预算调度进行消融实验：移除不进行简化的初始优化（预热，$t_w = 0$）或渐进式顶点预算缩减（递增阶段，$t_r = 0$）。我们采用表1 中 $K = 1000$ 的 FlexiCubes 重建设置，并以应用于相同基础重建的 QEM 后处理结果为参照，根据严重的表面劣化或过多的内部壳层伪影来定义失败。表S6 报告了 25 个失败案例

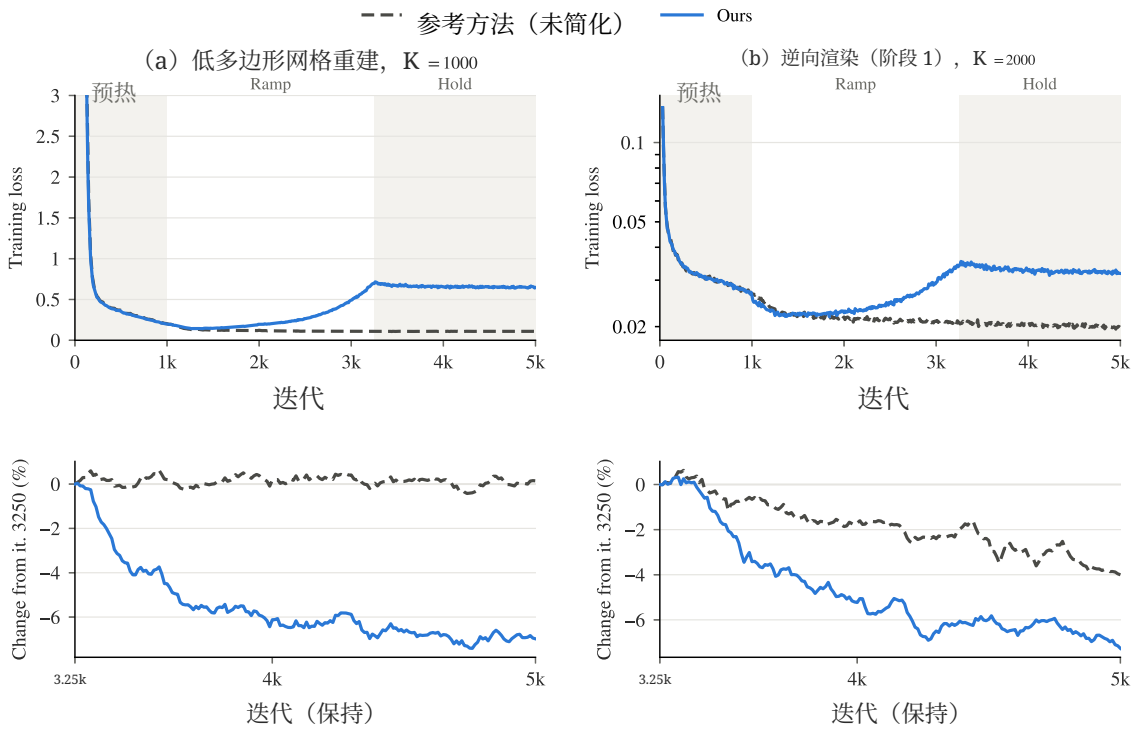


图 S2. 未简化参考方法与 DIFFSIMP的训练损失曲线，采用形状或场景间的几何平均数进行汇总。(a) 对 $K = 1000$ 个形状进行低多边形网格重建（表 1）。(b) 在 $K = 2000$ 个 NeRF合成场景上使用 nvdiffrec 的阶段 1（表 4）。上：在当前顶点预算下评估我们的损失。下：保持阶段内相对于第 3250 次迭代的损失变化率。

表 S6。 $K = 1000$ 上 79 个 FlexiCubes 形状的预算调度消融实验。CD（$\times 10^{-5}$）、F1 分数和简化率在最大连通分量（LCC）上进行评估。失败案例统计的是：相较于对相同稠密重建应用 QEM 后处理，出现破损表面或过多内部壳层的形状。对于失败计数，排除从未简化的 FlexiCubes 重建中继承的预先存在的内部壳层。共同成功子集包含三个调度方案均成功的 50 个形状。

