MeshUp: Multi-Target Mesh Deformation via Blended Score Distillation

Hyunwoo Kim
University of Chicago

Itai Lang
University of Chicago

Noam Aigerman
University of Montreal

Thibault Groueix
Adobe Research

Vladimir G. Kim
Adobe Research

Rana Hanocka
University of Chicago

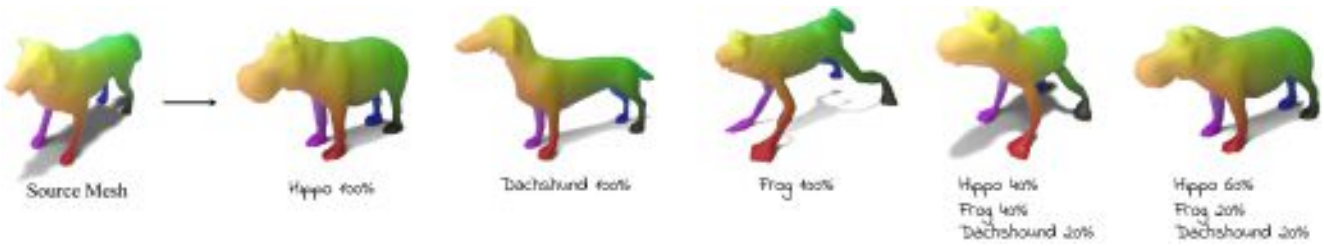

Figure 1. MeshUp is capable of deforming a source mesh into various concepts and into their weighted blends. The target objectives can be text prompts, images, or even mesh. Users can also input a set of *control vertices* to explicitly define where on the mesh the particular concepts should be expressed (Figure 8). The colors on the mesh visualize the point-wise correspondence between the source and the deformed mesh.

Abstract

We propose MeshUp, a technique that deforms a 3D mesh towards multiple target concepts, and intuitively controls the region where each concept is expressed. Conveniently, the concepts can be defined as either text queries, e.g., "a dog" and "a turtle," or inspirational images, and the local regions can be selected as any number of vertices on the mesh. We can effectively control the influence of the concepts and mix them together using a novel score distillation approach, referred to as the Blended Score Distillation (BSD). BSD operates on each attention layer of the denoising U-Net of a diffusion model as it extracts and injects the per-objective activations into a unified denoising pipeline from which the deformation gradients are calculated. To localize the expression of these activations, we create a probabilistic Region of Interest (ROI) map on the surface of the mesh, and turn it into 3D-consistent masks that we use to control the expression of these activations. We demonstrate the effectiveness of BSD empirically and show that it can deform various meshes towards multiple objectives. Our project page is at this URL

1. Introduction

Deforming mesh is a central task in geometry processing [11, 14, 40, 52, 55, 56]. In particular, it maintains valuable predefined attributes, such as artist-generated tessellation, UV map, textures, and motion functions. Deforming a mesh, however, still remains a task that requires significant expertise, making it difficult for non-experts to creatively manipulate 3D models without knowing their low-level attributes. Addressing this challenge requires an intuitive, high-level control over 3D shapes in a way that can induce any non-expert users' creative workflows. In this work, we explore the use of diffusion to enable a user-friendly deformation-based 3D content generation.

In addition to the ease of use, creative workflows in generative tasks are also inspired from their ability to synthesize novel imagery–namely, by combining a range of diverse concepts [5]. Some cognitive theories even suggest that the ability to synthesize novel combinations of known concepts and exploring these conceptual ideas is essential to human creativity [43]. While most methods that achieve 3D content generation optimize an implicit representation defined over 3D space [8, 34, 44], these representations are often inappropriate for mesh-specific tasks and cannot reuse any of the attributes defined over an artist-generated mesh. On the other hand, deformation-based approaches such as [18] lack the tools to enable a high-level, creative workflow for users to create novel conceptual imagery (e.g., "a creature with a bear's head and a frog's legs", or mixing across multiple targets) and achieve precise control over their expressions.

Motivated by this observation, we propose MeshUp, a novel approach that deforms a source mesh towards multiple target concepts defined using a variety of inputs (texts, images, and even meshes), and localizes the region where these concepts are manifested. Given as input various types of user-defined "concepts," their respective weights, and optionally a set of vertex points on the mesh, our method deforms a mesh to appropriately conform to a localized, weighted mixture of these concepts.

In order to create a mixture of various concepts, we blend the activation maps by running the denoising process for each target and injecting the corresponding maps into a unified denoising U-Net, a method we call Blended Score Distillation (BSD). We then estimate the gradients from the diffusion using Score Distillation Sampling (SDS), a method that enables the inference step of a diffusion model to be performed in a stochastic manner [44, 60], and optimize the mesh deformation parameters, which we represent as Jacobians [1]. For fine-grained control over user-specified local regions, our framework additionally takes as input a set of selected vertices, each for a corresponding concept. Then for these concepts, we create a probability map over the mesh surface by extracting the self-attention maps from a diffusion process run on a batch of multi-viewpoint renderings, and reversely mapping them back to the surface. We then rasterize this probability map to create an attention mask that we use to control the region of deformation within our BSD pipeline (see Figure 6).

We leverage this novel pipeline to build a comprehensive creative modeling tool for concept mixing. The key features of our tool are (1) the support for mesh deformation towards multiple targets, (2) the capability to control both the strength and the region of their expression, (3) the ability to use either text, images, or other meshes as inputs.

2. Related Work

Image Editing Using Diffusion. Following the success of text-to-image generative Diffusion Models [24, 25, 33, 42, 46, 48, 54], many diffusion-based image editing models [4, 6, 10, 12, 21, 22, 31, 41, 65, 70] have been developed. These methods allow introducing custom concepts [17, 47], or enable fine-grained control of which regions and aspects of the image change [6, 12, 21] by weighting, modifying, and transferring the attention weights and activation of the diffusion networks.

Text-to-3D. These pretrained 2D diffusion generative techniques have also been used to enable 3D generation. This is usually accomplished by optimizing a 3D representation so that its rendering matches the desired text prompt [2, 8, 9, 13, 32, 34, 35, 38, 44, 49–51, 57, 61, 62, 64, 69, 71]. These methods often rely on implicit fields as a 3D representation (e.g., NeRFs [39]), which limits their editability, and often requires additional mesh conversion to support standard graphics pipelines. While some techniques allow editing these implicit fields [3, 30, 49, 67], it is harder to provide local surface control, preserve correspondences (or use them to define continuous interpolations), with these models. A mesh can be extracted as a post-process [44] using marching cubes [36] and even further fine-tuned to match the desired prompt [34, 63], but these meshes would not be consistent with one another, and automatic methods do not produce artist-quality tessellations or UV mappings, necessary for a production-ready asset. In this work, we instead use deformation of a single reference shape guided by multiple concepts (e.g,. textual prompts), which enables retaining necessary characteristics of the artist-created asset and enables to create a continuous semantic space interpolating between the concepts.

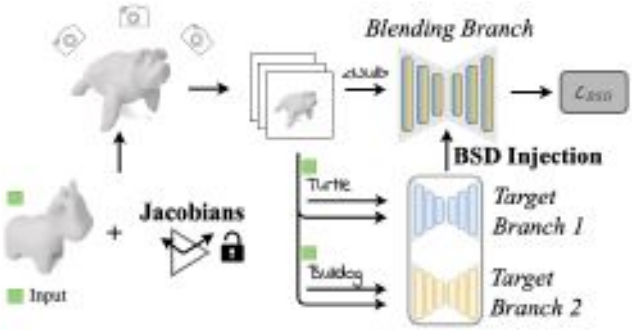

Figure 2. **Overview of Concept Blending.** MeshUp takes as input a 3D mesh and several target objectives, such as the text "Sea turtle" and "Bulldog." We deform the source mesh by optimizing the per-triangle Jacobians of the mesh. At each iteration, we render the mesh and apply the same random noise for each target objective. Then, we pass the noised renderings and the text input through the U-Net of a pretrained text-to-image model and store the activations associated with each objective (the *Target Branch)*. In the *Blending Branch*, we feed the noised rendering of the mesh to the U-Net, but condition it on the null-text embedding. We blend and inject the activations stored at each target branch into the blending branch. The gradients from the blending branch are then back-propagated via Score Distillation Sampling (SDS). After running this process iteratively, the mesh is deformed into a blend of "Sea turtle" and "Bulldog."

Mesh Deformation. Traditional mesh deformation is typically based on optimization of correspondences between vertices, faces, or other predictors that derive from these properties. [55, 56] use energy minimizing functions to give users control over the deformation space, while [16, 27] use skinning-based methods that interpolates the coordinate space with respect to the user handles. [53] uses optimal transportation to approximate correspondence across shapes. ARAP [55] and Laplacian surface editing [56] use a variational formulation to regularize the deformation in a

way that preserves details and prevents drifting of the geometry. However, these methods do not contain any semantics in their deformation and do not perform concept mixing.

Deforming a template mesh to various concepts has been explored even before the advances in neural networks [68], but these techniques required user annotations for rigging meshes via handles and assigning semantic labels.

Several data-driven techniques have been used to learn deformations [1, 15, 19, 20, 37, 58, 66]. A recent class of works leverages text prompts as user inputs for driving a deformation towards an arbitrary textual prompt [18, 28, 38]. These methods use various deformation representations, and we opt to leverage Jacobians since they produce smooth and large-scale global deformations.

We also observe that the CLIP objective lacks a full understanding of object details and that Diffusion-based objective, such as SDS [44] provides better guidance. The main goal of this work is to extend these techniques to multi-target deformation, and provide tools to mix, edit, and explore the space of concept combinations.

3. Method

The primary goal of our method is, given N sets of texts or image inputs that define the target "concepts," and their associated weights, w, to deform a source mesh into a shape that represents an effective mixture of these concepts, and control the "strength" of their expression using the weights. To that end, our method runs multiple diffusion pipelines in parallel and mixes their activation matrices within a unified pipeline to yield a single gradient direction that respects the appropriate weighted mixture of the target concepts.

For a framework that deforms a mesh into a specific target, two major design choices should be considered: the objective function (loss) and the representation of the mesh (the parameters to be optimized). For the objective function, we choose the Score Distillation Sampling (SDS) approach [44, 61], a prevalent generative technique that allows the diffusion inference process to be performed in a stochastic manner and thus enables our deformation process to be performed with viewpoint-consistency. While a straightforward application of this objective to mesh deformation would be to directly optimize the vertex positions, this method often leads to sharp artifacts [28] or restricts deformations to only local adjustments [38]. On the other hand, Jacobian-based deformation has been proposed for smooth, continuous, and global deformations, but it has only been used with an L_2 supervision [1] and CLIP similarity loss [18]. Using the SDS objective to supervise the optimization of the Jacobians offers a robust deformation framework with a powerful diffusion-based objective.

