

Supplementary Material

The following sections will provide more details of our method, present additional experiments, and evaluate our method using quantitative/qualitative metrics. Section A will delineate the technical and mathematical details of our method, including model configurations, runtime, Score Distillation Sampling (SDS), localized deformation, and the modified classifier free guidance score used in our self-blending experiments. Section B will present the experiments, and Section C will provide evaluations. We also provide a comprehensive collection of various deformation results in Figure 24.

A. Technical Details

First, we briefly justify our choice of DeepFloyd-IF XL model as our denoiser. For our purposes, we noticed that IF performs significantly better than the widely used latent Diffusion models, most likely as it operates directly in the rgb space. 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 leave more extensive comparison across various diffusion models to future work.

For runtime, we ran all of 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 converges reasonably close to the final results after 600-800 iterations, which takes about 25-30 minutes to run for a single target. Running the localized deformation method also increases the run-time approximately by an factor of 1.5, not including the optional LoRA-finetuning stage, which takes an extra 10-15 minutes to converge. The optimization time also depends heavily on the number of faces of the source mesh, and we can therefore only provide a rough approximation of the run-time. Most of the meshes that we used in this paper had 10,000-30,000 faces.

For rendering, we use nvdiffrast to rasterize the mesh 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 (0.5, 0.5, 0.5) texture, and rasterize it to get the renderings. We found the grey texture works reasonably well for our purpose, but we leave the investigation of more sophisticated texturing schemes up for future work. We also rasterize the mesh in 512x512 resolution and downsample it to 64x64 using bilinear interpolation before inputting 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 θ 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 $\epsilon \sim \mathcal{N}(0, \mathbf{I})$:

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

where t is the randomly sampled timestep 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 according to timestep t using the re-parameterization trick, $\mathbf{z}_t = \sqrt{\alpha_t}\mathbf{z} + \sqrt{1 - \alpha_t}\epsilon$. $w(t)$ 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 (θ) are optimized to represent an object that is as close as possible to what is predicted by the denoiser to be of highest probability. A deeper analysis of diffusion models and Score Distillation Sampling can be found in [24, 45, 61]

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

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

It is known from [45] that instead of having to backpropagate through the denoiser we can approximate an effective gradient for θ , 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,

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

In addition to SDS, we also use classifier-free guidance with an extremely high classifier-free guidance weight of 100 to help with our 3D objective [45].

Using SDS to guide Mesh Deformation In order to optimize for the Jacobians J_i of mesh faces instead of some arbitrary parameter θ , we can simply replace θ in (13) with J_i ,

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

We denote \mathbf{z}_t to be dependent on θ since we denote it to encompass the span of the entire computational pipeline, 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 poisson 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 $\mathbf{z}_t(\mathbf{J}_i)$.

Localized Deformation We will now detail the implementation of our localized deformation experiment. This explanation is specific to the DeepFloyd IF model that we use

First, we note that the UNet denoiser consists of multiple attention layers, and each of these layer takes as input the text encoding and the hidden states, output by the previous layer. We then project the hidden states each as Key, Query, and Value matrices using learned mlp layers. These matrices are then concatenated with matrices similarly projected from the text embeddings, giving us the matrices, Q , K , V . [7, 60] Specifically, the attention map is defined as

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

While the activation is

$$\phi = MV.$$

The attention matrix M consists of both the self- and cross-attention map. The cross-attention map encapsulates the correlation between the text embeddings and the various “patches” of the image, while the self-attention map finds the correlation across these patches. Using this observation, we first utilize the self-attention maps to identify the regions on the image that have high correspondence with the *control vertices*, and convert this map into a 2D Region of Interest (ROI) mask, R_m . We then extract the 3D-consistent ROI mask R'_m using our rasterization function, project it back onto 2D, and use it to mask out the cross-attention map of each target. Effectively, we are distilling the regions within the image where target concepts should be expressed.