调度方案	倒角距离 mean ↓	倒角距离 med ↓	F1↑	简化率(%) ↑	失败数↓
全部 79 个形状					
不使用预热（$t_w = 0$）	1,321.681	0.745	0.262	1.52	25
不使用递增阶段（$t_r = 0$）	528.277	**0.742**	**0.266**	4.37	8
Ours	**1.259**	0.757	0.261	**0.62**	**0**
共同成功子集					
不使用预热（$t_w = 0$）	1.254	**0.524**	0.347	0.99	–
不使用递增阶段（$t_r = 0$）	1.115	0.526	**0.350**	1.03	–
Ours	**1.065**	0.532	0.343	**0.77**	–

不使用预热阶段和不使用递增阶段的 8 种情况，与采用完整调度方案的情况进行比较。在优化早期，随机初始化的符号距离函数可能生成高度碎片化的网格，并产生内部壳层伪影。这类几何结构会给简化带来问题：边折叠无法合并不连通分量，而与外表面相连的内部壳层也无法通过最大连通分量筛选移除。这些观察结果促使我们引入预热阶段，使上游几何在开始简化前变得更加连贯。不使用递增阶段时，在预热后突然施加目标预算会导致破损表面，并使 SI 从 0.62% 增至 4.37%（7。0×）。尽管不同调度方案下的中位数 CD 和 F1 分数仍具有可比性，但我们的完整调度方案避免了两组消融实验中观察到的严重几何失效。图 S3 展示了具有代表性的失效案例。

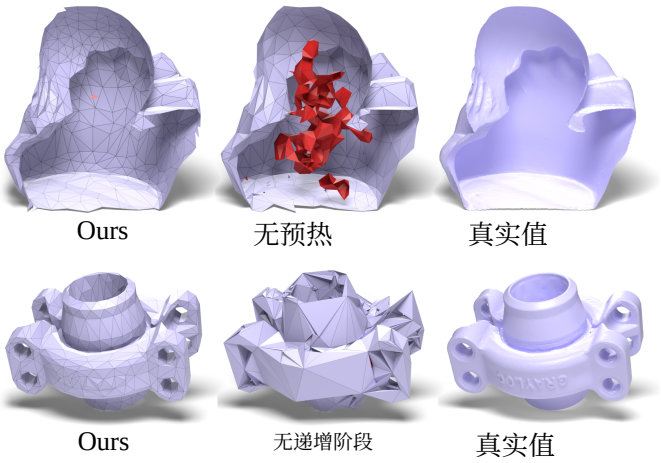


图 S3. 预算调度消融中的典型失败案例 $K = 1000$ （表 S6）。上：内部壳层与外表面保持连通，因此被保留在最大连通分量中，如剖视图所示。下：重建表面在预算骤减时破裂。红色部分突出显示内部面。

D. 其他逆向渲染结果

D.1. nvdiffrec 详细结果

表 S7报告了表 4 中使用 nvdiffrec 进行逆向渲染实验时各场景的峰值信噪比 $K = 2000$。Reference 表示基础 nvdiffrec 流程生成的未简化重建。图S4 展示了八个 NeRF合成场景中 Reference 和我们的方法的完整图像渲染结果，以及对应的真实图像。

表 S7。在 $K = 2000$ 下，每个 NeRF-synthetic 场景的新视角峰值信噪比（dB），对应表 4。Reference 表示基础 nvdiffrec 流程生成的未简化重建结果。标记为 * 的方法会简化最终的带纹理参考结果；其在表 4 中的阶段 1 数值继承自 Reference，故此处省略。

方法	椅子	鼓	榕树	热狗	Lego	Mat.	Mic	Ship	Mean
阶段 *1*									
参考	28.97	23.18	27.93	30.44	26.52	26.04	28.58	23.35	26.88
低分辨率	27.35	19.68	21.68	27.41	22.06	22.78	27.18	21.72	23.73
QEM	28.18	19.65	20.14	29.98	21.16	21.89	28.12	20.69	23.73
我们的方法（仅前向）	28.60	20.47	19.63	30.10	22.97	22.18	28.38	21.11	24.18
Ours	**28.73**	**21.41**	**23.29**	**30.56**	**24.65**	**24.21**	**28.51**	**23.13**	**25.56**
完整运行									
参考	31.32	24.59	30.96	32.70	29.24	27.19	30.47	25.71	29.02
TQEM*	24.90	19.06	21.08	27.71	19.46	20.59	27.45	19.53	22.47
ADS*	29.09	21.99	24.54	30.49	23.34	23.62	29.42	23.18	25.71
低分辨率	29.60	20.67	24.26	29.37	22.80	24.71	29.38	23.66	25.56
QEM	30.34	20.52	22.39	31.31	22.49	24.16	30.23	21.97	25.43
我们的方法（仅前向）	**30.72**	22.12	23.29	31.35	23.28	24.50	30.20	22.81	26.03
Ours	30.42	**22.79**	**27.68**	**31.55**	**25.68**	**25.24**	**30.40**	**24.56**	**27.29**

