Avatar Concept Slider: Manipulate Concepts In Your Human Avatar With Fine-grained Control

Yixuan He,* **Lin Geng Foo***
Singapore University of Technology and Design
{yixuan_he, lingeng_foo}@mymail.sutd.edu.sg

Ajmal Saeed Mian
University of Western Australia
ajmal.mian@uwa.edu.au

Hossein Rahmani
Lancaster University
h.rahmani@lancaster.ac.uk

Jun Liu †
Singapore University of Technology and Design
jun_liu@sutd.edu.sg

Abstract

Language based editing of 3D human avatars to precisely match user requirements is challenging due to the inherent ambiguity and limited expressiveness of natural language. To overcome this, we propose the Avatar Concept Slider (ACS), a 3D avatar editing method that allows precise manipulation of semantic concepts in human avatars towards a specified intermediate point between two extremes of concepts, akin to moving a knob along a slider track. To achieve this, our ACS has three designs. 1) A Concept Sliding Loss based on Linear Discriminant Analysis to pinpoint the concept-specific axis for precise editing. 2) An Attribute Preserving Loss based on Principal Component Analysis for improved preservation of avatar identity during editing. 3) A 3D Gaussian Splatting primitive selection mechanism based on concept-sensitivity, which updates only the primitives that are the most sensitive to our target concept, to improve efficiency. Results demonstrate that our ACS enables fine-grained 3D avatar editing with efficient feedback, without harming the avatar quality or compromising the avatar's identifying attributes.

1 Introduction

Creating high-fidelity human avatars and editing them according to user demands has shown its importance in multiple scenarios such as game development, film production and virtual character creation for the metaverse and live streaming applications [1, 42, 39]. To meet this demand, many works [16, 20, 19, 24, 47, 26, 4] have made significant progress in generating 3D human avatars with simple yet user-friendly language prompts, demonstrating a strong ability to generate high-fidelity and realistic avatars. At the same time, there has also been an increased demand for better control over the creation and customization of personalized digital avatars [46], e.g., changing the hair style, clothing, or adding some accessories to the avatar. To achieve this while avoiding the time-consuming process of repeated generation and prompt modifications, *avatar editing* techniques, which aim to edit and sculpt some target details of an avatar while preserving most of the original identifying information, have emerged as a promising paradigm.

Due to its importance, avatar editing has attracted much research attention recently [46, 30, 2, 11]. Drawing inspiration from the successes achieved in harnessing diffusion models for 3D content generation [34, 25, 40], previous works such as HeadSculpt [11] edit avatars by leveraging an instruction-guided diffusion model [3]. Likewise, TECA [46] performs localized editing using a text-driven diffusion model with guidance from a 2D segmentation model [27].

Despite the significant progress, existing 3D avatar editing methods [46, 11] are limited due to their reliance on text prompts as the sole guidance signals. Specifically, text prompts can be quite

---

*Equal Contribtion, † Corresponding Author. Project page: Link

ambiguous, making it difficult to edit the avatar to *precisely align with* the users' expectations. For instance, to describe the desired hair length of a human avatar, users may prefer descriptive words such as 'long' or 'short' – which are ambiguous and limited in expressing the exact degree of length. Moreover, this difficulty in achieving precise editing with text alone becomes even more evident when manipulating complex concepts like 'sharpness' of facial features, or other adjectives which may be hard to put into words, e.g., 'kind' or 'evil'. In these cases, multiple aspects of the avatar need to be simultaneously considered, such as the shape of each facial feature, body posture, hairstyle and so on, which are very challenging to be clearly expressed in one or even a few text prompts, much less allowing precise control over them.

Driven by the problems mentioned above, our aim in this work is to *precisely edit target concepts* of avatars in a convenient manner, akin to moving the knob on a continuous slider bar, from which users may select *exact values along the bar* to specify their desired avatar. An overview is shown in Fig. 1. However, achieving this aim is challenging, mainly due to three reasons: ① For each concept (e.g., body shape), it can be challenging to find such a continuous bar or axis to express the intermediate degree of avatar concepts, since these concepts are often high-level abstract concepts that exist beyond the pixel level. This makes it difficult to identify where these high-level concepts reside in feature
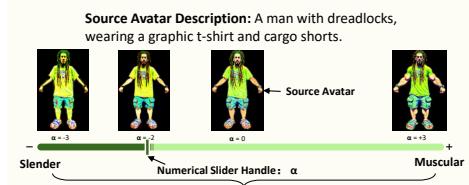

Figure 1: Illustration of our fine-grained concept editing. In this example, the user has provided text descriptions for two opposing ends of a concept: 'slender' vs 'muscular'. Our ACS allows users to precisely specify the exact level of concept expression (e.g., amount of muscle) that is desired, by moving the knob on the slider bar.

space, as well as pinpoint the precise axis that links the two opposing ends of a specific concept within the large feature space. ② Furthermore, it is challenging to isolate and edit only the desired concept without changing other identifying information of the avatar. This is because each avatar is generated with a mixture of elements, such as face shape, hairstyle, and clothing type, which are also entangled with concepts such as gender, race, and age. While editing the target concept, we may inevitably alter other attributes that we want to retain, and it is challenging to prevent this from happening. ③ Lastly, while achieving the slider-like fine-grained control, the editing process should also ideally be quick, to provide quick feedback so that users can iteratively refine the avatar towards the desired degree of concept expression (e.g., age of the avatar) by adjusting the slider knob. This puts high demands on the rendering and editing speed.