We will now delineate some of the implementation details of the localized deformation method. First, as briefly mentioned in the method section, we aim to replicate the effect of inverting the renderings by finetune the denoising UNet of our DeepFloyd IF model by the objective proposed in [48]. While dreambooth finetunes the denoiser so that it can reproduce the input image, we do so for multi-viewpoint mesh renderings, encouraging the model to be capable of reconstructing the mesh renderings given a pre-defined token. Moreover, since fine-tuning the entire UNet 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. While this finetuning process betters the quality of the localization map, we also observe that the map is reasonable

enough even without finetuning. Users who desire faster generation time over better localization could skip this process with minimal compromise in quality.

Additionally, we use the finetuned weights to stabilize the localized deformation method. Notably, when a particular weight w_i is extremely low (over 0.2) for a certain target, the blended activation $\phi^{blend} = w_i\phi^i + w_j\phi^j$ (where i and j each represent different target concepts) might become extremely unstable as the localized region corresponding to the i^{th} concept could only receive minimal activation signals. In order to compensate for this instability, whenever the $(1 - \max(w_j, w_i)) < 0.2$ (when one target has extremely small weight), we create an inverse boolean mask, $\sim R'_m$ and apply this mask to the cross attention map of the finetuned UNet. We then weight this masked cross attention by $w_{lora} = 0.8$, and simply add it to the cross-attention mask of the i^{th} target. This operation ensures that all the regions of the mesh receive at least a minimal amount of attention required to keep the activation ϕ^j stable throughout the deformation process.

With the localization process, the entire deformation process takes about 2 hours to complete (for a 2-way blending of concepts).

Modified Classifier Free Guidance for Self-Blending In this section, we will discuss how we modified the Classifier Free Guidance (CFG) for our self-blending experiments (Figure 11). In the self-blending experiment, we utilize our blending pipeline to control the expression of a single concept. We thus reduce the branches to two (one target branch and one blending branch), and extract the activation as $\phi^{blend} = w\phi + (1 - w)\phi^{null}$, where ϕ^{null} is the activation from the blending branch (activation from the null text prompt, “”).

One caveat to this method, however, is that ϕ^{null} could introduce undesired bias for $\nabla_{\theta} L_{SDS}$, especially when its corresponding weight, $1 - w$, is high.

To tackle this problem, we slightly modify our equation for CFG. The original CFG, a method introduced by [23], 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 controls the strength of CFG. We modify our equation into

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

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

Such modification allows our model to achieve a more stable, unbiased control of the deformation strength, as opposed to the original CFG, as shown in Figure 17. Intuitively, we want the result to identity (no deformation) when

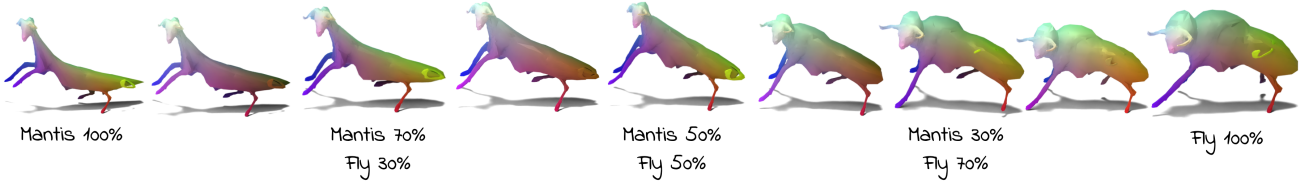


Figure 15. **Keyframe interpolation.** We create a continuous combinatorial space of blends by running our method for a discrete number of keyframes and interpolating their vertices to obtain the intermediate shapes in between (the ones without text below). Our correspondence-preserving deformation enables a smooth transition between the keyframes.