In this section, we first overview how one might approach a single target deformation using a combination of Jacobian-based mesh deformation and SDS guidance. We then extend this concept to multi-target deformation via our novel Blended Score Distillation and explain how we achieve local control over the deformations.

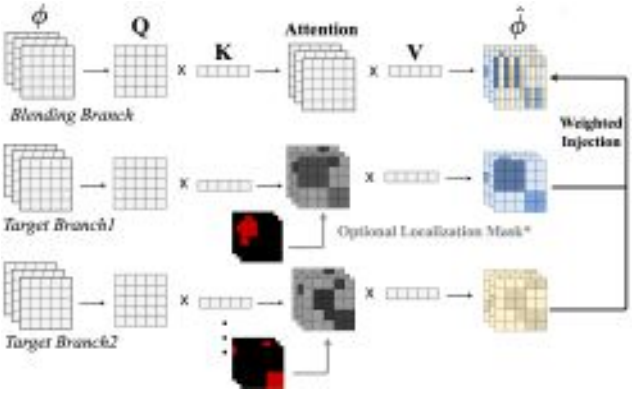

Figure 3. **Overview of Blended Score Distillation (BSD).** For each attention layer in the denoising U-Net, we inject the activation maps from *Target Branch1* and *Target Branch2* to the *Blending Branch* (*top*), blending the feature representations for each target. Optional Localization Mask* (*bottom*) indicates the additional mask that we optionally apply over the cross-attention maps for localization control. The mask identifies local regions described by the selected control vertices and different weights are assigned to each of these regions. For more details, please see Figure 6 and Localization Control part of Section 3.

Jacobian-Based Mesh Deformation. Our mesh deformation is represented by a per-face Jacobian matrix $J_i \in \mathbf{R}^{3\times3}$, where the deformation of a mesh (vertex positions) is computed by optimizing the following least squares problem (*i.e.*, Poisson Equation):

$$\gamma^* = min_\gamma \sum_i t_i ||\nabla_i(\gamma) - J_i||_2^2, \tag{1}$$

where γ^* is the deformation map embedding the mesh such that its Jacobians $\nabla_i(\gamma)$ are as close as possible to the target Jacobians $\{J_i\}$, the parameters we optimize, and $\{t_i\}$ are the triangle areas. Similar to previous works, we use a differentiable Poisson solver layer [1] to compute the deformation map, and a differentiable renderer [29] to connect this representation to image-based losses [18].

SDS Guidance for a Single-Target Mesh Deformation. To stochastically optimize any arbitrary parameters with respect to a pre-trained 2D diffusion model, [44] proposed the Score Distillation Sampling (SDS) process, where given a rendered image $\mathbf{z}$ and a text condition y, the objective is to minimize the L_2 loss between a sampled noise $\epsilon \sim \mathcal{N}(0, \mathbf{I})$ added to the image, and the noise ϵ_ω predicted by a denoising unet ω at some timestep t, sampled from a uniform distribution $t \sim U(0, 1)$:

$$\mathcal{L}_{\text{Diff}}(\omega, \mathbf{z}, y, \epsilon, t) = w(t) \left\| \epsilon_\omega \left(\mathbf{z}_t, y, t \right) - \epsilon \right\|_2^2, \tag{2}$$

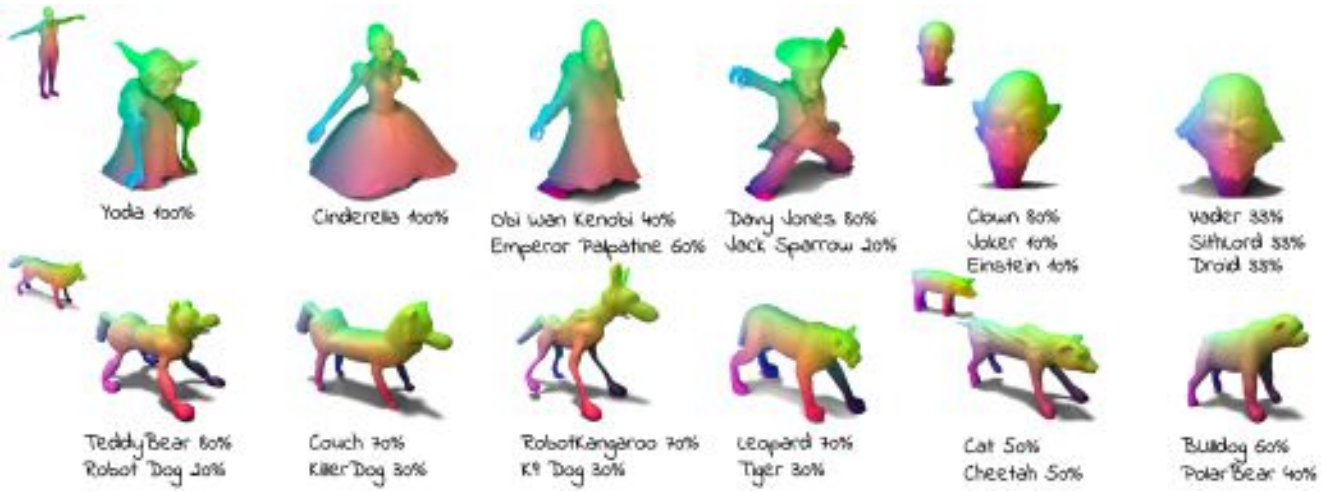

Figure 4. **Results Gallery.** We present a diverse set of 1-way, 2-way, and 3-way blending results of MeshUp. MeshUp can operate on various kinds of source shapes like human body, face, or animals, and can deform them into a blend of multiple concepts.

Figure 5. **Interpolation Between Two Objectives.** We show that we can vary the ratio between two objectives (*e.g.* going from hippo 100% on the left to Hippo 70%-Dachshund 30%, Hippo 30%-Dachshund 70% and finally Dachshund 100% on the right), effectively interpolating between the shape of the two targets.

where $w(t)$ is a weighting term used in the pretrained diffusion model [44], and $\mathbf{z}_t$ is the rendered image noised at the timestep t. In practice, to compute the gradient of the optimizable parameters efficiently with respect to the loss $\mathcal{L}_{\text{Diff}}$, it has been shown that the gradients through the U-Net of the diffusion model can be omitted [44, 61]. Since we aim to minimize the loss $\mathcal{L}_{\text{Diff}}$ by optimizing each jacobian J_i, we can estimate the gradient of the loss with respect to each jacobian as follows:

$$\nabla_{J_i}\mathcal{L}_{\text{SDS}}(\omega,\mathbf{z},y,\epsilon,t)=w(t)\left(\epsilon_\omega(\mathbf{z}_t,y,t)-\epsilon\right)\frac{\partial\mathbf{z}_t(J_i)}{\partial J_i}, \tag{3}$$

Using this SDS gradient, one can deform a mesh to a single target prompt. A detailed derivation could be found in the supplementary materials.

Following [18], we also find it beneficial to regularize the deformation by adding a Jacobian regularization loss

$$\mathcal{L}_I=\alpha\sum_{i=1}\|J_i-I\|_2\,, \tag{4}$$

where α is a hyperparameter determining the regularization strength. This loss penalizes the Jacobians against the identity matrix (which represents the identity deformation) to effectively restrict the magnitude of the deformation. Next, we describe how we extend this framework to multiple targets.

Multi-target Guidance via BSD. Our multi-target architecture is composed of several parallel diffusion branches: one that takes a null text prompt as input (the blending branch), and others with a user-specified target input prompt(the target branches) (see Figure 2). These branches also take the same batch of mesh renderings as input images.

For clarity, let j denote the index for the j^{th} target-branch, each associated with a target "concept." The j^{th} branch would take as input its associated target text, y_j, and a weight w_j that controls the degree to which y_j should be expressed. The key observation is that the activation matrices (ϕ^j) we get at the end of each attention layer represent the"weighted feature space" of each concept, defined over the space of the patch of the input renderings. To blend the features over this space, we perform a weighted interpolation of the activations across the patch dimension, and inject them into the corresponding patch location in the blending branch. Formally, we inject the activation for a single concept as follows:

$$\phi^{blend}\leftarrow w_j\dot{\phi}^j+(1-w_j)\dot{\phi}^{blend} \tag{5}$$

where ϕ^{blend} and ϕ^j are the activation matrices in the blending branch and target branch j, respectively. To blend two

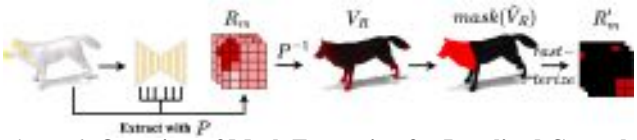

Figure 6. **Overview of Mask Extraction for Localized Control.** For Localized control, we first extract the self-attention maps that correspond to each *control vertex*. Using the inverse vertex-to-pixel map, P^{-1}, we then project the self-attention maps onto the mesh vertices and create a 3D attention map that describes the Region of Interest (ROI) per target concept (V_R). We normalize and threshold this ROI map to create a 3D mask ($mask(\hat{v}_R)$), and rasterize the map from the same viewpoints as the mesh renderings to generate R'_m, the localization masks to be applied to the cross attention layers of the BSD pipeline.

concepts from the i^{th} and j^{th} target branch, we would inject:

$$\phi^{blend} \leftarrow w_i\phi^i + w_j\phi^j + (1 - w_i - w_j)\phi^{blend} \quad (6)$$

The denoising U-Net from the blending branch utilizes the blended activations ϕ^{blend} to predict the noise added to the image, and the gradients are backpropagated using Equation (3) to update the Jacobians.

Localized Control. Notably, the BSD pipeline is designed in a way that can incorporate a more fine-grained control over the location where each concept is manifested. Specifically, we select a set of *control vertices* as additional inputs, and impose a novel localization constraint over our concept-mixing pipeline by leveraging the self-attention maps extracted from these vertices. We will first go over how we can achieve local control for a single target, then extend this concept to enable localized blending of multiple concepts.

We first begin by mapping the 3D vertex positions to their corresponding pixel locations in a set of rendered images by using a mapping function r that takes as input v, the vertex positions, and c, the camera parameters, to find a vertex-to-pixel mapping P:

$$P = r(v, c). \quad (7)$$

Next, we perform a denoising iteration on these renderings, and using the map P, we extract all the self-attention maps corresponding to the selected control vertices. We then average these maps across the attention layers to form a probabilistic region of interest (ROI) for each rendering, which we henceforth denote as R_m (the ROI map for the m^{th} rendering). We then use the inverse pixel map, P^{-1} to map R_m back to its corresponding vertex positions on the mesh surface:

$$V_R = \sum_m P^{-1}(R_m), \quad (8)$$

where V_R is the 3D probabilistic ROI defined over the mesh vertices. We iteratively update V_R during the BSD optimization process. We can then get a 3D-consistent 2D ROI map, $R_m{}'$, by normalizing V_R with $\hat{V_R} = \frac{V_R - \min(V_R)}{\max(V_R) - \min(V_R)}$, thresholding it at $th = 0.8$ to create a binary mask, $mask_{\hat{V_R}}$.