In this work, we propose a novel Avatar Concept Slider (ACS) for 3D human avatar editing, which enables users to edit their 3D avatar *precisely towards the desired degree of expression* of a given concept, offering much more fine-grained manipulation of concepts than with text inputs alone. To overcome the three challenges highlighted above, our ACS comprises three corresponding designs, as follows: ① To pinpoint the precise concept axis that links the two opposing ends of a specific concept, we introduce the Concept Sliding Loss based on Linear Discriminant Analysis (LDA) to fine-tune an adapter. This facilitates precise slider-like editing using the score distillation sampling (SDS) optimization editing pipeline. LDA facilitates the identification of a continuous 1D axis (which we call the concept axis) that is the most discriminative in linking the two opposing sides of the user-provided concept. ② To preserve key identifying attributes during editing, we further devise an Attribute Preserving Loss based on Principal Component Analysis (PCA), leveraging PCA to extract key attribute information from the features that are orthogonal to the target concept. Then, by encouraging these key attribute information to be retained, it enables users to edit their target concept in isolation, disentangled from other identifying attributes. ③ For higher efficiency in rendering and optimization, we employ the 3D Gaussian Splatting (3DGS) representation for our avatar during editing. Moreover, we further propose a 3DGS primitive selection mechanism based on their concept-sensitivity, which enables us to selectively optimize a small set of the most crucial Gaussian primitives to reduce redundancy and improve efficiency. With the above designs incorporated into the deployed ACS, users can input descriptions of two opposing ends of a target concept, e.g., with a pair of text phrases. Subsequently, users can finely adjust the degree of expression of a desired concept on a given avatar, by simply moving a knob across a slider bar.

2 Related Work

3D Gaussian Splatting (3DGS). Recently introduced by [23], 3DGS is a 3D representation that has attracted much attention across various domains due to its fast rendering speed. For instance,

some works explore using 3DGS to model dynamic scenes [41, 28] or for SLAM [18, 22], leading to improved rendering speed and visual quality in both cases. Moreover, some works also explore using 3DGS for generative models, e.g., text-to-3D content generation [37, 45, 6], or human avatar generation [51, 26]. In this work, we employ 3DGS due to its efficiency and quality. To further improve the rendering speed of 3DGS for our avatar editing task, we introduce a mechanism to selectively edit only the primitives that are the most relevant to the target concept.

Text-Driven 3D Human Avatar Generation. To streamline the labour-intensive 3D avatar creation process, many recent works [16, 20, 48, 47, 4, 24, 19] focus on text-driven 3D avatar generation, relying solely on text prompt inputs to create a desired high-fidelity 3D human avatar. One line of works optimize their avatar model with text guidance in CLIP [35] space, e.g., AvatarCLIP [16]. More recent works such as DreamHuman [24], DreamAvatar [4], DreamWaltz [19], and AvatarVerse [47] rely on a text-to-image 2D diffusion prior (e.g., [36, 49]) to optimize their 3D avatar via an SDS optimization pipeline [34]. More recently, HumanGaussian [26] proposes to use 3DGS to represent the human avatar, resulting in faster convergence and high-quality avatars. However, these text-driven avatar generation methods are limited by the expressivity and ambiguity that is inherent in language, and often struggle with precisely generating the user-desired avatars with text prompts alone.

Text-Driven 3D Human Avatar Editing. Besides generation, some works [30, 46, 2] focus on editing existing 3D human avatars, sculpting the avatars according to the desired shape and texture. AvatarStudio [30] presents a text-driven method to edit head avatars based on the neural volumetric scene representation [38]. TECA [46] leverages a segmentation model to achieve localized editing based on text prompts. Another related line of works [12, 50, 5] further investigates the editing of entire 3D scenes via text instructions. Despite impressive generation and editing results, these methods are limited by their reliance on language instructions, which can be ambiguous regarding the degree of expression of the target concept. Beyond these works, this paper aims at improving avatar editing methods with finer-grained control, enabling users to manipulate the avatar more precisely with respect to a target concept.

3 Background: SDS Optimization Pipeline

Score Distillation Sampling (SDS) [34] leverages 2D diffusion models as a strong prior for 3D representation learning, e.g., text-to-3D content generation [25, 20] and 3D avatar editing [30]. Given a 3D representation θ (e.g., NeRF [31], 3DGS [23]), SDS optimization pushes the renderings of the 3D representation towards the target distribution (e.g., high-quality natural images), by distilling the predicted score from a pre-trained 2D diffusion prior. By using *text-to-image* diffusion models [36] as a prior, we can facilitate *text-to-3D* generation by optimizing the 3D representation via SDS.