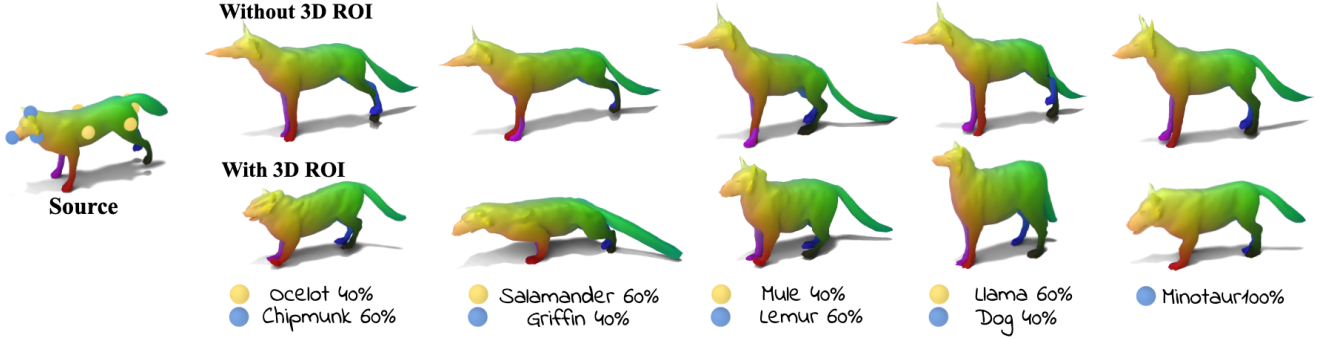


Figure 16. **3D ROI map Ablation.** We show an ablation of the 3D ROI map V_R for our localized deformation method. (*top row*) shows the results of using just the 2D ROI maps, R_m , extracted from the self-attention maps of each viewpoint, as masks for the cross-attention map. (*bottom row*) uses R'_m , the 3D-consistent masks extracted from the 3D ROI map V_R .

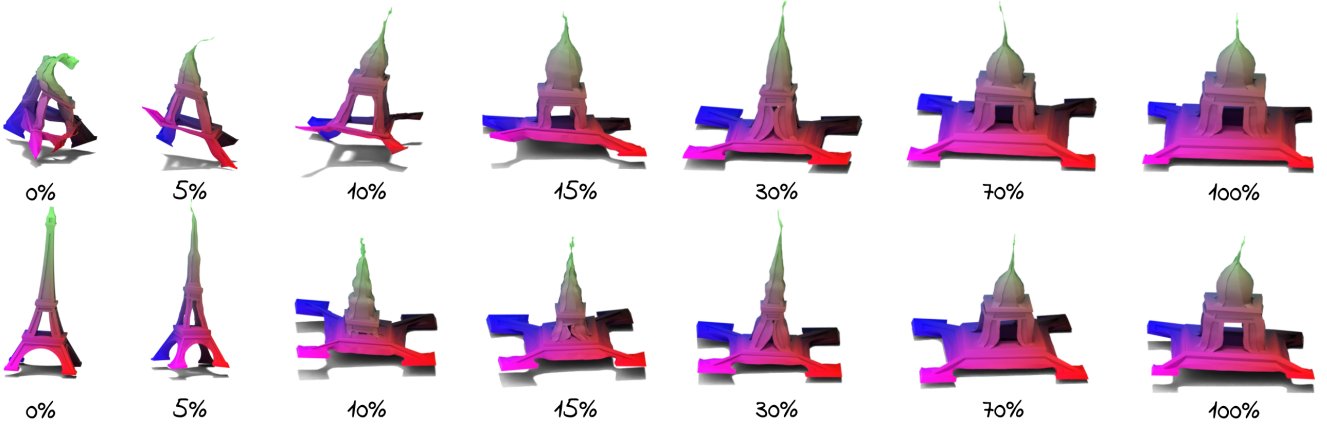


Figure 17. **Modifying the classifier-free guidance ablation.** We show an ablation for the modified classifier-free guidance version of our method for single-target self-blending deformation. Results are shown for various blending scales, ranging from 0% to 100%. The top row is the regular classifier-free 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.

the weight is set to 0, which is precisely what the modified equation achieves. As follows, the modified CFG ensures that the gradient $\nabla_{\theta} L_{\text{Diff}} = 0$ when the deformation

strength is 0, where our equation gives,

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

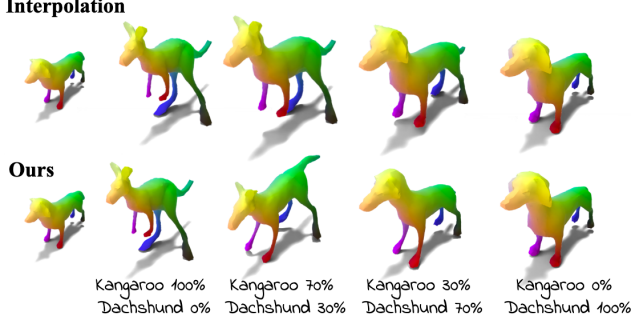


Figure 18. **Comparison to naive interpolation.** We compare our two-way blending results (*bottom*) with a naive interpolation of vertices coordinates (*top*). The control weights w are 1.0, 0.7, 0.5, 0.3, 0.1 for each column.