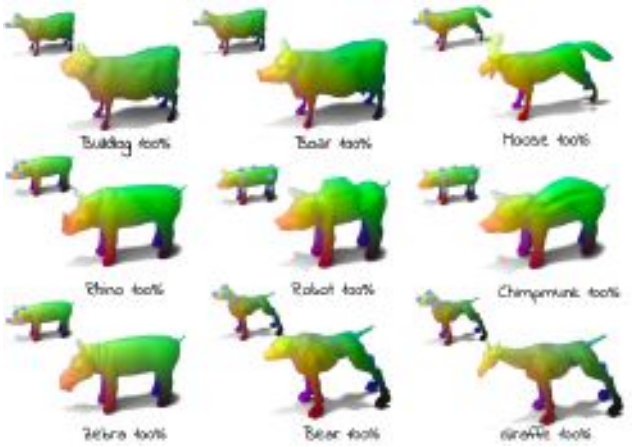

Figure 7. **Local Deformation for a Single Target** We show local deformation results for single text targets. We visualize the source mesh over the top-left corner of each result, and the selected *control vertices* as blue dots. Note how the deformation is constrained to the selected region.

For single-target deformation, we first deform the entire source mesh by regularly updating the jacobians, and at the end of all iterations, we use this binary mask to manually assign any jacobians that falls out of this mask region to the identity matrix:

$$mask_{\hat{V_R}}(J_i) = I. \quad (9)$$

By solving the poison equation 1 after such assignment, we effectively get a mesh that smoothly deforms to the target only within the region specified by the 3D consistent mask. As we visualize in Figure 7, our method's significant capability to deform the specified region while preserving its smooth connectivity to the preserved region offers our work to be used as a geometry-editing tool, where given a pre-defined source mesh, the users can select and partially edit specific regions of the mesh using text prompts.

Localized Control for Multiple Concept Blending. To "blend" guidance from a variety of these localized objectives, we first rasterize each of the 3D-mask $mask_{\hat{V_R}}(J_i)$, back to the 2D rendering space,

$$R_m{}' = Rast(mask_{\hat{V_R}}, v, c). \quad (10)$$

We then use $R_m{}'$ to mask-out the cross-attention maps, eliminating any association between the target and the unwanted regions of the mesh. Using BSD to mix activation from these masked attention maps yield a guidance score that respects both the weighted blend of multiple targets, as well as their associated local regions, as noticeable in Figure 8

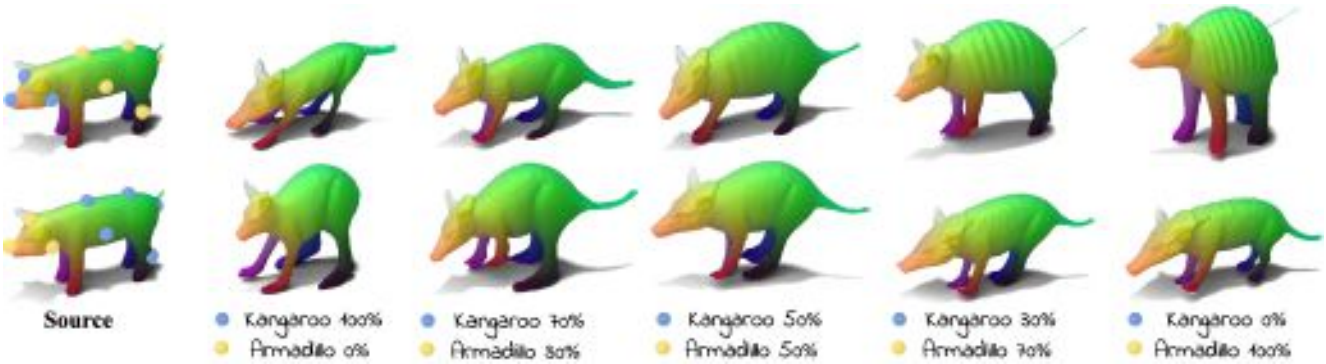

Figure 8. **Multiple Local Deformation Control.** We show deformation results for different selection of control points. Both columns show deformation results combined with various BSD weights of 1.0, 0.7, 0.5, and 0.3, 0.0, respectively, for the targets"Kangaroo" and "Armadillo." *(Top row)* the points around the head region (*blue dots*) are assigned to the target "Kangaroo," and points around the body to "Armadillo" (*yellow dots*). *(Bottom row)* flips the assignment. The figure demonstrates how the deformation results vary according to the assignment of selected control points.

Since mask R'_m is rasterized from a unified 3D ROI map V_R, it is consistent across multiple-viewpoints, and thus for the various renderings. Additionally, because V_R is continuously accumulated as the sum of the attention probabilities projected from multiple R_ms, it is guaranteed that the influence from a single attention map is minimized, preventing any particular viewpoints from adding significant variations to the ROI map. We show an ablation of this method in the supplements.

The rasterized R'_m is then used as a binary mask in our usual BSD pipeline to be applied over the cross-attention maps of a desired concept, constraining the area over which the concept can be manifested. Additionally, since self-attention maps extracted from real, non-inverted renderings can be less informative, we optionally fine-tune and overfit LoRA weights to precisely predict the noise from a large batch of multi-viewpoint renderings using the objective from [47]. We supply further details about this, as well as the localization method in the supplements.

Image Targets with Textual Inversion. Text prompts might often be insufficient to describe the desired target and images could be more descriptive in some settings. We leverage textual inversion [17], which converts an image target into a prompt encoding, and use the encoded prompt in place of the target prompt y of the target branch in our BSD framework.

4. Experiments

In this section, we first show multi-target deformation results driven by text or image targets. Additionally, we demonstrate deformations with local control and mesh targets. Finally, we also describe how our method can be used as a regularization term that controls the strength of a deformation. We provide comparisons with various baselines,

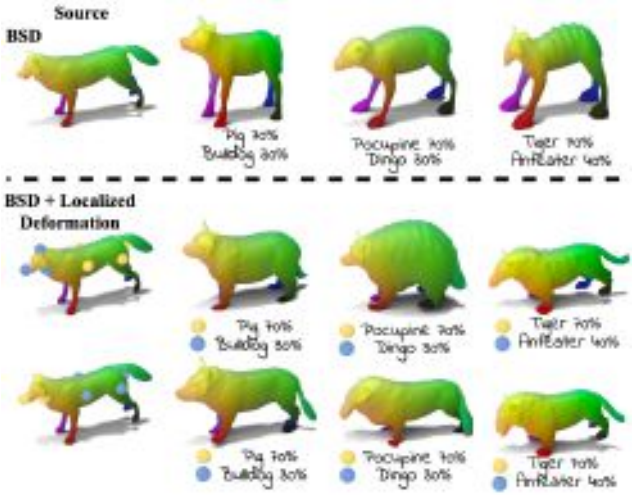

Figure 9. **Evaluation of BSD with and without Local Deformation.** We evaluate the deformation results with and without the Localized Deformation method. The *top row* shows results using just the naive BSD (our regular multi-target deformation), while the *middle* and *bottom row* shows the results using our localization method. We visualize the selected *control vertices* as blue and yellow dots on the mesh. Note how the results using our Localized Deformation method respect the assigned *control points*, in addition to the mixture of multiple targets.

show an experiment that uses our method to perform keypoint interpolation between concepts, and show a qualitative user study of our method in the supplementary material.

4.1. Concept Mixing Results

Multi-Target Results. We demonstrate various multi-target concept mixing results in Figures 1, 4, and 5. Our method successfully mixes diverse concepts (animals, faces, fantasy creatures, and vehicles) with various weights.

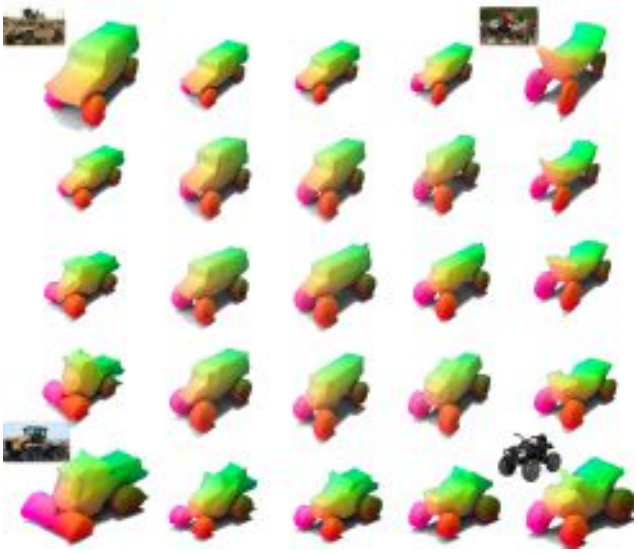

Figure 10. **Four-way Blending with Image Targets.** We use textual inversion [17] to condition the Text-to-image model with inspiration images represented by their inverted textual token. MeshUp supports as many targets as desired. We demonstrate a four-way blending with four target image concepts. The closer the shape is to the target image, the higher the corresponding blending weight of that target.

The figures illustrate how the same concepts can be mixed with different weights, enabling the user to control which features emerge more prominently. For example, in Figure 5, we can clearly see how with a high weight for the hippo shape, the fat body and the rounded face is prominent. On the other hand, dachshund's long body and facial features are dominant for examples with higher weights on dachshund.

Localization Control Results. In Figure 7 and Figure 8, we provide examples of localization control, where users can indicate (by selecting the control vertices, visualized in blue and yellow dots) which part of the model should be affected by each target. Note how each of the target features emerges in the user-specified region. This method offers a high level of control over how both mixed/unmixed concepts manifest in the deformed mesh. We also demonstrate how the local deformation is affected by changes in the assignment of weight (w) in Figure 8. We observe that if we give a different emphasis to different parts of the source mesh via the selection of control vertice, BSD conditions the emphasis accordingly on various scales, creating a more versatile space for user control.

Image Targets and the Concept Space. We show the ability of our system to take image concepts as inputs in Figure 10. We find this feature to be especially useful for some concepts that have significant shape variations (e.g., "trucks"), and for those that are difficult to engineer a precise prompt for. In this figure, we also illustrate how one can generate a continuous blending space spanning as many as four concepts by sampling different relative weights for each one of them.