D.2. 使用 ExMesh 进行场景重建

为评估 DIFFSIMP 在不同逆渲染流水线中的通用性，我们将其集成到 ExMesh 中，并使用作者发布的实现和实验设置，在 $K = 1000$下评估 DTU 数据集中的 15 个扫描数据。ExMesh 通过自适应拓扑更新联合优化显式网格几何和 UV 纹理。我们保留其初始化、损失函数、逐扫描损失权重以及 10,000 次迭代的优化设置。我们遵循第 5节中的相同渐进式顶点预算调度，其中 $t_w = 0.2T$ 和 $t_r = 0.45T$。遵循 ExMesh 的图集重建调度，我们重建简化网格的 UV 图集

在第 2000、4000 和 6000 次迭代时进行，并在训练结束后再进行一次更新。在这些更新之间，通过最近表面纹理映射维持 UV 对应关系。与 nvdiffrec 不同，ExMesh 在简化后没有单独的纹理细化阶段。我们将未经修改的 ExMesh 流程输出的最终稠密纹理网格作为参考。因此，我们以 TQEM 和 ADS 作为纹理感知后处理基线，并将二者都应用于该参考网格。对于 ADS，我们遵循第 A.2节中的细化设置。

如表 S8所示，我们的方法在全部 15 个扫描数据上均取得最低的倒角距离，并在 15 个扫描数据中的 14 个上取得简化方法中的最高峰值信噪比。图 S5 进一步展示了其对平面和尖锐几何特征更好的保留效果。

表 S8。在 $K = 1000$ 条件下对 15 个 DTU 扫描数据进行 ExMesh 重建。参考表示未经修改的 ExMesh 流程在后处理简化之前生成的未简化结果。我们报告训练视图上的官方 DTU 倒角距离（mm）和峰值信噪比（dB）。参考网格平均包含 103,423 个顶点，而所有简化网格均包含 1,000 个顶点。

Scan	24	37	40	55	63	65	69	83	97	105	106	110	114	118	122	Mean
CD↓																
参考	0.502	0.697	0.331	0.359	0.784	0.654	0.562	1.116	0.976	0.611	0.479	0.758	0.320	0.396	0.405	0.597
TQEM	0.702	0.833	0.420	0.473	0.823	0.839	0.829	1.155	1.040	0.667	0.738	0.810	0.458	0.637	0.561	0.732
ADS	0.749	0.815	0.415	0.431	0.851	0.809	0.800	1.217	1.054	0.704	0.695	0.824	0.417	0.596	0.543	0.728
Ours	**0.591**	**0.727**	**0.353**	**0.413**	**0.782**	**0.734**	**0.649**	**1.129**	**0.965**	**0.634**	**0.602**	**0.710**	**0.386**	**0.495**	**0.464**	**0.642**
峰值信噪比↑																
参考	22.97	20.48	24.00	26.68	27.65	25.59	25.47	32.50	23.99	27.55	29.86	24.02	26.32	32.75	33.11	26.86
TQEM	17.80	17.20	20.20	21.53	24.87	20.45	23.31	30.10	21.59	24.47	24.90	23.10	22.75	27.34	27.84	23.16
ADS	21.10	19.43	22.87	24.73	27.11	24.12	24.78	31.95	23.41	26.86	27.86	**23.76**	25.42	30.60	31.10	25.67
Ours	**21.88**	**20.00**	**23.14**	**25.23**	**27.27**	**24.86**	**25.03**	**32.04**	**23.56**	**26.93**	**28.42**	23.73	**25.77**	**31.18**	**32.00**	**26.07**