and consequently,

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

Injecting attention to this modified CFG score ensures a more stabilized control of the deformation strength by getting rid of the bias introduced by ϵ_{null} . 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 we can replace $\hat{\epsilon}_{\text{text}}$ with ϵ , with minimal compromise in quality because we use a significantly high guidance scale α of 100, following the findings of [45]. Such a high guidance scale makes ϵ_{text} relatively insignificant compared to the Classifier Free Guidance term, ensuring minimal change in quality.

B. Additional Experiments

Continuous Concept Space. Here, we explore a scenario when our 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). To address this challenge, we observe that our mesh-based representation provides a dense correspondence map between the source and the deformed shapes. Thus, the user could generate a few sample key frame shapes using our method and smoothly interpolated between them. We show this experiment in Figure 15

3D ROI Ablation. We show an ablation of the 3D ROI map V_R for our localized deformation method in Figure 16. 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. In contrast, the results we get without using the 3D ROI map (by directly using the 2D ROI map extracted from self-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 17, 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 more bias 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.

C. Comparisons

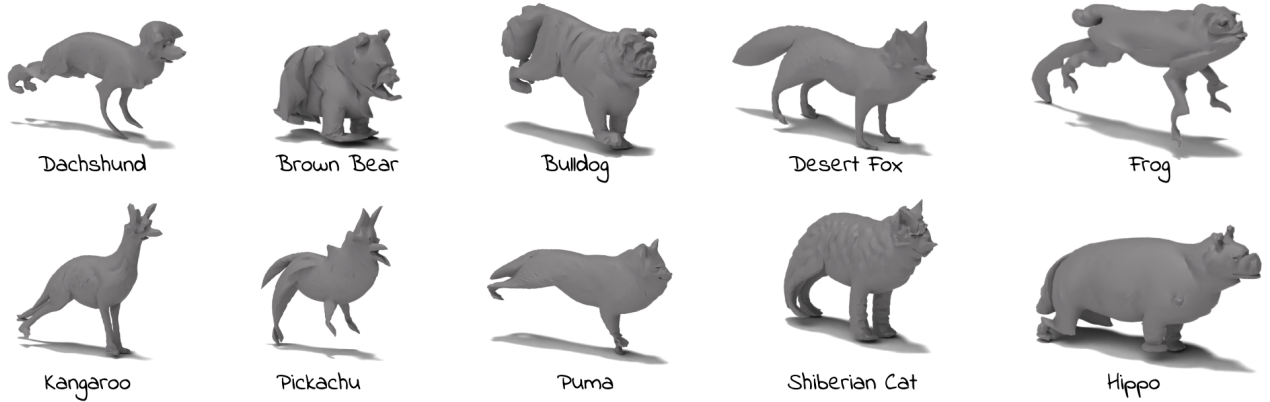
Since our method is the first to address the concept mixing in mesh deformation, we create simple baselines using existing methods and compare our results to what these baselines can achieve in both qualitative and quantitative metrics.

Shape Interpolation Baseline. We consider a simple interpolation baseline, where we generate deformations of two distinct targets using our single target SDS baseline, and directly interpolate the vertex positions between the two meshes. We then compare this naive vertex-wise interpolation baseline to results generated from our BSD method in Figure 18. We notice that the shape interpolation baseline does not enable new features to emerge for the interpolated shapes, while our 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).

