Sketch-guided Cage-based 3D Gaussian Splatting Deformation

Tianhao Xie[1] Noam Aigerman[2,3] Eugene Belilovsky[1,3] Tiberiu Popa[1]
[1]Concordia University, Montréal, Canada [2]Université de Montréal, Montréal, Canada
[3]Mila, Montréal, Canada

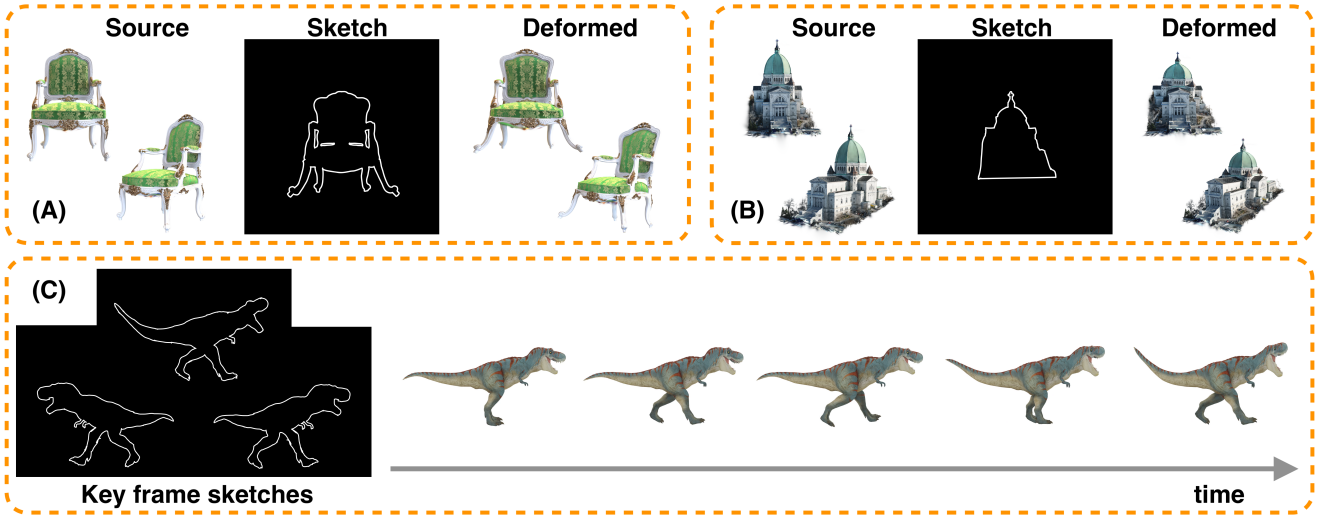

Figure 1. Deformations of 3D Gaussian splats (GS) using our sketch-guided deformation method. Given a 3D GS scene, the user can deform the 3D GS by drawing a deformed silhouette sketch of a single view. (A) and (B) show examples with synthetic data and real-world large-scale data, respectively. We can also produce an animation of a of 3D GS by few (≥ 2) keyframe sketches, as shown in (C).

Abstract

3D Gaussian Splatting (GS) is one of the most promising novel 3D representations that has received great interest in computer graphics and computer vision. While various systems have introduced editing capabilities for 3D GS, such as those guided by text prompts, fine-grained control over deformation remains an open challenge. In this work, we present a novel sketch-guided 3D GS deformation system that allows users to intuitively modify the geometry of a 3D GS model by drawing a silhouette sketch from a single viewpoint. Our approach introduces a new deformation method that combines cage-based deformations with a variant of Neural Jacobian Fields, enabling fine-grained control. Additionally, it leverages 2D diffusion priors and ControlNet to ensure the generated deformations are semantically plausible. Through a series of experiments, we demonstrate the effectiveness of our method and showcase its ability to animate static 3D GS models as one of its applications.

1. Introduction

Editing of 3D models and shapes often arises in computer graphics, computer animation, and geometric modeling. This editing is usually carried out by *deforming* the 3D models. In this work, we aim to provide such editing-through-deformation capabilities for one of the most promising novel representations of 3D geometry - 3D Gaussian Splatting (GS) [17], which offers great real-time novel view rendering ability and better photorealistic reconstruction than previous methods.

Similarly to other geometric representations such as NeRFs [25] and triangle meshes, various control mechanisms were proposed for editing GS, such as text prompts [4, 5, 35, 37] and video priors [21, 29]. Unfortunately, these types of controls are designed for broad, high-level edits (ones within the capabilities of a novice user), without enabling fine-grained control over the deformation. On the other hand, some investigation has been made into more direct, geometrical editing, e.g., via physics-based simulation [39] - this again offers limited editing capabil-