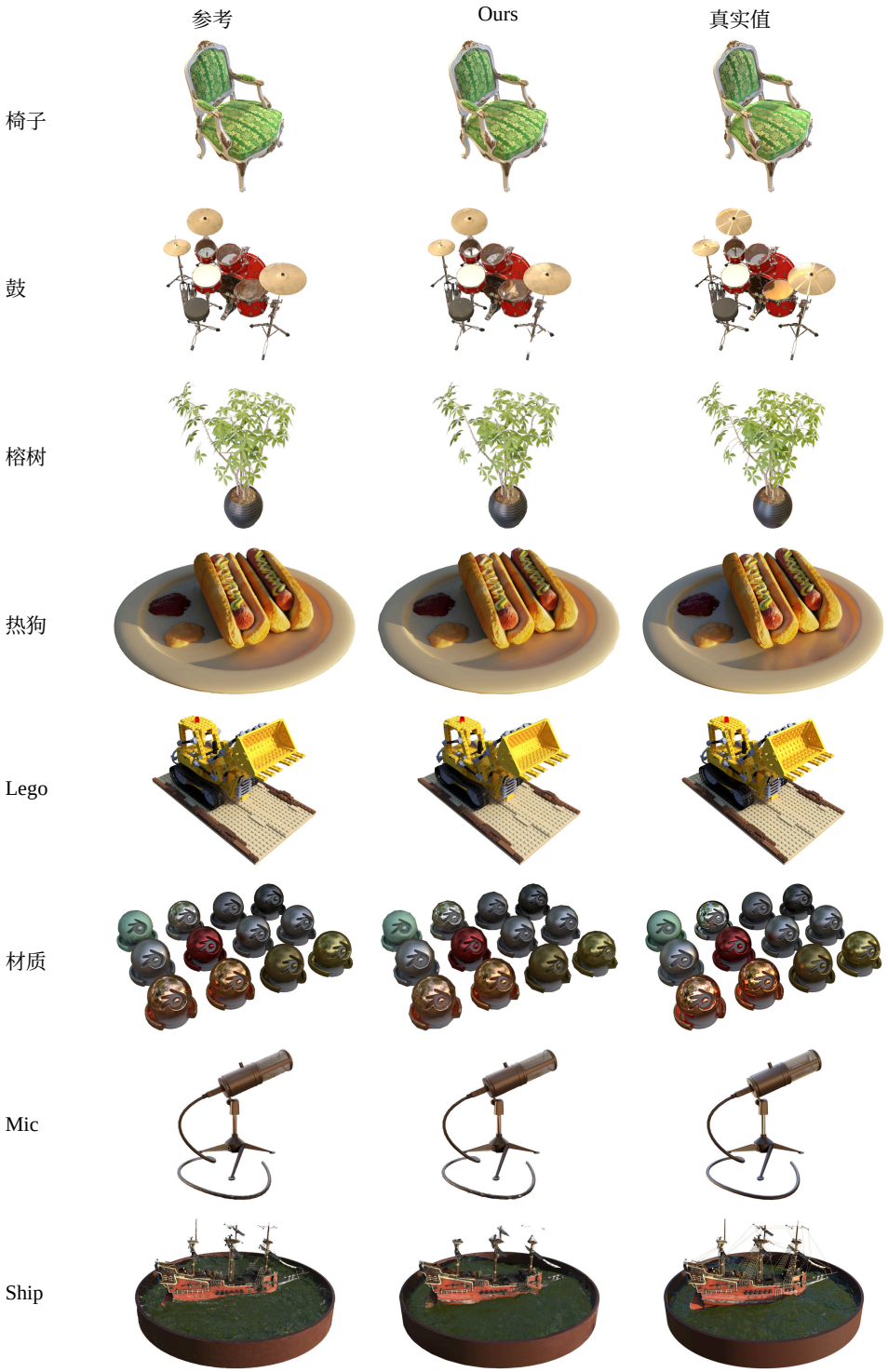


图 S4. 完整运行后，对八个 NeRF合成场景使用 nvdiffrec 进行逆向渲染的定性比较，见 $K = 2000$ （表 4）。展示了由参考方法和我们的方法渲染的测试视图，以及真实图像。参考方法的平均顶点数为 51,512，我们的方法为 2,000。参考方法指基础 nvdiffrec 流程得到的未简化重建。