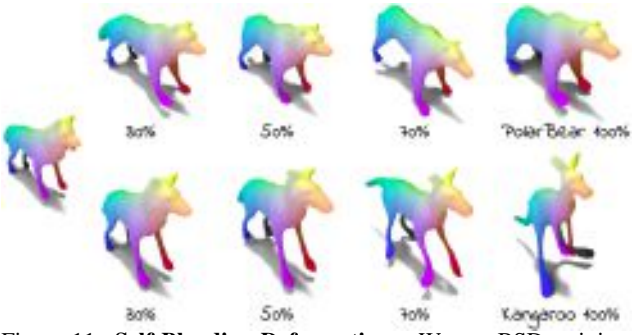

Figure 11. **Self-Blending Deformations**. We use BSD to inject varied strengths of activations, each from a single target to the blending branch, effectively controlling for the degrees to which the target is expressed in the resulting deformation.

Figure 12. **Interpolating Mesh Using Self-Blending Deformation.** We show how the Self-Blending capability of MeshUp can be used to interpolate the shapes of two meshes, the **Source** and the **Target**, by using dreambooth to learn the shape of the **Target**, and deforming the **Source** using various weights. Note how the muscular features of the **Target** mesh gradually emerge as we increase the blending weight from 30% to 70%.

Figure 13. **Texture Transfer.** We show how the texture map initially defined over the source mesh gets transferred without distortion to meshes deformed using our method.

Regularizing Mesh Deformation via Self-Blending. In Figure 11 we demonstrate that our BSD pipeline can take a single target objective, and be used to control the strength of a single-target deformation by using various weights, w_j. We use a modified classifier free guidance to achieve this (we provide its details and ablation in the supplements).

Using Mesh as Targets. In Figure 12, we further expand our model's self-blending capability to interpolate the shapes of two meshes, by gradually deforming the source

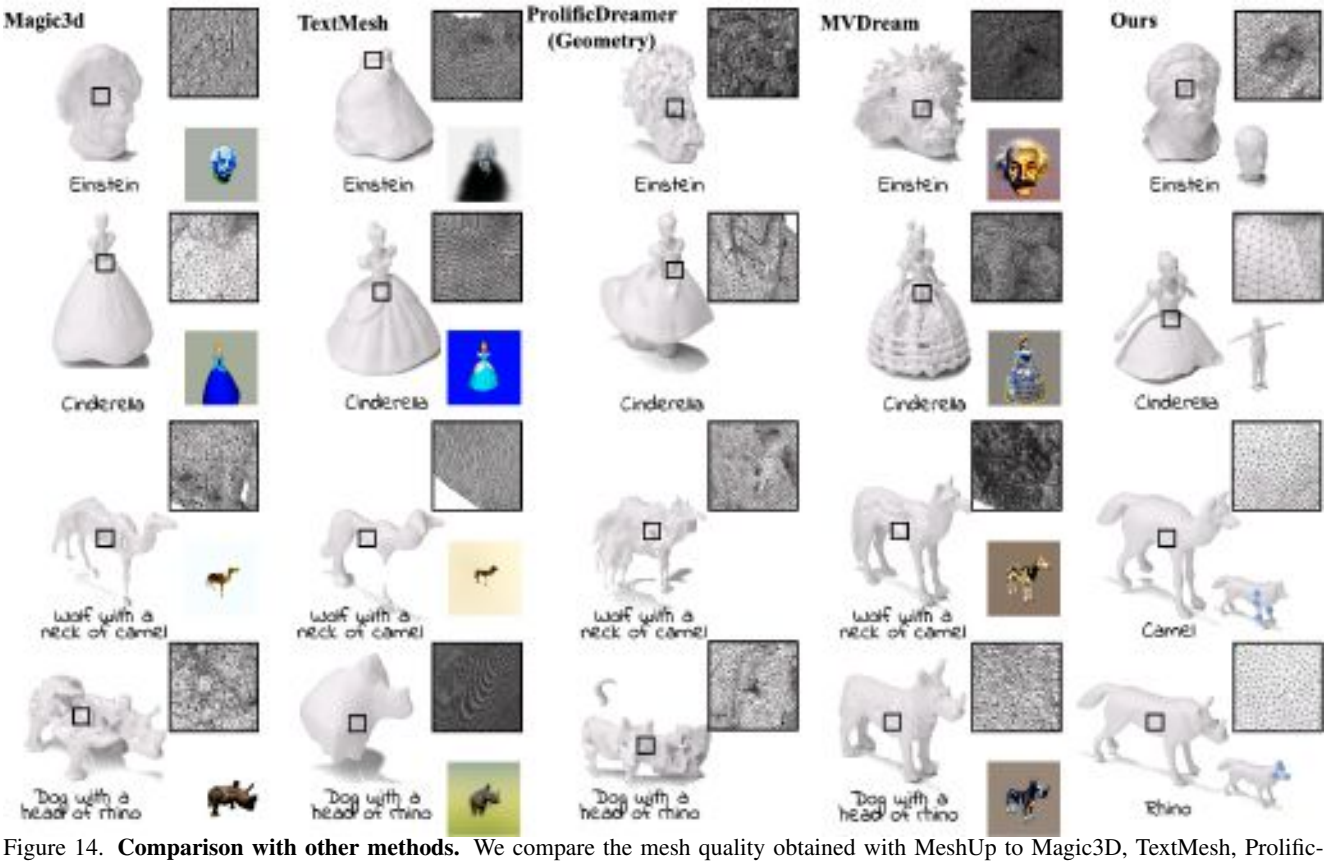

Figure 14. **Comparison with other methods.** We compare the mesh quality obtained with MeshUp to Magic3D, TextMesh, ProlificDreamer, and MVDream. For Magic3D, TextMesh, and MVDream, we visualize the textured, implicit shape representation on the bottom right of each figure (for ProlificDreamer, we only show the result after the geometry refinement stage). For our first two results, we deform the source mesh (visualized on the bottom right of each result) into the specified targets. For the last two results, we use our localized control method to confine the deformation to the region specified by the control vertices (visualized as blue dots over the source mesh).

mesh into a **Target** mesh. To achieve this, we utilize the multi-viewpoint renderings of the target, and batch 48 renderings per-iteration to fine-tune the UNet of the diffusion model using the objective from DreamBooth [47]. To avoid memory overload, we fine-tune the LoRA weights [26] instead of the whole model. Using the fine-tuned weights with the associated token as the objective, we deform the **Source** using various weights, w_j. Please refer to [47] for details of the training procedure.

Texture Transfer. We demonstrate the utility of deforming from a source shape using our method, as opposed to generating new 3D shapes from scratch. In Figure 13 we show how the texture map defined over the source seamlessly transfers over to other meshes deformed using our method. We can extend this property to transfer other attributes such as motion functions, and we show this example in the supplementary video, which can be found in our project page.

Comparison with Other Methods. Finally, we compare the quality of our mesh outputs to those extracted from Magic3D, TextMesh, ProlificDreamer (geometry refinement stage), MVDream. Not only does our method yield a geometry of much better detail and quality, but the tessellations (visualized on the right side of each figure) are also superior, a crucial advantage for any mesh-based graphic applications. We also show in the last two *bottom rows* that our localized control method significantly outperforms other methods that use text description to depict the localized deformation results we can achieve using MeshUp. More details of the comparison, including the specific models we used for these experiments, can be found in the supplements.

5. Conclusion and Limitations

In this paper, we propose a versatile framework for mesh deformation that supports creative workflows, enabling deformation via text or image-based concepts, mixing these concepts using various weights, and localizing their expressions.

The deformation of MeshUp is focused on preserving the topology of the initial mesh, and treating it as a shape-prior which prescribes the aesthetic of the deformed results. Thus, our technique would not be suitable for inducing topological changes (such as deforming a sphere to an object with topological holes). We leave the task of topology-modifying mesh modifications to future work. Although we limit our focus to deformation in

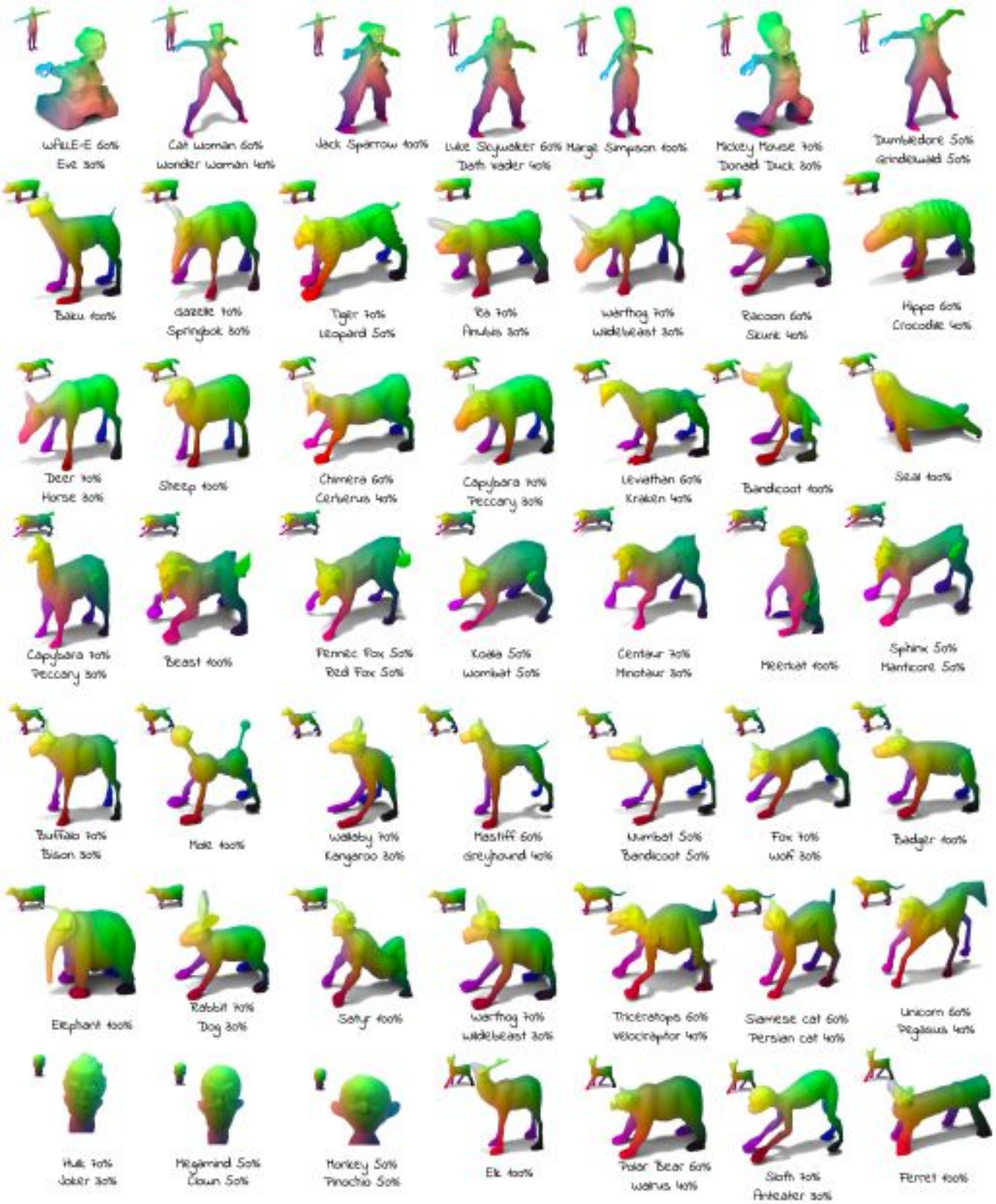

Figure 15. **Gallery.** We present a diverse set of 2-way MeshUp deformation results.

this paper, another potential application of our method would be to leverage our technique for generating other mesh parameters, such as textures, materials, and normals.