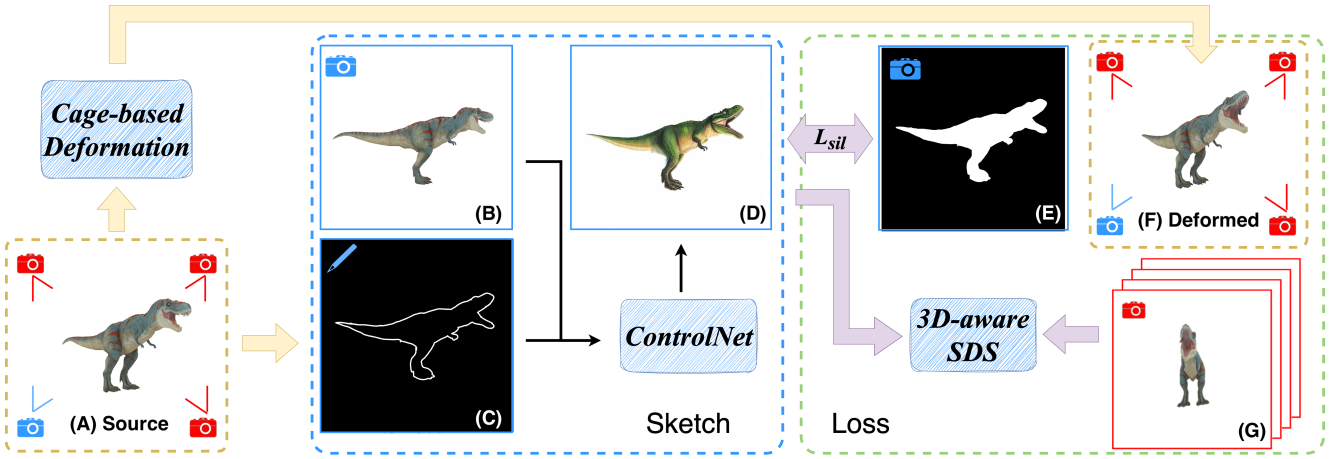

Figure 2. Overview of our method. We start with a 3D GS model (A). Users can select one specific sketch view (B) to draw an edited silhouette sketch (C). This user-drawn sketch (C) and rendering from the chosen view (B) are fed into ControlNet [42] resulting in a deformed reference image (D). We then optimize the cage of the source 3D GS model (A) using two losses: (1) a silhouette loss L_{sil} between the silhouette (E) of the deformed GS model (F) and the silhouette of the generated reference image (D), and (2) a 3D-aware SDS loss from 4 random views (G) conditioned on the reference image (D).

ities.

The problem in providing fine-grained geometric control lies in the GS representation, which is made up of an unstructured array of different 3D Gaussians whose aggregation forms visuals when splatted onto a 2D canvas. This often leads to a global dependence between different Gaussian - changes the position of one and the plausibility of the scene is ruined. Hence, it is difficult to provide the ability to perform local edits while maintaining the integrity of the resulting visuals.

To tackle these issues, in this work, we introduce the first sketch-guided 3D GS deformation system, which enables the user to intuitively interact with a simple 2D sketch of the object, and by which induces a 3D deformation of the Gaussians. To achieve this, we propose several technical contributions: 1) geometrically, to ensure the deformations produced are regulated, we propose a novel deformation framework for GS, based on cage-based deformations, which are in turn controlled by deformation Jacobians [1]. 2) Semantically, we leverage ControlNet [42] and Score distillation Sampling(SDS) [28] to ensure a semantically-meaningful, plausible 3D GS deformation. Together, these two contributions enable the user to deform the shape freely, while preserving its integrity, see Figure 5.

Our experiments verify our method's ability to provide deformation of Gaussian splats.

2. Related Work

2.1. Sketch-based 3D shape Editing

Sketching is a widely used modeling paradigm in geometric modeling and computer graphics. Early methods such as Teddy [14] and Fibremesh [27] construct smooth shapes guided by user-specified 2D sketches. After generating the initial shape, deformations can be done by drawing reference strokes or deforming the generated curve.

Some works [18, 20, 26, 45] focus on the sketch-based deformation that uses individual drawn strokes as a deformation clue. (For a thorough review of the 3D shape modeling, we refer to [6].) All these methods combine the user constraints from sketches with some geometric regularizer such as the Laplacian. Whereas, these geometric energies were designed to preserve certain properties, such as smoothness, and couldn't take into account the semantic information of the object. This can sometimes lead to unnatural deformation, e.g. bending the straight shape.

More recently, data-driven methods were applied to sketch-based 3D shape modeling [3, 9, 22, 24]. Some works [3, 9] trained neural networks to generate 3D meshes conditioned on the input sketch. These methods always need large-scale datasets to train the networks and the editing can only be done for generated shapes. For more general sketch-based editing, [22, 24] edited the shapes (represented by Neural Radiance Field) by 2D sketch matching and employed Score Distillation Sampling(SDS) [28] to produce natural-looking editing that satisfied the semantic of the text prompt.

2.2. SDS in 3D shape Editing

Score Distillation Sampling(SDS) was first introduced in [28] as a 3D shape generation method based on 2D diffusion prior. The core idea of SDS is to make the renderings of generated 3D shapes look natural from any random viewpoint. Since it is an image-based score, it can be ap-

plied to 3D editing of any representation, e.g. triangular mesh [38, 41], NeRF [24] and 3D GS [21]. However, the original SDS used a 2D image diffusion model with only 2D knowledge, leading to the view inconsistency problem of SDS. Recently, by using 3D data in training the diffusion model, Multi-view diffusion [13, 23, 30] was introduced to generate 3D shapes with better geometric consistency. By replacing the image diffusion model in the SDS with multi-view Diffusion, the cross-view consistency can be improved in the 3D shape editing [22, 29].

2.3. Editing 3D Gaussian Splatting

Neural Radiance Field(NeRF) [25] and 3D Gaussian Splating(3D GS) [17] are new 3D representations designed for novel-view synthesis that the 3D scene can be reconstructed from a set of images. Some work has been done for NeRF editing [11, 24, 31, 44]. Since our work focuses on sketch-based deformation of 3D GS, we will talk in more detail about editing 3D GS. [35] introduce a method that edits the 3D Gaussian Splatting(GS) scene with text instruction, powered by LLM and 2D image diffusion prior, which can achieve object texture editing and environment changing. [5] also uses 2D image diffusion prior as the guidance of the editing but introduced the Hierarchical GS to improve the editing quality. It enables object removal and addition by employing inpainting techniques.

Instead of guiding the editing by 2D image diffusion prior, [4, 37] introduced methods that edit the rendered image of original 3D GS from multi-views with consistency control, and fit the changes in the edited images to 3D GS directly, which improve the efficiency and quality of editing significantly.

Some works focus on the deformation of the 3D GS. Align-Your-Gaussians(AYG) [21] and DreamGaussian4D [29] introduced methods that animate a static 3D GS object to a 4D GS sequence. AYG [21] employs Video Diffusion prior to drive the temporal deformation between frames and using 2D Diffusion prior for every frame respectively to constrain the deformation validly. Instead of using Video Diffusion prior, DreamGaussian4D [29] uses a reference video in one specific view to animate the static 3D GS and applies 3D-aware Score Distillation Sampling(SDS) to propagate the deformation in every frame. The reference video was generated from an image-to-video Diffusion model based on the rendered image of static 3D GS from that specific view. Compared to AYG, DreamGaussian4D is more efficient but limited by the generation quality and universality.

PhysGaussians [39] seamlessly integrates physically grounded Newtonian dynamics within 3D GS to achieve high-quality novel motion synthesis. It adapts the Material Point Method(MPM) to 3D GS which enriches 3D GS with meaningful kinematic deformation and mechanical stress attributes.