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 mix-

TextDeformer



Ours

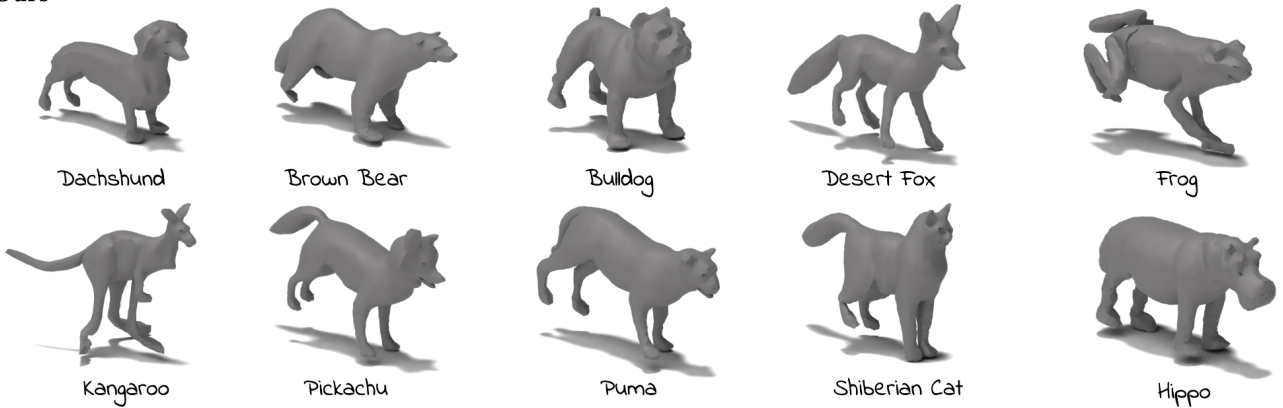


Figure 19. **Comparison of our method and TextDeformer.** MeshUp archives higher detail deformation results with less artifacts.

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.

ing 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 intuitively plausible manner. The set of renderings that we use for both evaluations can be found in

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.

Figure 19.

Quantitative Comparison We use the same dataset to quantitatively compare our method to TextDeformer using

Method	CLIP R-Precision \uparrow		
	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 [46] scores.

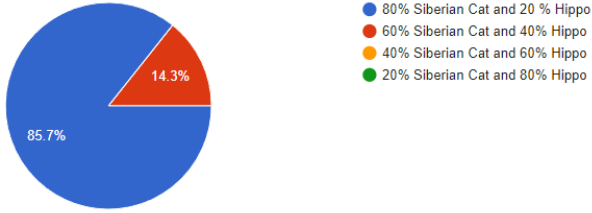


Figure 20. User response for 80% Siberian cat and 20% Hippo result.

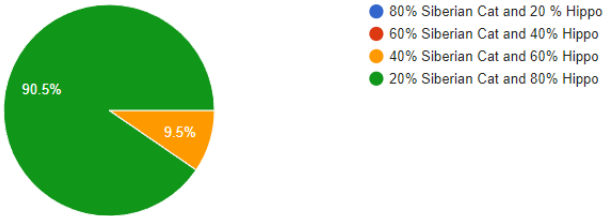


Figure 21. User response for 20% Siberian cat and 80% Hippo result.

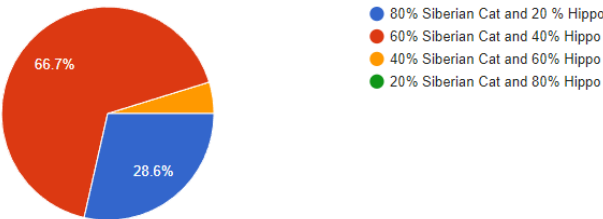


Figure 22. User response for 60% Siberian cat and 40% Hippo result.

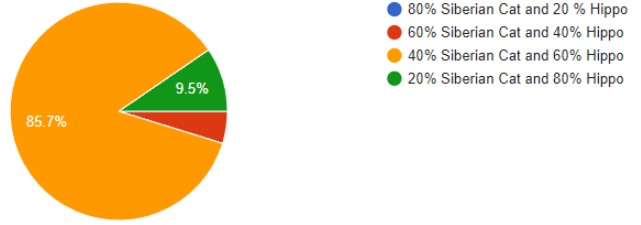


Figure 23. User response for 40% Siberian cat and 60% Hippo result.

former, since its optimization is directly supervised by CLIP score.