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. *SIGGRAPH*, 2022. 2, 3

[2] Mohammadreza Armandpour, Ali Sadeghian, Huangjie Zheng, Amir Sadeghian, and Mingyuan Zhou. Re-imagine the negative prompt algorithm: Transform 2d diffusion into 3d, alleviate janus problem and beyond, 2023. 2

[3] Chong Bao, Yinda Zhang, Bangbang Yang, Tianxing Fan, Zesong Yang, Hujun Bao, Guofeng Zhang, and Zhaopeng Cui. Sine: Semantic-driven image-based nerf editing with prior-guided editing field, 2023. 2

[4] Manuel Brack, Felix Friedrich, Dominik Hintersdorf, Lukas Struppek, Patrick Schramowski, and Kristian Kersting. Sega: Instructing text-to-image models using semantic guidance, 2023. 2

[5] Lindsay Brainard. The curious case of uncurious creation. *Inquiry*, 0(0):1–31, 2023. 1

[6] Tim Brooks, Aleksander Holynski, and Alexei A. Efros. Instructpix2pix: Learning to follow image editing instructions, 2023. 2

[7] Chun-Fu Chen, Quanfu Fan, and Rameswar Panda. Crossvit: Cross-attention multi-scale vision transformer for image classification, 2021. 14

[8] Zilong Chen, Feng Wang, and Huaping Liu. Text-to-3d using gaussian splatting, 2023. 1, 2

[9] Xinhua Cheng, Tianyu Yang, Jianan Wang, Yu Li, Lei Zhang, Jian Zhang, and Li Yuan. Progressive3d: Progressively local editing for text-to-3d content creation with complex semantic prompts, 2023. 2

[10] Jooyoung Choi, Yunjey Choi, Yunji Kim, Junho Kim, and Sungroh Yoon. Custom-edit: Text-guided image editing with customized diffusion models, 2023. 2

[11] Etienne Corman and Maks Ovsjanikov. Functional characterization of deformation fields. *ACM Transactions on Graphics (TOG)*, 38(1):1–19, 2019. 1

[12] Guillaume Couairon, Jakob Verbeek, Holger Schwenk, and Matthieu Cord. Diffedit: Diffusion-based semantic image editing with mask guidance, 2022. 2

[13] Dale Decatur, Itai Lang, Kfir Aberman, and Rana Hanocka. 3d paintbrush: Local stylization of 3d shapes with cascaded score distillation, 2023. 2

[14] Ana Dodik, Oded Stein, Vincent Sitzmann, and Justin Solomon. Variational barycentric coordinates. *ACM Transactions on Graphics (TOG)*, 42(6):1–16, 2023. 1

[15] Marvin Eisenberger, David Novotny, Gael Kerchenbaum, Patrick Labatut, Natalia Neverova, Daniel Cremers, and Andrea Vedaldi. Neuromorph: Unsupervised shape interpolation and correspondence in one go, 2021. 3

[16] Lawson Fulton, Vismay Modi, David Duvenaud, David I. W. Levin, and Alec Jacobson. Latent-space dynamics for reduced deformable simulation. *Computer Graphics Forum*, 2019. 2

[17] Rinon Gal, Yuval Alaluf, Yuval Atzmon, Or Patashnik, Amit H. Bermano, Gal Chechik, and Daniel Cohen-Or. An image is worth one word: Personalizing text-to-image generation using textual inversion, 2022. 2, 6, 7

[18] William Gao, Noam Aigerman, Groueix Thibault, Vladimir Kim, and Rana Hanocka. Textdeformer: Geometry manipulation using text guidance. In *ACM Transactions on Graphics (SIGGRAPH)*, 2023. 1, 3, 4, 18

[19] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan C. Russell, and Mathieu Aubry. Deep self-supervised cycle-consistent deformation for few-shot shape segmentation. *SGP*, 2019. 3

[20] Rana Hanocka, Noa Fish, Zhenhua Wang, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. Alignet: Partial-shape agnostic alignment via unsupervised learning. *ACM Trans. Graph.*, 2018. 3

[21] Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Prompt-to-prompt image editing with cross attention control, 2022. 2

[22] Amir Hertz, Kfir Aberman, and Daniel Cohen-Or. Delta denoising score, 2023. 2

[23] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance, 2022. 14

[24] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020. 2

[25] Susung Hong, Gyuseong Lee, Wooseok Jang, and Seungryong Kim. Improving sample quality of diffusion models using self-attention guidance, 2023. 2

[26] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021. 8, 14

[27] Alec Jacobson, Zhigang Deng, Ladislav Kavan, and JP Lewis. Skinning: Real-time shape deformation. In *ACM SIGGRAPH 2014 Courses*, 2014. 2

[28] Nasir Mohammad Khalid, Tianhao Xie, Eugene Belilovsky, and Tiberiu Popa. CLIP-mesh: Generating textured meshes from text using pretrained image-text models. In *SIGGRAPH Asia 2022 Conference Papers*. ACM, 2022. 3

[29] Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, and Timo Aila. Modular primitives for high-performance differentiable rendering, 2020. 3

[30] Verica Lazova, Vladimir Guzov, Kyle Olszewski, Sergey Tulyakov, and Gerard Pons-Moll. Control-nerf: Editable feature volumes for scene rendering and manipulation, 2022. 2

[31] Dongxu Li, Junnan Li, and Steven C. H. Hoi. Blip-diffusion: Pre-trained subject representation for controllable text-to-image generation and editing, 2023. 2

[32] Weiyu Li, Rui Chen, Xuelin Chen, and Ping Tan. Sweetdreamer: Aligning geometric priors in 2d diffusion for consistent text-to-3d, 2023. 2

[33] Yuheng Li, Haotian Liu, Qingyang Wu, Fangzhou Mu, Jianwei Yang, Jianfeng Gao, Chunyuan Li, and Yong Jae Lee.

Gligen: Open-set grounded text-to-image generation, 2023. 2

[34] Chen-Hsuan Lin, Jun Gao, Luming Tang, Towaki Takikawa, Xiaohui Zeng, Xun Huang, Karsten Kreis, Sanja Fidler, Ming-Yu Liu, and Tsung-Yi Lin. Magic3d: High-resolution text-to-3d content creation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 1, 2

[35] 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, 2023. 2

[36] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *SIGGRAPH*, pages 163–169. ACM, 1987. 2

[37] Arman Maesumi, Paul Guerrero, Noam Aigerman, Vladimir G. Kim, Matthew Fisher, Siddhartha Chaudhuri, and Daniel Ritchie. Explorable mesh deformation subspaces from unstructured 3d generative models. *SIGGRAPH Asia (Conference track)*, 2023. 3

[38] Oscar Michel, Roi Bar-On, Richard Liu, Sagie Benaim, and Rana Hanocka. Text2mesh: Text-driven neural stylization for meshes. *arXiv preprint arXiv:2112.03221*, 2021. 2, 3

[39] 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 *ECCV*, 2020. 2

[40] Niloy J Mitra, Simon Flöry, Maks Ovsjanikov, Natasha Gelfand, Leonidas J Guibas, and Helmut Pottmann. Dynamic geometry registration. In *Symposium on geometry processing*, pages 173–182, 2007. 1

[41] Ron Mokady, Amir Hertz, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Null-text inversion for editing real images using guided diffusion models, 2022. 2

[42] Chong Mou, Xintao Wang, Liangbin Xie, Yanze Wu, Jian Zhang, Zhongang Qi, Ying Shan, and Xiaohu Qie. T2i-adapter: Learning adapters to dig out more controllable ability for text-to-image diffusion models, 2023. 2

[43] Elliot Samuel Paul and Scott Barry Kaufman, editors. *The Philosophy of Creativity*. Oxford University Press, New York, 2014. 1

[44] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv*, 2022. 1, 2, 3, 4, 13, 15

[45] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021. 18

[46] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2022. 2

[47] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation, 2023. 2, 6, 8, 14

[48] Patrick Schramowski, Manuel Brack, Björn Deiseroth, and Kristian Kersting. Safe latent diffusion: Mitigating inappropriate degeneration in diffusion models, 2023. 2

[49] Etai Sella, Gal Fiebelman, Peter Hedman, and Hadar Averbuch-Elor. Vox-e: Text-guided voxel editing of 3d objects, 2023. 2

[50] Junyoung Seo, Wooseok Jang, Min-Seop Kwak, Jaehoon Ko, Hyeonsu Kim, Junho Kim, Jin-Hwa Kim, Jiyoung Lee, and Seungryong Kim. Let 2d diffusion model know 3d-consistency for robust text-to-3d generation, 2023.