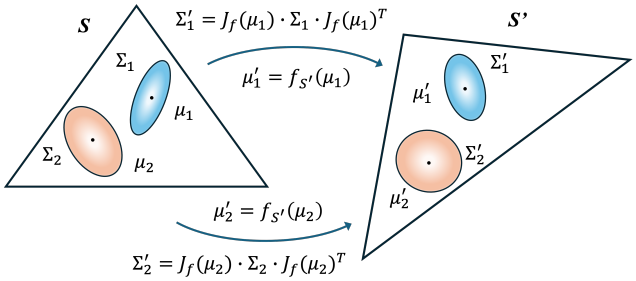

Figure 3. A simplified 2D illustration of cage-based (in this case, a triangle) deformation of Gaussian splats. S and S' are the original and deformed cage, respectively. μ_i and Σ_i are the centroid and covariance of the original Gaussian splats, and μ_i' and Σ_i' of the deformed ones. $f_S(\mu_i)$ is the cage interpolation function and $J_f(\mu_i)$ is the Jacobian matrix of the interpolation function f.

SuGaR [8] employs new training energies based on original 3D GS [17] that produce 3D GS with better surface alignment and more even density distribution. These new properties make it possible to extract mesh from 3D GS only using traditional methods, such as Poisson reconstruction [16]. Given the extracted mesh, The 3D GS can be bound to the mesh and deformed by mesh deformation algorithms, such as ARAP [32]. Similar to SuGaR, [7] also proposes to bind the 3D GS kernels to the mesh, which is reconstructed by the existing methods from the input multi-view images directly. However, the result significantly depends on the extracted mesh, which can fail in scenes with complex geometry and transparent components.

3. Method

We next detail the various components of our framework, starting with an overview of the representation of 3D Gaussians [17], moving on to our cage-based deformation through jacobians, and concluding with applying this deformation technique using sketches and Score Distillation Sampling [28].

3.1. 3D Gaussians

A $3D$ Gaussian is defined by a full $3D$ matrix $\Sigma \in \mathbb{R}^{3\times3}$ and a centroid $\mu \in \mathbb{R}^3$, defining the density of the Gaussian:

$$G(x) = e^{-(1/2)(x-\mu)^T\Sigma^{-1}(x-\mu)}. \quad (1)$$

Given a diagonal scaling matrix $S \in \mathbb{R}^3$ and rotation matrix $R \in SO(3)$, the corresponding Σ is constructed as:

$$\Sigma = RSS^TR^T, \quad (2)$$

with S and R being the variables that are optimized during the reconstruction of the scene using $3D$ Gaussians. We consider a collection of such Gaussians, $(\Sigma_i, \mu_i)_{i=1}^n$.

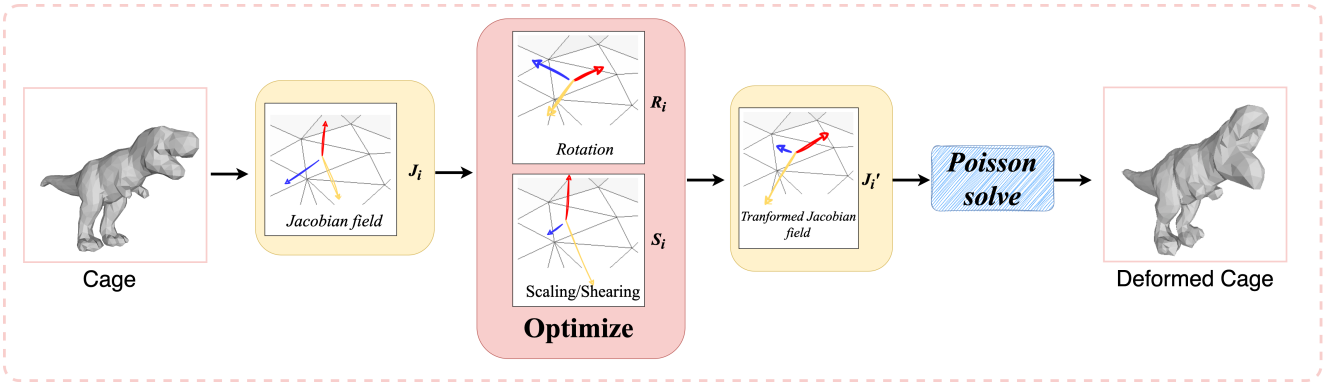

Figure 4. Controlling Cages via decomposed Neural Jacobian Fields. Instead of optimizing the NJF of the cage J_i, we optimized the rotation component R_i and the stretch component S_i of the NJF's transformation respectively. The deformed NJF can be computed easily by applying the transformation to the original NJF. Finally, the deformed cage was obtained by solving the Poisson equation.

3.2. A Regularized Framework for Deforming 3D Gaussians

Deforming 3D Gaussians entails assigning a new centroid position μ and a new covariance matrix Σ to each Gaussian. However, we wish to regulate the space of possible deformations, to avoid Gaussians floating apart, and exposing only meaningful deformations of the object the Gaussians represent. Towards this goal, we design a novel, tailor-made deformation scheme, incorporating two components: 1) a cage-based deformation [2], tailored to Gaussian Splats; 2) a method to control this cage deformation, inspired by neural jacobian fields [1]. We detail these two components next.

3.2.1. Cage-Based Deformation of Gaussians

A cage-based deformation uses a triangular mesh S with vertices V and triangles T. The mesh is deformed into another state, S', by moving its vertices. By that, the mesh defines a deformation $f_{S'} : \mathbb{R}^3 \to \mathbb{R}^3$, mapping every point in $\mathbb{R}^3$ as a function of the positioning of the vertices of S'. This enables exposing a more meaningful, low-dimensional deformation space, controlled by the cage's vertices. There are many different approaches to define this function; we choose to use [2] - see the supplementary material for the full details.