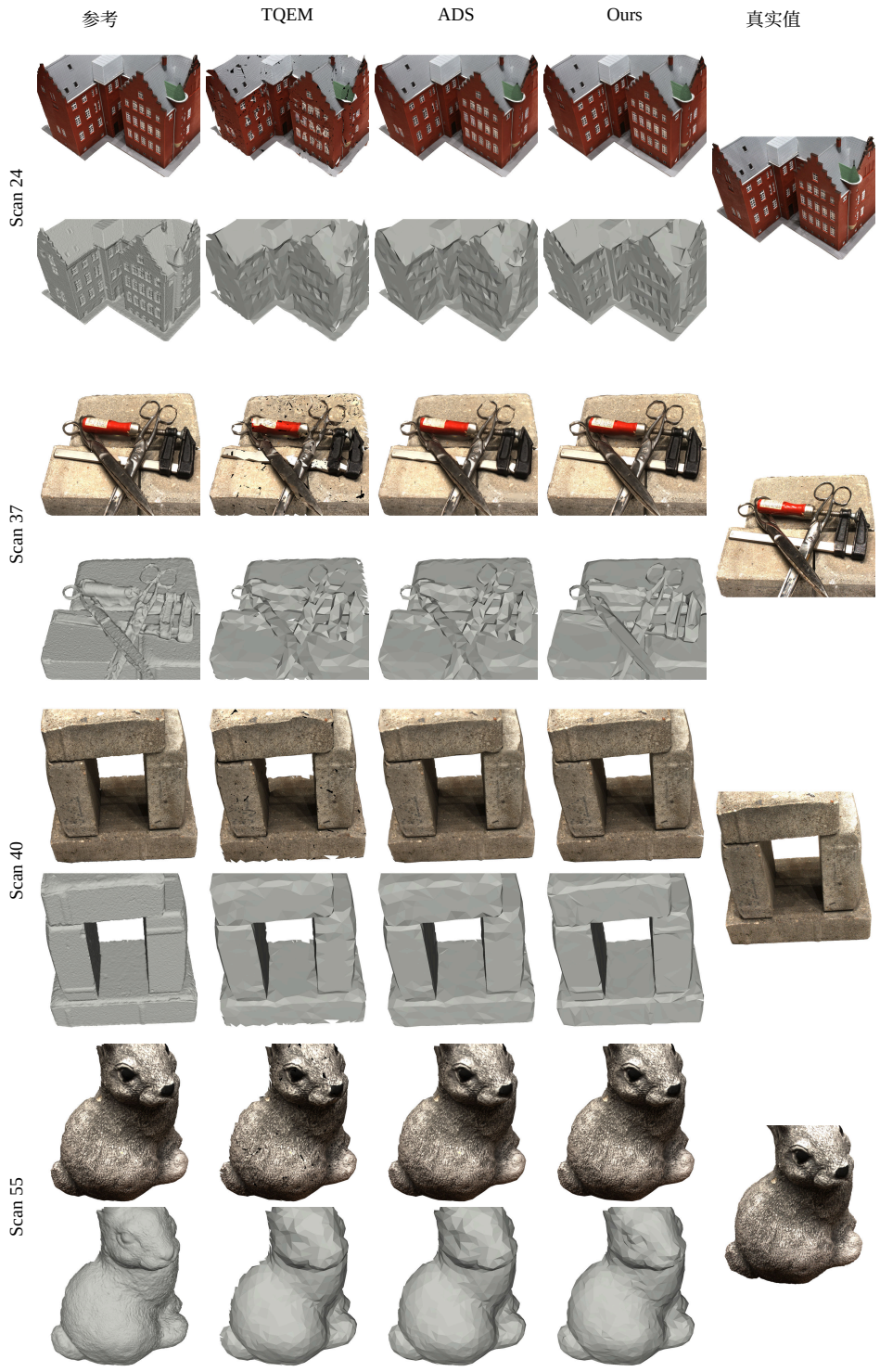


图 S 5 24, 37, 40, 55 K = 。在（表 S）中，对 1000 8 ExMesh 在扫描数据和 DTU 数据集上的场景重建进行定性比较。参考表示由原始 ExMesh 流程重建的稠密纹理网格。上方各行显示纹理渲染图，下方各行显示网格几何，真实值显示对应的输入照片。

E. 局限性与未来工作

尽管 DIFFSIMP 是一个无需训练的即插即用可微网格简化层，但其有效性取决于优化过程中生成的中间网格的质量。当上游网格生成器产生碎片化几何体或内部壳层伪影时，在这些缺陷得到解决之前应用简化可能会加剧这些问题，并可能导致优化失败。在进行激进简化时，塌缩映射的一对多性质可能使中间几何的某些部分受到的约束较弱，从而可能产生悬浮伪影。

未来工作。 我们的逆向渲染实验展示了几何与外观的联合优化。然而，UV 对应关系由独立于我们简化器的流程处理。通过采用具备 UV 感知的折叠代价和接缝感知的属性传播机制，扩展我们基于几何的前向简化器，可以减少对独立 UV 处理的依赖，并促进其集成到更广泛的纹理网格优化流程中。