[51] Yichun Shi, Peng Wang, Jianglong Ye, Mai Long, Kejie Li, and Xiao Yang. Mvdream: Multi-view diffusion for 3d generation, 2024. 2

[52] Justin Solomon, Mirela Ben-Chen, Adrian Butscher, and Leonidas Guibas. As-killing-as-possible vector fields for planar deformation. In *Computer Graphics Forum*, pages 1543–1552. Wiley Online Library, 2011. 1

[53] Justin Solomon, Fernando de Goes, Gabriel Peyré, Marco Cuturi, Adrian Butscher, Andy Nguyen, Tao Du, and Leonidas Guibas. Convolutional wasserstein distances: efficient optimal transportation on geometric domains. *ACM Trans. Graph.*, 34(4), 2015. 2

[54] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models, 2022. 2

[55] Olga Sorkine and Marc Alexa. As-rigid-as-possible surface modeling. In *Symposium on Geometry processing*, pages 109–116. Citeseer, 2007. 1, 2

[56] Olga Sorkine, Daniel Cohen-Or, Yaron Lipman, Marc Alexa, Christian Rössl, and H-P Seidel. Laplacian surface editing. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 175–184, 2004. 1, 2

[57] Junshu Tang, Tengfei Wang, Bo Zhang, Ting Zhang, Ran Yi, Lizhuang Ma, and Dong Chen. Make-it-3d: High-fidelity 3d creation from a single image with diffusion prior, 2023. 2

[58] Mikaela Angelina Uy, Vladimir G. Kim, Minhyuk Sung, Noam Aigerman, Siddhartha Chaudhuri, and Leonidas Guibas. Joint learning of 3d shape retrieval and deformation. *CVPR*, 2021. 3

[59] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2017. 14

[60] Haochen Wang, Xiaodan Du, Jiahao Li, Raymond A. Yeh, and Greg Shakhnarovich. Score jacobian chaining: Lifting pretrained 2d diffusion models for 3d generation, 2022. 2

[61] Haochen Wang, Xiaodan Du, Jiahao Li, Raymond A. Yeh, and Greg Shakhnarovich. Score jacobian chaining: Lifting pretrained 2d diffusion models for 3d generation. *arXiv preprint arXiv:2212.00774*, 2022. 2, 3, 4

[62] Zhengyi Wang, Cheng Lu, Yikai Wang, Fan Bao, Chongxuan Li, Hang Su, and Jun Zhu. Prolificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation, 2023. 2

[63] Zhengyi Wang, Cheng Lu, Yikai Wang, Fan Bao, Chongxuan Li, Hang Su, and Jun Zhu. Prolificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation. *arXiv preprint arXiv:2305.16213*, 2023. 2

[64] Menghua Wu, Hao Zhu, Linjia Huang, Yiyu Zhuang, Yuanxun Lu, and Xun Cao. High-fidelity 3d face generation from natural language descriptions, 2023. 2

[65] Binxin Yang, Shuyang Gu, Bo Zhang, Ting Zhang, Xuejin Chen, Xiaoyan Sun, Dong Chen, and Fang Wen. Paint by example: Exemplar-based image editing with diffusion models, 2022. 2

[66] Wang Yifan, Noam Aigerman, Vladimir G Kim, Siddhartha Chaudhuri, and Olga Sorkine-Hornung. Neural cages for detail-preserving 3d deformations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 75–83, 2020. 3

[67] Yu-Jie Yuan, Yang-Tian Sun, Yu-Kun Lai, Yuewen Ma, Rongfei Jia, and Lin Gao. Nerf-editing: Geometry editing of neural radiance fields, 2022. 2

[68] Mehmet Yumer, Siddhartha Chaudhuri, Jessica Hodgins, and Levent Kara. Semantic shape editing using deformation handles. *ACM Transactions on Graphics*, 34:86:1–86:12, 2015. 3

[69] Jingbo Zhang, Xiaoyu Li, Ziyu Wan, Can Wang, and Jing Liao. Text2nerf: Text-driven 3d scene generation with neural radiance fields, 2023. 2

[70] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models, 2023. 2

[71] Minda Zhao, Chaoyi Zhao, Xinyue Liang, Lincheng Li, Zeng Zhao, Zhipeng Hu, Changjie Fan, and Xin Yu. Efficientdreamer: High-fidelity and robust 3d creation via orthogonal-view diffusion prior, 2023. 2

Supplementary Material

Technical Details In this section, we will go over some of the technical details of our model.
First, we note that we use the IF model (XL size) by Deepfloyd for our denoiser. While the IF model consists of three components–a Text-to-Image Diffusion model (Unet) and two super-resolution models–we only use its Text-to-Image Denoiser. As opposed to the widely used Stable-Diffusion that operates in the latent embedding space, IF does so in the rgb space, and reducing both memory usage and training time, without compromising quality. In addition to the denoiser, the t5 text encoder on which IF is trained has a much more expressive embedding space than that of CLIP. Since we aim to employ Textual Inversion to learn a set of images through this embedding space, the t5 encoder comes to our great advantage.

We ran our results using a single A40 gpu, and ran 2400 iterations with a batch size (number of renderings per iteration) of 16. It takes approximately 1.5 hours to optimize for a single target. The time increases by a factor of approximately 1.5 for each target that we add for blending. However, we have observed that the deformation results get reasonably similar to the final results after 500-600 iterations, which takes about 20 minutes to run for a single target. Running the localized deformation method also increases the run-time by an approximate factor of 1.5, not including the optional LoRA-finetuning stage, which takes about an extra 10-15 minutes. The optimization time also depends on the number of vertices/faces of the source mesh, and we can therefore only provide a rough approximation of the run-time.

We also note that we use the nvdiffrast model to rasterize the mesh into 2D images from multiple viewpoints. Since our main objective is to optimize the geometry of the source mesh (and not its texture), we simply paint the mesh with a uniformly grey texture, and rasterize it to get the input renderings. We found that the grey texture works best for our purpose, but we leave the question of improving the geometry with a more sophisticated texturing scheme up for future work. We first rasterize the mesh in 512x512 resolution and downsample it to 64x64 using bilinear interpolation before feeding it to the diffusion model.

Specific Overview of SDS We will go over the SDS perspective of diffusion in detail, and how we apply this perspective for our purpose of deforming the Jacobians. We first limit our scope to 2D image-to-image generation, where, given an input image $\mathbf{z}$, the objective is to optimize some parameter ϕ in the 2D space. Given a text condition $\mathbf{t}$, we define the loss function to be the squared L2 norm between the noise predicted by the denoising model ϵ_ϕ and the sampled noise $\epsilon \sim \mathcal{N}(0, \mathbf{I})$.

$$\mathcal{L}_{\text{Diff}}(\phi, \mathbf{z}, y, \epsilon, t) = w(t) \left\| \epsilon_\phi \left(\mathbf{z_t}, y, t \right) - \epsilon \right\|_2^2, \tag{11}$$

Note t is the timestep randomly sampled from a uniform distribution, $t \sim \mathcal{U}(0, 1)$, ϵ is a random noise sampled from a standard normal distribution, and $\mathbf{z_t}$ is the input image noised for timestep t using the re-parameterization trick whereby we conveniently get the noise at timestep $\mathbf{t}$ using the equation, $\mathbf{z_t} = \sqrt{\alpha_t}\mathbf{z} + \sqrt{1-\alpha_t}\epsilon$. $w(t)$ is a weighting function for which we will not go into details. In simple terms, we aim to optimize some parameters so that the (frozen) model can precisely predict the noise sampled from timestep t. When this loss is minimized, the parameters are optimized to represent an object (or an image rendered from the object) that is as close as possible to that "guided" by the denoiser.

To optimize for some parameter ϕ, which, in most image generation applications, will be the image pixel value, we guide ϕ toward the highest probability region predicted by the denoiser.

The gradient of $\mathcal{L}_{\text{Diff}}$ with respect to ϕ, which we denote $\nabla_\theta \mathcal{L}_{\text{Diff}}$, can be derived by,

$$\nabla_\theta \mathcal{L}_{\text{Diff}} = \left(\epsilon_\phi \left(\mathbf{z_t}, y, t \right) - \epsilon \right) \frac{\partial \epsilon_\phi(\mathbf{z}, y, t)}{\partial \mathbf{z_t}} \frac{\partial \mathbf{z_t}}{\partial \theta}. \tag{12}$$

It is known from [44] that instead of having to backpropagate through the denoiser we can approximate an effective gradient for ϕ, denoted as $\nabla_\theta \mathcal{L}_{\text{SDS}}$, simply by omitting the gradient with respect to the denoiser, $\frac{\partial \epsilon_\phi(\mathbf{z}, y, t)}{\partial \mathbf{z_t}}$, giving us the equation,

$$\nabla_\theta \mathcal{L}_{\text{SDS}}(\mathbf{z}, y, \epsilon, t) = \left(\epsilon_\phi \left(\mathbf{z_t}, y, t \right) - \epsilon \right) \frac{\partial \mathbf{z_t}}{\partial \theta} \tag{13}$$

In addition to SDS, we also use classifier-free guidance with an extremely high weight of 100 to help with our 3D objective, following the observation from [44].

Using SDS to guide Mesh Deformation In order to optimize for the Jacobians $\mathbf{J_i}$ of mesh faces instead of some arbitrary parameter ϕ, we can simply multiply $\nabla_{\mathbf{J_i}} \mathcal{L}_{\text{SDS}}$ to any additional gradients involved in calculating the image renderings from the optimized per-face Jacobians. We can change our equation into,

$$\nabla_{\mathbf{J_i}} \mathcal{L}_{\text{SDS}} = \left(\epsilon_\phi \left(\mathbf{z_t}, y, t \right) - \epsilon \right) \frac{\partial \mathbf{G}(\mathbf{J_i})}{\partial \mathbf{J_i}}.$$

where we abuse notation and abstractly denote $\mathbf{G}(\mathbf{J_i})$ as the span of the entire computational pipeline, ranging from the optimized 3x3 Jacobians to the renderings of the deformed mesh (this includes the operations of projecting the 3x3 Jacobians to the 3x2 space, running the poison solve

to calculate the deformation map ϕ, deforming the mesh, rendering this deformed mesh onto the image space). In practice, we use Pytorch's autograd library to automatically handle the differentiation of $\mathbf{G}(\mathbf{J_i})$.

Localized Deformation We will now detail the implementation of our localized deformation experiment.

First, we note that each attention layer of the Unet denoiser takes as input the text encoding and the hidden states, or activations, given by the previous attention layer. We then project the hidden states each onto the embedding space of Key, Query, and Value. These projected matrices are then concatenated with the similarly projected text encoding vector, giving us the Key, Query, and Value matrices. [7, 59] Specifically, the attention map is defined as

$$M = \text{Softmax}\left(\frac{QK^T}{\sqrt{d}}\right)$$

While the activation is

$$\phi = MV.$$

Note that the attention matrix M contains both the self- and cross-attention map, where the cross-attention maps the correlation between the text prompt embeddings and various "patches" (or regions) within the image and the self-attention map does so between the "patches" themselves. Using this observation, we utilize the self-attention maps to identify the probable regions that have high correspondence with the patches to which the *control vertices* are mapped. Once we extract the 3D-consistent ROI mask R'_m using our localized control method, we apply this mask to the cross-attention map of each target, thereby masking out the regions within the image where the target concepts (represented by the text prompt embeddings) can be expressed.

We will now delineate some of the implementation details of this localized deformation method. First, as briefly mentioned in the method section, we fine-tune the denoising Unet of our DeepFloyd IF model to make it precisely predict the sampled noise from the renderings of the input mesh. In other words, we train the model such that it can precisely reconstruct the renderings during the inference time. The objective of our fine-tuning is similar to that proposed in [47], but we use multi-viewpoint mesh renderings as inputs and do not use the Class Prior Preservation Loss. This fine-tuning has a similar effect to inverting the renderings of the mesh, encouraging the model to be capable of reconstructing the mesh renderings.

Moreover, since fine-tuning the entire U-net as implemented in the original Dreambooth paper is prohibitively expensive, we instead finetune the LoRA weights [26] with the same training objective as described above. We run 150 fine-tuning iterations on a batch of 16 images, and the entire operation takes approximately 15 mins.