We next define the deformation of the Gaussians w.r.t the cage's deformation: the deformed position μ' of each centroid μ is defined as mapping it through the cage deformation:

$$\mu' = f_{S'}(\mu). \tag{3}$$

Similarly, to modify the covariance matrix Σ, as is standard, we use a local linear approximation of the deformation, via the Jacobian matrix $J_f = \nabla f_{S'}$, evaluated at the centroid μ. Then, the deformed covariance matrix Σ' can be expressed as:

$$\Sigma' = J_f R S S^T R^T J_f^T. \tag{4}$$

3.2.2. Controlling Cages via Neural Jacobian Fields

While the cage-based deformation already regularizes the deformation of the Gaussians, We found that optimizing directly on the cage vertices tends to produce entanglement and unsmooth cage which can lead to artifacts in the rendering of $3D$ GS, as in figure 8. We thus control the cage vertices' position using Neural Jacobian fields (NJF) [1]. In short, NJF positions a mesh's vertices V' from given per-triangle matrices $M_i \in \Re^{3\times3}$, by minimizing the squared error between those matrices and the mesh's per-face *Jacobians* $J_i \in \Re^{3\times3}$, defined as

$$J_i = V' \nabla_i^T, \tag{5}$$

where ∇_i^T is the gradient operator of triangle t_i. The solution to this least-squares problem is achieved via *Poisson's equation*, amounting to solving a single sparse linear system, which is easily implementable in a differentiable pipeline.

We represent jacobians in a manner better accommodating for optimization: Suppose the initial per-face Jacobian is $J_i \in \Re^{3\times3}$ for face i, and the deformed per-face Jacobian is $J_i^{'}$. There is a linear transformation $T \in \Re^{3\times3}$ s.t. $J_i^{'} = TJ_i$. By polar decomposition, this transformation matrix can be decomposed to

$$T = R \cdot S, \tag{6}$$

where R is an orthogonal matrix and S is the stretching component (symmetric semi-positive definite matrix) of the transformation.

As shown in Figure 4, we express the deformation of the cage as a per-triangle rotation and stretching in the Jacobian space. After the cage was deformed, the deformation of $3D$ GS was computed by equations 3 and 4.

We represent the rotational component using the smooth 6-DoF representation illustrated in [43] and the stretching

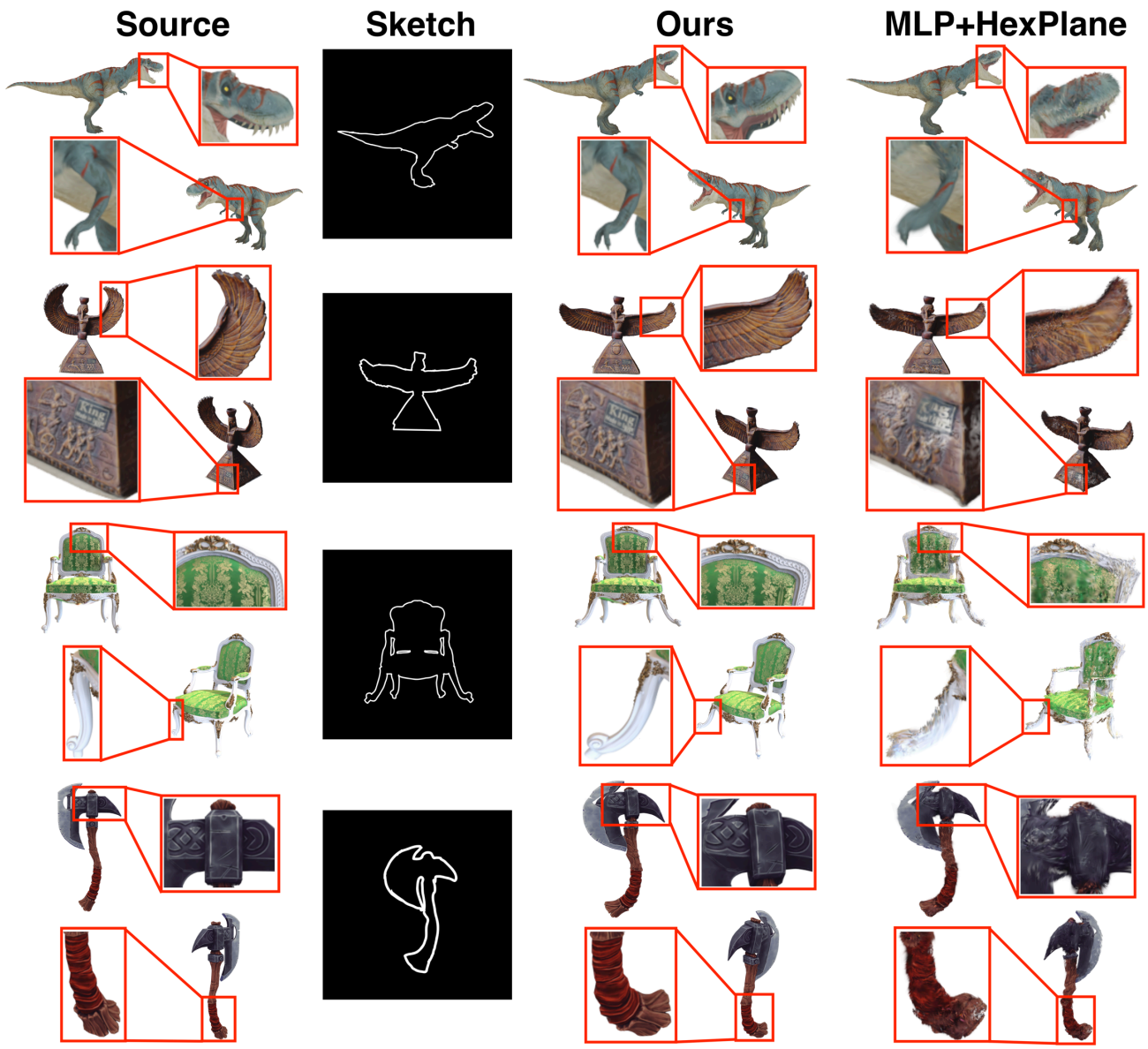

Figure 5. Comparison of using Cage or MLP+HexPlane [29] to represent deformation of the $3D$ Gaussians. MLP+Hexplane can cause severe fuzzy rendering of the deformed 3D GS because of the lack of geometry regularization. Our cage-based method produces almost lossless deformation regarding the rendering visual quality.

component as a symmetric 3 by 3 matrix with 6 degrees of freedom. Since we actually have more freedom after decomposing the Jacobian field for optimization, we found that it can produce lower energy values than optimizing the Jacobian field directly.