CLIP R-Precision Scores. We use a source dog mesh and warp it to the following 10 different prompts: “bear”, “bull-dog”, “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, despite the metric being inherently favorable to TextDe-

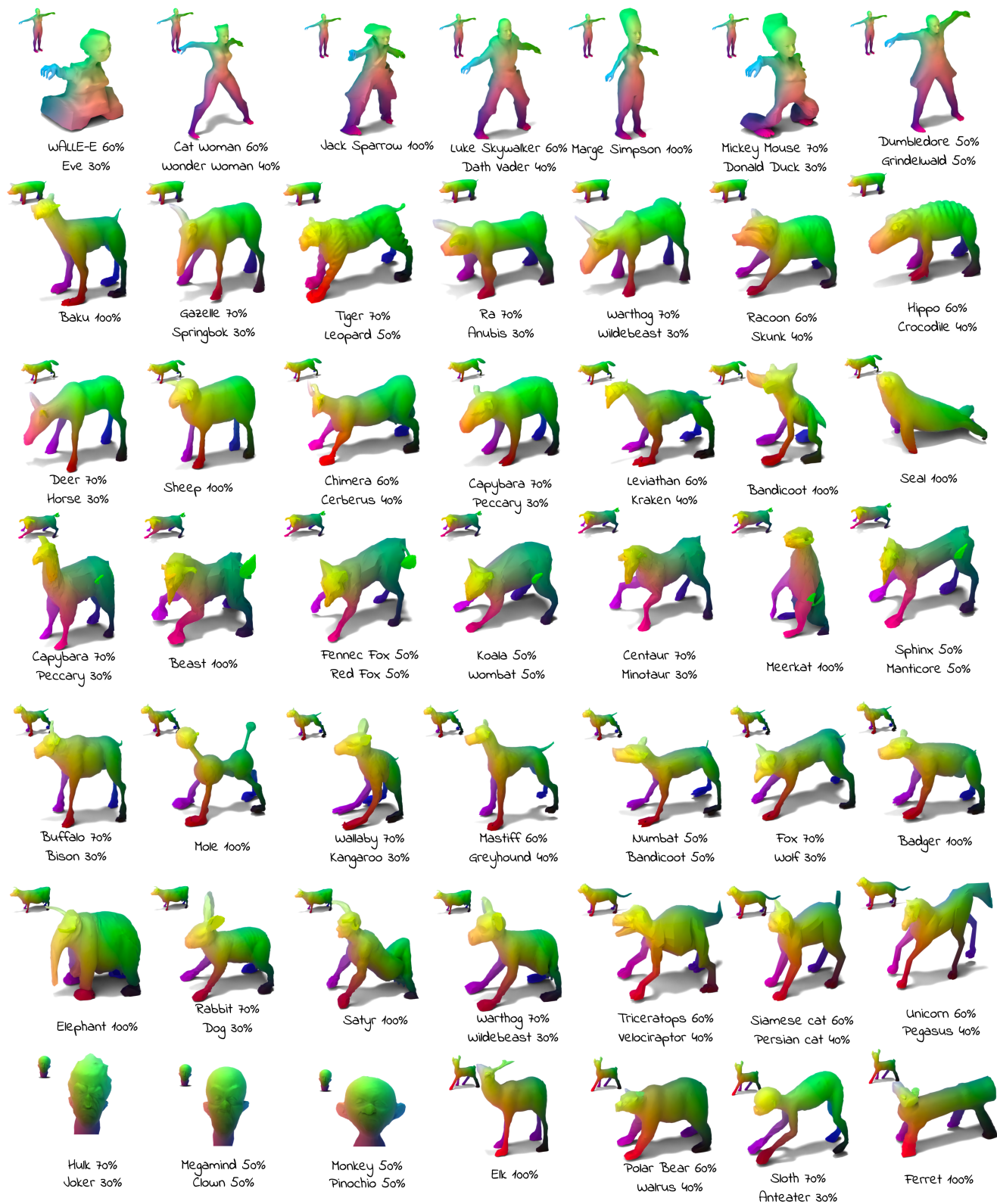


Figure 24. **Gallery.** We present a diverse set of 2-way MeshUp deformation results.