Additionally, we also use the fine-tuned weights to stabilize the localized deformation method. Notably, when the weight w_j is particularly high (over 0.8) for a certain target, the activation $\phi^{blend} = w_i\phi^i + w_j\phi^j$ might become extremely unstable as w_i could be too small so that the activation ϕ^{blend} receives extremely low amount of signal, or weights, from the region masked-out from the j^{th} localization targets. In order to compensate for this instability, whenever the $(1 - max(w_j, w_i)) < 0.2$ (when one target has a dominently high weight), we create an inverse boolean mask of R'_m, R'_m and apply this mask to the cross-attention from the LoRA-finetuned text-token. We then weight this cross attention by $w_{lora} = 0.2$, and add it to the cross-attention mask of the j^th target (the target within $w_j > 0.8$). This operation ensures that all the regions of the mesh receive at least a minimal amount of attention from all regions required to keep the activation ϕ^j stable throughout the deformation process.

The entire localization process takes about 2 hours to complete (for a 2-way blending of concepts) but could be reduced by a factor of approximately 1.3 by extracting the self-attention maps R_m from the original diffusion processes conditioned on the target concepts without fine-tuning the LoRA weights. The quality of localization, however, could be compromised.

Modified Classifier Free Guidance for Self-Blending In this section, we will delineate the modified classifier free guidance (CFG) method that we use for our self-blending experiments. In the self-blending experiment, we reduce our pipeline to take a single concept as input and control the weights w_j to regularize the expression of the concept. We therefore reduce the branches of the multi-target pipeline into one deformation and a target branch and run two parallel pipelines instead of many.

One caveat to this method, however, is that when the target branches are reduced to one, there is only one text/image target that can replace the target encodings of the deformation branch, which always leaves the deformation branch partially conditioned on the null text prompt "", introducing some undesired bias for $\nabla_{theta} L_{SDS}$.

To counter this problem, we slightly modify our equation for classifier-free guidance as follows. The original Classifier Free Guidance, a method introduced by [23] to alleviate the bias and complexity entangled with the classifier inputs of diffusion models, is formulated as follows,

$$\epsilon_\phi(\mathbf{z_t}, y, t) = \hat{\epsilon}_{text} + \alpha(\hat{\epsilon}_{text} - \hat{\epsilon}_{null}),$$

where $\hat{\epsilon}_{text}$ and $\hat{\epsilon}_{null}$ are the predicted noise conditioned on the target text prompt and null text prompt, respectively. α is the Guidance Scale, a parameter that defines the strength of the classifier-free guidance:

$$\epsilon_\phi(\mathbf{z_t}, y, t) = \epsilon + \alpha(\hat{\epsilon}_{text} - \hat{\epsilon}_{null}),$$

Figure 16. **Keyframe interpolation.** We create a continuous combinatorial space of blends by running our method for a discrete number of keyframes and interpolating their vertices to obtain the intermediate shapes in between (the ones without text below). Our correspondence-preserving deformation enables a smooth transition between the keyframes.

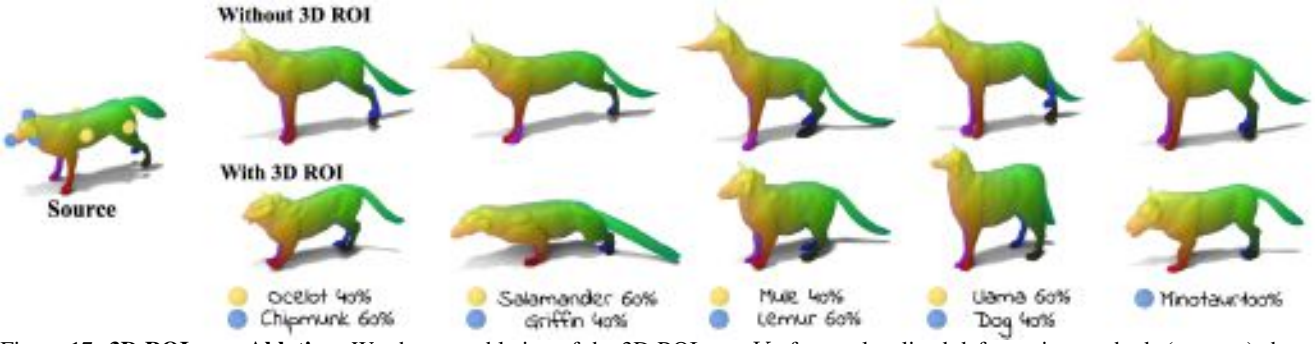

Figure 17. **3D ROI map Ablation.** We show an ablation of the 3D ROI map V_R for our localized deformation method. (*top row*) shows the results of using just the 2D ROI maps, R_m, extracted from the self-attention maps of each viewpoint, as masks for the cross-attention map. (*bottom row*) uses R'_m, the 3D-consistent masks extracted from the 3D ROI map V_R.

where we replace $\hat{\epsilon}_{text}$ with ϵ, the sampled noise.

Such modification to the CFG score allows our model to achieve a more unbiased control of the deformation strength, as opposed to the original CFG. More intuitively, we want there to be no deformation when the weight is set to 0. The modified CFG ensures that the gradient $\nabla_\theta L_{\text{Diff}} = 0$ when we set the deformation strength to 0, since if we do not inject any attention from the target branch, the predicted noise becomes:

$$\begin{aligned}\epsilon_\phi(\mathbf{z_t}, y, t) &= \epsilon + \alpha(\hat{\epsilon}_{text} - \hat{\epsilon}_{null}) \\ &= \epsilon + \alpha(\hat{\epsilon}_{null} - \hat{\epsilon}_{null}) \\ &= \epsilon,\end{aligned}$$

and consequently,

$$\nabla_\theta \mathcal{L}_{\text{SDS}}(\mathbf{z}, y, \epsilon, t) = (\epsilon_\phi(\mathbf{z_t}, y, t) - \epsilon)\frac{\partial \mathbf{z_t}}{\partial \theta} = 0 \qquad (14)$$

Injecting attention to this modified CFG score ensures a more stable control of the deformation strength by getting rid of the bias that ϵ_{null} introduces on the score. The modified CFG score thereby provides guidance that aligns more closely with the intuition of interpolating between the "identity deformation" and "deformation conditioned on the target prompt."

We additionally note that we can replace $\hat{\epsilon}_{text}$ with ϵ, with minimal sacrifice in quality because we use a significantly higher guidance scale α of 100, following the findings of [44]. Such a high guidance scale makes $\epsilon_{\hat{text}}$ relatively insignificant compared to the Classifier Free Guidance term, ensuring a minimal sacrifice in quality.

A. Additional Experiments

Continuous Concept Space. Finally, users might want to explore a continuous space of generated concepts. While we could run our pipeline multiple times with different relative weights between targets, this could easily become prohibitively expensive if users want a smooth continuity (e.g., generating morphing animations) and have a large combination of targets. To address this challenge, we observe that our mesh-based representation provides a dense correspondence map between the source and the deformed shapes. Thus, once the user generates a few sample key frame shapes, they can be smoothly interpolated continuously, offering a powerful space for concept exploration.

In Figure 16, we show that any keyframe concepts generated in our system can be further interpolated easily, thanks to dense correspondences maintained by our mesh deformation technique. Although our method deforms the source mesh into a blend of various shapes on discreet weight scales w, we can use our method to define key-point meshes and interpolate these key points to get an even more fine-grained blend of targets.

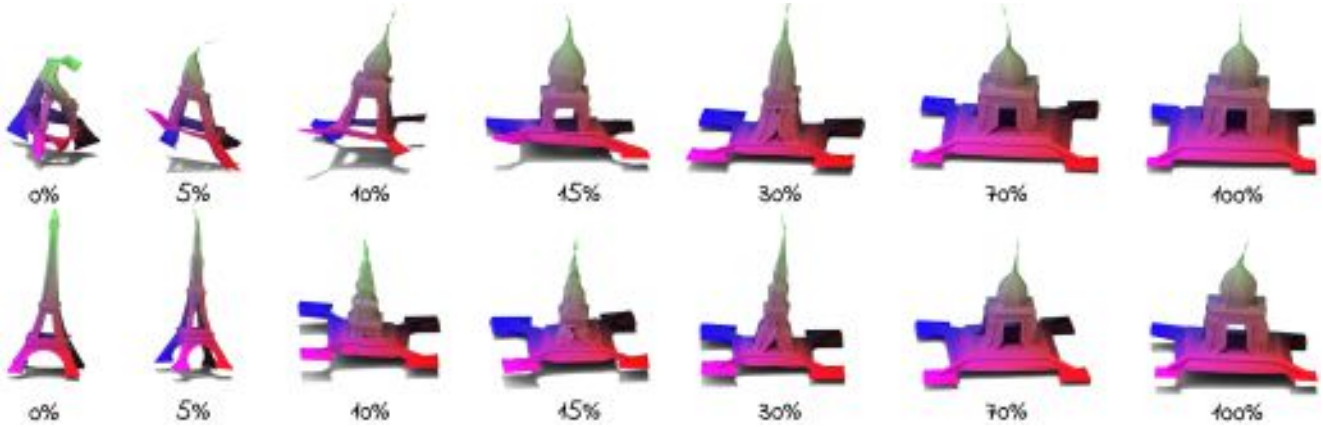

Figure 18. **Modifying the classifier-free guidance ablation.** We show an ablation for the modified classifier-free guidance version of our method for single-target self-blending deformation. Results are shown for various blending scales, ranging from 0% to 100%. The top row is the regular classifier-free guidance and the bottom one is with our modified version. The regular classifier-free guidance creates artifacts and does not reflect well the mixing percentage. In contrast, our self-blending scheme yields a smooth transition from the source shape to the target deformation objective.

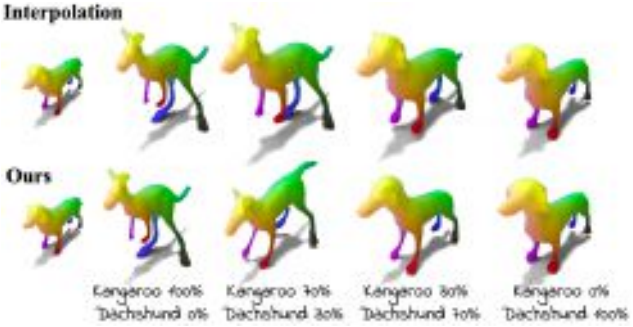

Figure 19. **Comparison to naive interpolation.** We compare our two-way blending results (*bottom*) with a naive interpolation of vertices coordinates (*top*). The control weights w are 1.0, 0.7, 0.5, 0.3, 0.1 for each column.

3D ROI Ablation. We show an ablation of the 3D ROI map V_R for our localized deformation method in Figure 17. Due to the viewpoint consistency of our 3D ROI map, our method can generate smooth, meaningful mixing results that respect the specified local regions for each target. This is in stark contrast to the results we get without using the 3D ROI map (directly using the self-attention maps extracted from each rendering to mask cross-attention maps), where we observe sharp artifacts as well as significant loss of details for the specified targets.