3.3. 3D-aware Score Distillation Sampling

We use Score Distillation Sampling(SDS) [28] to guide our deformation to be plausible. In short, SDS renders a 3D model from different view points, and then leverages a trained 2D image diffusion model, and backpropagates the diffusion process through the differentiable renderer, to the degrees of freedom of the 3D model. we further make use of a 3D-aware image diffusion model [23] to enable a more accurate 3D consistency of the generated images. In short, this model can be conditioned on a specific viewing direction, and produces more consistent images of the same object from different view point. See Supplemental for the full details.

3.4. Sketch-guided 3D GS Deformer

The overview of the editing pipeline is shown in Figure 2. The user first selects a viewpoint vp (blue camera). This viewpoint is used to extract a silhouette of the object. We render the $3D$ GS from viewpoint vp to get an image I_{vp} (Figure 2 B). The user additionally deforms the silhouette into S_{vp} (Figure 2 C). To obtain the deformed image I_{vp}^{CN} (Figure 2 D) guided by the user's sketch, the rendering I_{vp} is fed into an image-to-image diffusion model conditioned on the S_{vp} by using ControlNet [42]. Our loss measures the silhouette difference $\mathcal{L}_{sil}$ between I_{vp}^{CN} and I_{vp}^{def} (Figure 2 E), the rendering of deformed 3D GS from viewpoint vp:

$$\mathcal{L}_{sil} = \|I_{vp}^{CN} - I_{vp}^{def}\|_2^2. \tag{7}$$

We chose to only penalize the silhouette and not the full RGB deformed image, as our experiments showed the texture of the objects can be otherwise changed drastically.

To keep the deformation natural in all views, we apply 3D-aware SDS on randomly sampled views (red camera) in every iteration. Thus the final gradient used during optimization is:

$$\nabla\mathcal{L}_{total} = \alpha\nabla\mathcal{L}_{sil} + \nabla\mathcal{L}_{SDS}, \tag{8}$$

α is set to 10000 for all examples. When optimizing the 3D GS by the objective function 8, the 3D GS is deformed by the differentiable Cage-based block as shown in section 3.2.

3.5. Implementation details

The cage is generated automatically by first extracting a mesh from the GS model using the coarse stage of the SuGaR [8] method. This mesh is very large, sometimes it is not a closed manifold and it has many fold-overs so it would not be suited for a cage. Therefore, we compute a triangulated offset surface using a function from Libigl [15] that applies marching cubes on a grid of signed distance values from the input triangle mesh. The resulting mesh is a close manifold triangular mesh with an adjustable number of vertices depending on the model size and level of detail desired. We used StableDiffusion-XL as the diffusion model of the ControlNet. For each deformation, we optimized for 2000 iterations, with a learning rate of 0.002, and optimized by Adam optimizer [19]. Except for the reference view, 4 random views were sampled for the 3D-aware SDS. The diffusion model used in 3D-aware SDS is zero-1-to-3 XL [23]. All experiments were run on a single Nvidia RTX A6000 GPU. The running time was related to the number of Gaussian splats and cage resolution in different scenes. For a scene with 268k gaussians and 376 cage vertices, the running time is 12 minutes. For a scene with 90k gaussians and 1343 cage vertices, the running time is 9 minutes. Thus, the running time is primarily influenced by the number of Gaussians. For large-scale scenes containing more Gaussians (e.g., 500k), the computation reaches a bottleneck, resulting in slower performance (around 40 minutes). More efficient diffusion models is a growing area of research and our method would be directly accelerated by the many methods that are being developed [12, 33].

Figure 6. The comparison of our method with mesh-binding method GaussianMesh [7]. From left to right: 1) the deformed proxy mesh obtained by our sketch-guided pipeline for GaussianMesh. 2) the deformed 3D GS by GaussianMesh. 3) deformation result obtained by our cage-based method. 4) undeformed original 3D GS.

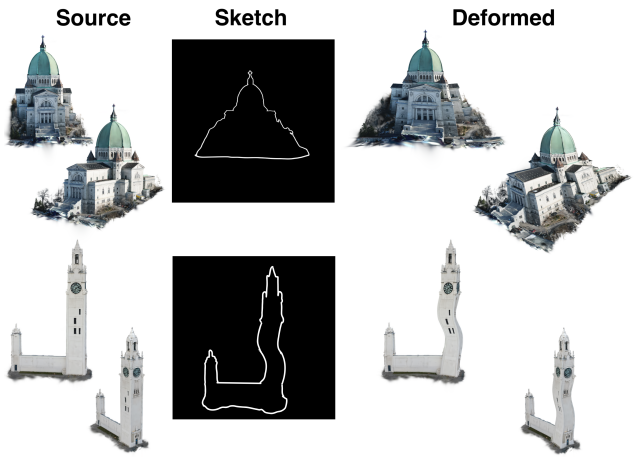

Figure 7. Deforming UAV-captured real-world large scene. We converted the UAV-captured images of a great Oratory and a clock tower into 3D GS scenes and conducted experiments, as shown in the top and bottom rows respectively.

4. Experiments

4.1. Results

We tested our method on various 3D objects, from human-made objects to animals and humans, as shown in Figure 1, 5, and 9. It shows that our method can deform the objects precisely with the guidance of the sketch and produce natural-looking results. These 3D shapes were collected from the SketchFab and TurboSquid and transferred into a 3D GS scene. The 3D shapes were originally mesh and converted into 3D GS by training from 100 sampling images from random views of the shape. Moreover, we also explored the ability of our method to deform large-scale real-

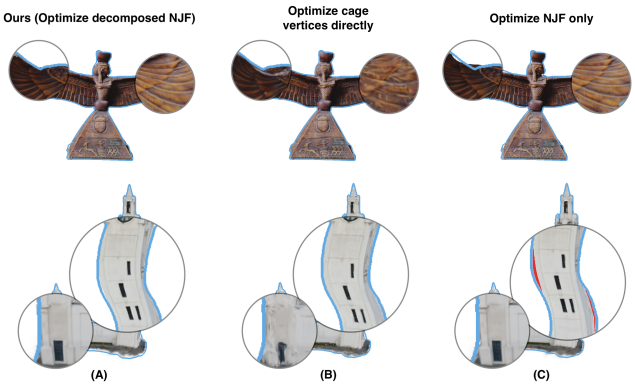

Figure 8. Ablation study on cage optimization. The blue line is the target silhouette and the the gap area between the target silhouette and the actual silhouette is marked in red. (A) Our final result uses our proposed modifications of NJF. (B) Optimizing the cage vertices directly. (C) Optimizing the original Neural Jacobian Field (without our modifications)

Table 1. Quantitative comparison: we report the relative CLIP-IQA score [34], and relative Q-align score[36].

	MLP+HexPlane	Ours
Relative CLIP-IQA ↑	75.32%	99.03%
Relative Q-align ↑	78.11%	99.88%

world captured data. We used the UAV-captured dataset of a great Oratory and a Clock Tower to reconstruct the 3D GS scene respectively. As shown in Figure 7.

As one of the important applications of 3D deformation is animation, we also conducted experiments of animating static 3D GS by our method. As shown in Figure 1(C), and the accompanying video, the user can provide multiple input sketches as some key-frame sketches in the animation sequence. Then, by running our method for each key frame and interpolating between the deformed cage of key frames, we can get animations of the static 3D GS.

We first compared our Cage-based deformation for 3D GS with the SOTA method MLP+HexPlane [29]. For comparison, we deform the source 3D GS using our pipeline and render the sketch view of the deformed 3D GS as the reference image of the MLP+Hexplane. Thus, except for the difference in the deformation method, MLP+Hexplane also has extra RGB image guidance in the sketch view when our pipeline is only guided by silhouette. Even in that case, our pipeline can deform the 3D GS with much higher fidelity than using MLP+Hexplane. The qualitative comparison was shown in Figure 5, because of the good space continuity of the cage deformation, the local detail features can be fully preserved, e.g. the pattern on the statue, and the teeth of the dinosaur. Though the Hexplane was applied as a geometric regularizer, the detailed features can still be destroyed and cause fuzzy renderings when using MLP+Hexplane. We quantitatively compare the visual quality of the deformed shapes based on image quality. We use the CLIP-IQA [34], a metric measuring the image quality based on the CLIP score, and Q-align[36], an image quality assessment based on large multi-modality models (mPLUG-Owl2 [40]), to evaluate 8 deformed results. For every deformed 3D GS, 8 views were rendered as the images to be assessed. Since our task is to measure how the image quality was changed after the deformation process, we report the relative score of both CLIP-IQA and Q-align, which is calculated as the metric score of the undeformed renderings divided by the metric score of the deformed renderings. As shown in table 1, the average decreases of the CLIP-IQA score and Q-align score for the MLP+Hexplane method are 24.68% and 21.89 respectively, however, our method almost keeps the same CLIP-IQA and Q-align score as original renderings with less than 1% decreasing. We also conducted a user study with 15 participants comparing the deformation fidelity to using MLP+HexPlane across 6 examples with instruction: which deformed result has better visual fidelity?. Among these 90 comparisons, our method gains 90.0% preference from the participants.

We also include a comparison with mesh-binding GS (GaussianMesh [7]); however, a direct substitution of our cage-based method with their method is not straightforward, as their deformation approach is not implemented as a differentiable process. We perform the comparison with the following setup: 1) Run the mesh reconstruction method [10] to obtain a proxy mesh M_p, which is the base mesh to reconstruct the 3D GS. 2) Run GaussianMesh [7] reconstruction with input M_p to get the 3D GS. 3) Run our pipeline as shown in figure 2 in which the 3D GS was replaced by M_p and regularized by NJF to get a deformed mesh M_p'. 4) Run GaussianMesh Deformation based on the deformed mesh M_p' to get the deformed 3D GS. The result is shown in figure 6. The deformed mesh exhibits severe artifacts, including entanglement and faces inversion, even with the NJF regularization. These artifacts are reflected in the final rendering of the 3D GS. Another drawback of the mesh-binding method is that the reconstruction is based on a reconstructed mesh obtained by other methods [10], which can fail in complex scenes. We show an example of a reconstructed mesh in the Supplementary material.

Additionally, we evaluated the silhouette adherence by the Intersection of Union (IoU) between the alpha rendering of 3D GS and the target silhouette from the sketch view. Among the examples, the average IoU before deformation is 0.654 and after deformation is 0.901.

4.2. Ablation

We tested the effects of two important components in our method, decomposed NJF and 3D-aware SDS. We explored

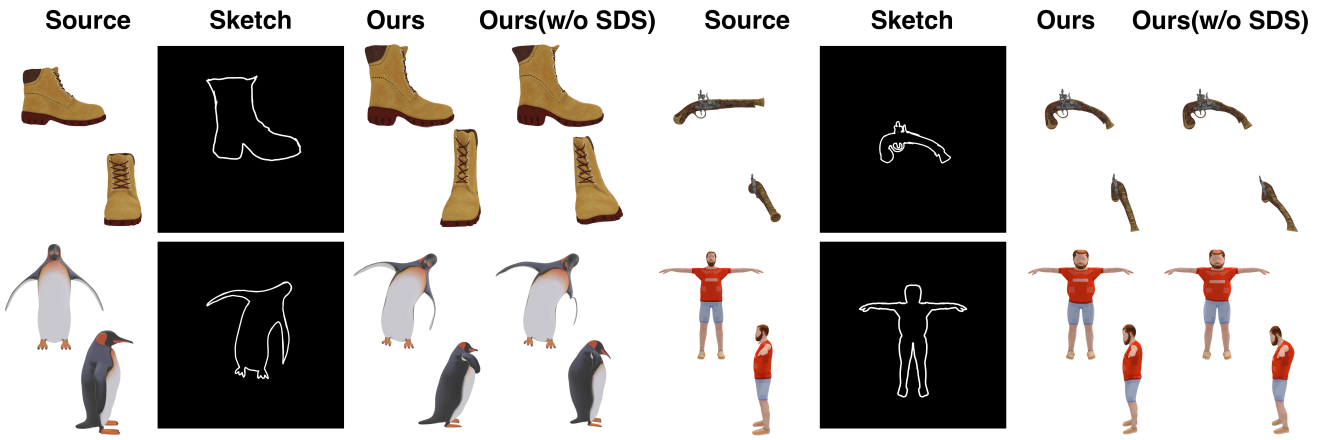

Figure 9. Ablation study of 3D-aware SDS. 3D-aware SDS was used to produce natural-looking deformation in all views. The result shows undesired distortion without 3D-aware SDS.

the difference of optimizing directly on the cage vertices, via NJF and our proposed decomposed NJF. As shown in Figure 8, optimizing directly on the cage vertices can lead to undesired fuzzy rendering, which is caused by entanglement of the cage during the optimization. As shown in Figure 8 (B), the detailed feather of the statue and the door on the side of the clock tower were destroyed. Applying NJF can solve this problem since it works as a geometry regularizer for the cage. However, it can be too strong to fit the sketched silhouette precisely. As shown in Figure 8 (C), the wing of the statue is not fitted on the end and the clock tower is not bent enough. When using decomposed NJF as stated in section 3.2.2, we can obtain a more efficient optimization which leads to a lower $\mathcal{L}_{sil}$ (average of 49.36% decreasing in these two examples), but with same rendering visual quality compared to NJF (Figure 8 A).