Modified Classifier Free Guidance Ablation. In Figure 18, we show an ablation of our modified Classifier Free Guidance for self-blending experiments. We notice that the effect of this modification is particularly crucial on smaller weights (typically from 0% to 15% of target weight) since for the self-blending application, a low target weight implies that there is more bias introduced from the null text prompt. Notice that the results conditioned on smaller weights without our modified classifier free guidance are severely irregular and biased, while using the modified classifier free guidance stabilizes this bias, regulating the deformation in a much stable, smooth manner.

B. Comparisons

Since our method is the first one to address the concept mixing problem in mesh deformation, we create several baselines using existing methods and compare our results to what these baselines can achieve.

Shape Interpolation Baseline. We also consider a simple interpolation baseline. We use our single-branch SDS with Jacobian optimization to create two deformations of the source shape with respect to two different textual targets. Then, we directly interpolate the vertex positions between these two targets, as presented in Figure 19. The shape interpolation baseline does not enable new features to emerge for the in-between shapes, thus, leading to non-descriptive, overly-smooth interpolation results. In contrast, our BSD method clearly prioritizes the emergence of notable features of each target (notice how the legs of the Kangaroo emerge first while the face of the Dachshund is clearly prioritized, respective of the weights given to each target).

We provide a more extensive qualitative comparison for the single-target deformation between TextDeformer and our method in the supplements. Although the main focus of the paper is concept mixing, the figure shows that our method substantially improves the single-target deforma-

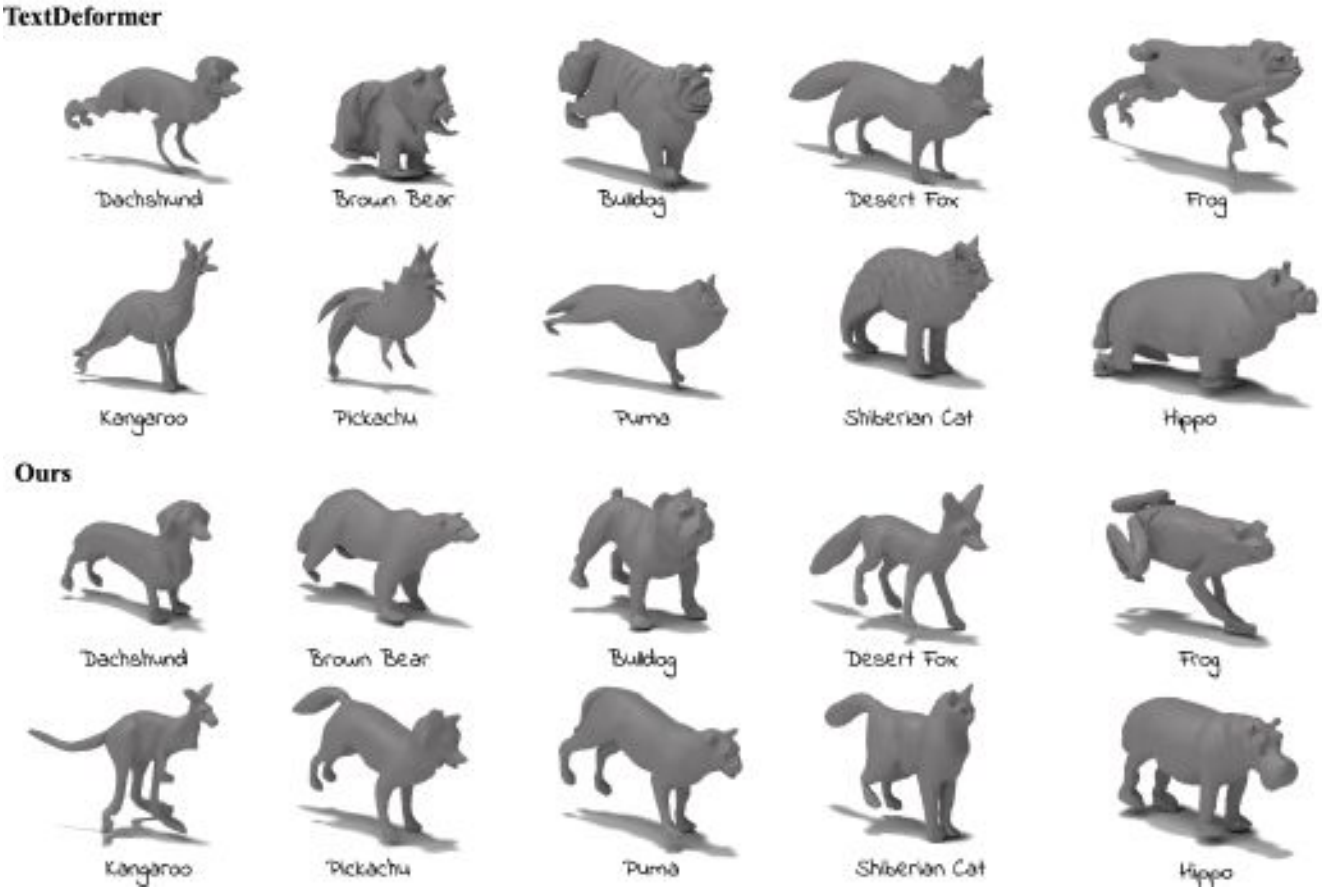

Figure 20. **Comparison of our method and TextDeformer.** MeshUp archives higher detail deformation results with less artifacts.

tion quality over TextDeformer.

Perceptual User Study. We present two perceptual user studies to evaluate the overall quality of our results. First, we asked 21 users to compare 5 single-target deformation results by TextDeformer and our method and choose the one that better depicts the input text targets, "bear," "bulldog," "dachshund," "kangaroo," and "frog" (see Table 1). We observe that the users clearly prefer the quality of our method over TextDeformer.

We also asked the same users to evaluate the accuracy of the various BSD weights applied to the blending of two targets, "Siberian Cat" and "Hippo," by guessing the correct weights from which the 4 different results were created. Specifically, they were asked to choose from the mixing weight pairs (0.2, 0.8), (0.4, 0.6), (0.6, 0,4), and (0.8, 0.2). Table 2 summarizes the results

We highlight the significant accuracy of the users' guesses, suggesting that the BSD weights control the concept blending in an intuitive and plausible manner. The set of renderings that we used for both evaluations can be found in the supplementary section.

Target	TextDeformer	MeshUp (ours)
Bear	0.048	**0.952**
Frog	0.095	**0.905**
Bulldog	0.0	**1.0**
Dachshund	0.0	**1.0**
Kangaroo	0.0	**1.0**

Table 1. **Perceptual user study for quality comparison.** We asked 21 users to compare the quality of our method and TextDeformer. The preference rate for each method is the portion of users who chose the result from one method over the other. The users have a strong preference for our method over the compared one.

Quantitative Comparison We compare our method to TextDeformer by running single-target deformation for both methods. We use a source dog mesh and warp it to the following 10 different prompts: "bear", "bulldog",

Targets: Siberian Cat % / Hippo %	User's Selection Accuracy
Siberian Cat 80% / Hippo 20%	**85.7%**
Siberian Cat 60% / Hippo 40%	**66.7%**
Siberian Cat 40% / Hippo 60%	**85.7%**
Siberian Cat 20% / Hippo 80%	**90.5%**

Table 2. **Perceptual user study for the blending weight of our BSD.** We asked 21 users to guess the weights applied to each BSD deformation. The percentages for each section denote the number of users who guessed the weights correctly. In the majority of the blending settings, the users selected the right mixing percentages. This finding suggests that the blending weights properly reflected the level of influence of each target objecting on the resulting deformed shape.

	CLIP R-Precision ↑		
Method	CLIP B/14@336px	CLIP B/16	CLIP B/32
TextDeformer	0.7	0.8	0.8
Ours	**0.8**	0.8	**0.9**

Table 3. **Quantitative evaluation.** We compare MeshUp to TextDeformer [18] and report CLIP R-Precision [45]. Note that TextDeformer has an advantage to our method since it is directly supervised with CLIP loss.

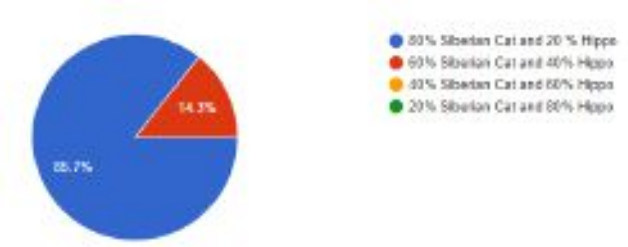

Figure 21. User response for 80% Siberian cat and 20% Hippo result.

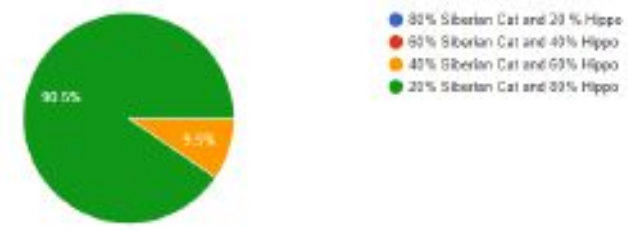

Figure 22. User response for 20% Siberian cat and 80% Hippo result.

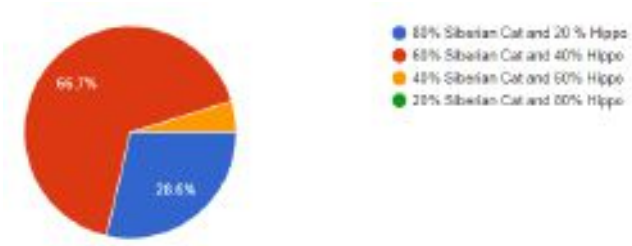

Figure 23. User response for 60% Siberian cat and 40% Hippo result.

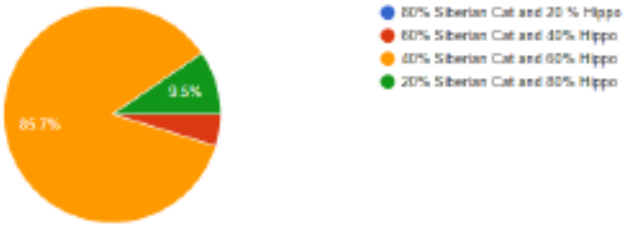

Figure 24. User response for 40% Siberian cat and 60% Hippo result.

"dachshund", "desertfox", "frog", "hippo", "kangaroo", "pig", "puma", "siberian cat." We evaluate our result using CLIP R-Precision score and show results in Table 3. Our method outperforms TextDeformer on most metrics. The results that we used for both this evaluation and user study can be found in figure 20.