We also explored the effect of 3D-aware SDS in our pipeline. As shown in Figure 9, this plays a critical role in preserving undesirable deformation in the views except for the sketch view in the pipeline. For instance, when the pistol was deformed only by $\mathcal{L}_{sil}$, the barrel was bent, which can be fixed by adding the 3D-aware SDS.

4.3. Failure cases and mitigations

Some failures can occur from single view mismatches (e.g. mismatching of left and right legs in a side view). This can be addressed by slightly changing the view (See supplementary material)

5. Conclusion

In this work, we propose a novel sketch-guided 3D Gaussian Splatting deformation framework, which enables intuitive, fine-grained control over the geometry of 3D GS by a single-view silhouette sketch. Geometrically, we designed a novel cage-based deformation tailored to Gaussian Splats and optimized its position using a modified Neural Jacobian Fields formulation. As the rendering of Gaussian Splats overlays intersecting splats on top of each other, deforming these splats can lead to visual artifacts due to misalignment. Our method provides accurate alignment of the splats in the deformed pose that yields crisp rendered results that adhere closely to the input sketch. To ensure semantically meaningful deformation from any viewpoint, our method leverages ControlNet as well as Score distillation sampling.

6. Limitation and Future work

First, our method relies on ControlNet and an image-to-image diffusion model to translate the user-drawn silhouette sketch into a usable constraint for the deformation system. However, the ability of ControlNet to generate a reference image of the deformed object is sometimes limited. Second, although the cage for 3D GS is generated automatically, the process of training the SuGaR model to extract a mesh from the 3D GS still takes several minutes, which limits the system's efficiency.

In future work, we will focus on improving the reliability of the sketch-guided deformation, particularly by enhancing the quality and consistency of the ControlNet-generated reference images. Additionally, making the cage generation process more efficient could enhance the practicality of our method. Exploring these avenues would contribute to both the robustness and efficiency of the system.

7. Acknowledgment

We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC), under funding reference numbers RGPIN-2021-03477 and RGPIN-2024-04605, as well as the support of Fonds de recherche du Québec – Nature et technologies (FRQNT), under funding reference number 365040.

References

[1] Noam Aigerman, Kunal Gupta, Vladimir G Kim, Siddhartha Chaudhuri, Jun Saito, and Thibault Groueix. Neural jacobian fields: learning intrinsic mappings of arbitrary meshes. *ACM Transactions on Graphics (TOG)*, 41(4):1–17, 2022. 2, 4

[2] Mirela Ben-Chen, Ofir Weber, and Craig Gotsman. Variational harmonic maps for space deformation. *ACM Transactions on Graphics (TOG)*, 28(3):1–11, 2009. 4

[3] Alexandre Binninger, Amir Hertz, Olga Sorkine-Hornung, Daniel Cohen-Or, and Raja Giryes. Sens: Part-aware sketch-based implicit neural shape modeling. In *Computer Graphics Forum*, page e15015. Wiley Online Library, 2024. 2

[4] Minghao Chen, Iro Laina, and Andrea Vedaldi. Dge: Direct gaussian 3d editing by consistent multi-view editing. *ECCV*, 2024. 1, 3

[5] Yiwen Chen, Zilong Chen, Chi Zhang, Feng Wang, Xiaofeng Yang, Yikai Wang, Zhongang Cai, Lei Yang, Huaping Liu, and Guosheng Lin. Gaussianeditor: Swift and controllable 3d editing with gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21476–21485, 2024. 1, 3

[6] Chao Ding and Ligang Liu. A survey of sketch based modeling systems. *Frontiers of Computer Science*, 10:985–999, 2016. 2

[7] Lin Gao, Jie Yang, Bo-Tao Zhang, Jia-Mu Sun, Yu-Jie Yuan, Hongbo Fu, and Yu-Kun Lai. Mesh-based gaussian splatting for real-time large-scale deformation. *arXiv preprint arXiv:2402.04796*, 2024. 3, 6, 7

[8] Antoine Guédon and Vincent Lepetit. Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5354–5363, 2024. 3, 6

[9] Benoit Guillard, Edoardo Remelli, Pierre Yvernay, and Pascal Fua. Sketch2mesh: Reconstructing and editing 3d shapes from sketches. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13023–13032, 2021. 2

[10] Yuan-Chen Guo. Instant neural surface reconstruction, 2022. https://github.com/bennyguo/instant-nsr-pl. 7

[11] Ayaan Haque, Matthew Tancik, Alexei A Efros, Aleksander Holynski, and Angjoo Kanazawa. Instruct-nerf2nerf: Editing 3d scenes with instructions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19740–19750, 2023. 3

[12] Chi Hong, Jiyue Huang, Robert Birke, Dick Epema, Stefanie Roos, and Lydia Y Chen. Sfddm: Single-fold distillation for diffusion models. *arXiv preprint arXiv:2405.14961*, 2024. 6

[13] Zehuan Huang, Hao Wen, Junting Dong, Yaohui Wang, Yangguang Li, Xinyuan Chen, Yan-Pei Cao, Ding Liang, Yu Qiao, Bo Dai, et al. Epidiff: Enhancing multi-view synthesis via localized epipolar-constrained diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9784–9794, 2024. 3

[14] Takeo Igarashi, Satoshi Matsuoka, and Hidehiko Tanaka. Teddy: a sketching interface for 3d freeform design. In *ACM SIGGRAPH 2006 Courses*, pages 11–es. 2006. 2

[15] Alec Jacobson, Daniele Panozzo, C Schüller, Olga Diamanti, Qingnan Zhou, N Pietroni, et al. libigl: A simple c++ geometry processing library. *Google Scholar*, 2013. 6

[16] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, 2006. 3

[17] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023. 1, 3

[18] Youngihn Kho and Michael Garland. Sketching mesh deformations. In *Proceedings of the 2005 symposium on Interactive 3D graphics and games*, pages 147–154, 2005. 2

[19] Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

[20] Vladislav Kraevoy, Alla Sheffer, and Michiel van de Panne. Modeling from contour drawings. In *Proceedings of the 6th Eurographics Symposium on Sketch-Based Interfaces and Modeling*, page 37–44, New York, NY, USA, 2009. Association for Computing Machinery. 2

[21] Huan Ling, Seung Wook Kim, Antonio Torralba, Sanja Fidler, and Karsten Kreis. Align your gaussians: Text-to-4d with dynamic 3d gaussians and composed diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8576–8588, 2024. 1, 3

[22] Feng-Lin Liu, Hongbo Fu, Yu-Kun Lai, and Lin Gao. Sketchdream: Sketch-based text-to-3d generation and editing. *ACM Transactions on Graphics (TOG)*, 43(4):1–13, 2024. 2, 3

[23] Ruoshi Liu, Rundi Wu, Basile Van Hoorick, Pavel Tokmakov, Sergey Zakharov, and Carl Vondrick. Zero-1-to-3: Zero-shot one image to 3d object. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9298–9309, 2023. 3, 5, 6

[24] Aryan Mikaeili, Or Perel, Mehdi Safaee, Daniel Cohen-Or, and Ali Mahdavi-Amiri. Sked: Sketch-guided text-based 3d editing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14607–14619, 2023. 2, 3

[25] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European Conference on Computer Vision*, pages 405–421. Springer, 2020. 1, 3

[26] Andrew Nealen, Olga Sorkine, Marc Alexa, and Daniel Cohen-Or. A sketch-based interface for detail-preserving mesh editing. In *ACM SIGGRAPH 2005 Papers*, pages 1142–1147. 2005. 2

[27] Andrew Nealen, Takeo Igarashi, Olga Sorkine, and Marc Alexa. Fibermesh: designing freeform surfaces with 3d curves. In *ACM SIGGRAPH 2007 papers*, pages 41–es. 2007. 2

[28] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988*, 2022. 2, 3, 5

[29] Jiawei Ren, Liang Pan, Jiaxiang Tang, Chi Zhang, Ang Cao, Gang Zeng, and Ziwei Liu. Dreamgaussian4d: Generative 4d gaussian splatting. *arXiv preprint arXiv:2312.17142*, 2023. 1, 3, 5, 7

[30] Yichun Shi, Peng Wang, Jianglong Ye, Mai Long, Kejie Li, and Xiao Yang. Mvdream: Multi-view diffusion for 3d generation. *arXiv preprint arXiv:2308.16512*, 2023. 3

[31] Hyeonseop Song, Seokhun Choi, Hoseok Do, Chul Lee, and Taehyeong Kim. Blending-nerf: Text-driven localized editing in neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14383–14393, 2023. 3

[32] Olga Sorkine and Marc Alexa. As-rigid-as-possible surface modeling. In *Symposium on Geometry processing*, pages 109–116. Citeseer, 2007. 3

[33] Nikita Starodubcev, Dmitry Baranchuk, Artem Fedorov, and Artem Babenko. Your student is better than expected: Adaptive teacher-student collaboration for text-conditional diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9275–9285, 2024. 6

[34] Jianyi Wang, Kelvin CK Chan, and Chen Change Loy. Exploring clip for assessing the look and feel of images. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 2555–2563, 2023. 7

[35] Junjie Wang, Jiemin Fang, Xiaopeng Zhang, Lingxi Xie, and Qi Tian. Gaussianeditor: Editing 3d gaussians delicately with text instructions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20902–20911, 2024. 1, 3

[36] Haoning Wu, Zicheng Zhang, Weixia Zhang, Chaofeng Chen, Liang Liao, Chunyi Li, Yixuan Gao, Annan Wang, Erli Zhang, Wenxiu Sun, et al. Q-align: Teaching lmms for visual scoring via discrete text-defined levels. *arXiv preprint arXiv:2312.17090*, 2023. 7

[37] Jing Wu, Jia-Wang Bian, Xinghui Li, Guangrun Wang, Ian Reid, Philip Torr, and Victor Adrian Prisacariu. Gaussctrl: multi-view consistent text-driven 3d gaussian splatting editing. *arXiv preprint arXiv:2403.08733*, 2024. 1, 3

[38] Tianhao Xie, Eugene Belilovsky, Sudhir Mudur, and Tiberiu Popa. Dragd3d: Vertex-based editing for realistic mesh deformations using 2d diffusion priors. *arXiv preprint arXiv:2310.04561*, 2023. 3

[39] Tianyi Xie, Zeshun Zong, Yuxing Qiu, Xuan Li, Yutao Feng, Yin Yang, and Chenfanfu Jiang. Physgaussian: Physics-integrated 3d gaussians for generative dynamics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4389–4398, 2024. 1, 3

[40] Qinghao Ye, Haiyang Xu, Jiabo Ye, Ming Yan, Anwen Hu, Haowei Liu, Qi Qian, Ji Zhang, and Fei Huang. mplug-owl2: Revolutionizing multi-modal large language model with modality collaboration. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, pages 13040–13051, 2024. 7

[41] Seungwoo Yoo, Kunho Kim, Vladimir G Kim, and Minhyuk Sung. 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*, pages 4315–4324, 2024. 3

[42] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3836–3847, 2023. 2, 6

[43] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5745–5753, 2019. 4

[44] Jingyu Zhuang, Chen Wang, Liang Lin, Lingjie Liu, and Guanbin Li. Dreameditor: Text-driven 3d scene editing with neural fields. In *SIGGRAPH Asia 2023 Conference Papers*, pages 1–10, 2023. 3

[45] Johannes Zimmermann, Andrew Nealen, and Marc Alexa. Sketching contours. *Computers & Graphics*, 32(5):486–499, 2008. 2