Specifically, we consider the case where a latent diffusion model (e.g., Stable Diffusion [36]) is used, which is denoted as $\epsilon_\phi(\cdot)$. SDS optimization starts by rendering the initial 3D representation θ into 2D images, and encoding them into latent space to obtain the initial latent features as $\mathbf{z}_0$. Then, to derive the SDS loss $\mathcal{L}_{SDS}$, noise is added to the latent features $\mathbf{z}_0$ based on t steps of forward diffusion [36] to obtain the noised latents $\mathbf{z}_t$, where the SDS loss aims to denoise $\mathbf{z}_t$. Therefore, the gradient of the SDS loss with respect to the 3D representation θ can be formulated as: $\nabla_\theta \mathcal{L}_{SDS}(\phi, \mathbf{z}_0) = \mathbb{E}_{t,\epsilon}[w(t)(\epsilon_\phi(\mathbf{z}_t; y, t) - \epsilon)\frac{\partial \mathbf{z}_0}{\partial \theta}]$, where $w(t)$ is a weighting function that depends on time step t, and y is a text prompt that describes the desired 3D output.

Notably, to add functionalities, control options, or editing capabilities (e.g., with dense pose [47], face landmarks [4]) to the SDS optimization process while leveraging a pre-trained diffusion model, many works use LoRA adapters [17]. Specifically, for a selected set of weight parameters (denoted as ϕ_0) in the diffusion prior ϵ_ϕ, LoRA aims to learn a low-rank weight shift $\Delta\phi$ to obtain adapted weights $\hat{\phi}$ as: $\hat{\phi} = \phi_0 + \alpha\Delta\phi$, where α is the *scale factor* of the weight shift. By adapting the diffusion prior ϵ_ϕ with $\Delta\phi$ and α, the SDS optimization can be defined as:

$$\nabla_\theta \mathcal{L}_{SDS}(\phi, \mathbf{z}_0) = \mathbb{E}_{t,\epsilon}\left[w(t)\Big(\epsilon_\phi\big(\mathbf{z}_t; y, t, (\Delta\phi, \alpha)\big) - \epsilon\Big)\frac{\partial \mathbf{z}_0}{\partial \theta}\right]. \tag{1}$$

Overall, this mechanism allows us to efficiently adapt a diffusion model with a small set of parameters $\Delta\phi$ to cater for specific requirements, e.g., for editing, or for using additional control information. Hence, we also adopt the SDS optimization and LoRA adapter pipeline as introduced in Eq. 1.
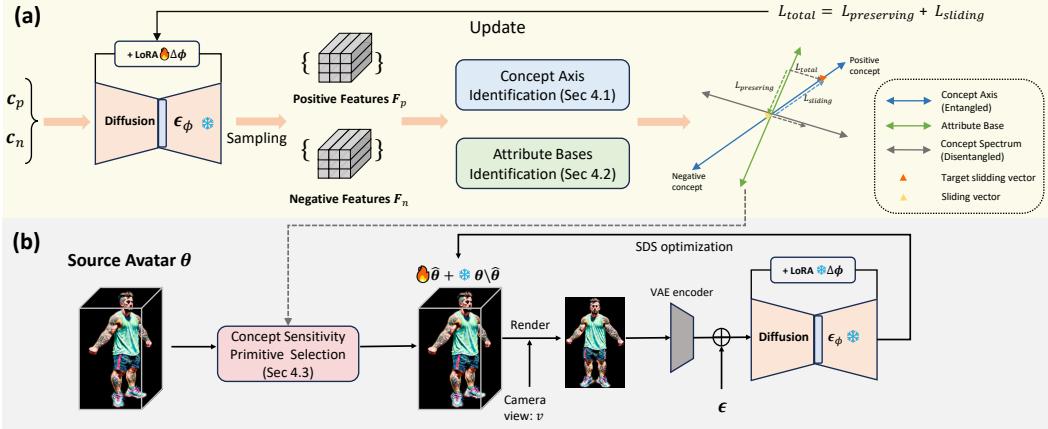
Figure 2: **(a)** Overview of the fine-tuning stage, where an adapter is fine-tuned to learn slider-like capabilities. Firstly, using the provided descriptions of the positive (c_p) and negative (c_n) side of the concept, we extract the corresponding positive and negative features. Then, to fine-tune the adapter $\Delta\phi$, our proposed Concept Sliding Loss (Sec. 4.1) is applied to learn the ability to slide across opposing ends of the target concept. The attribute-preserving loss (Sec. 4.2) helps to retain the key identifying attributes of the avatar. **(b)** After fine-tuning, the adapter is applied in an SDS optimization pipeline to achieve fine-grained concept-specific 3D avatar editing. For fast rendering and optimization speeds, we leverage 3DGS representation for our 3D avatar. The proposed primitive selection mechanism (Sec. 4.3) further improves efficiency, by optimizing only the most related primitives to the target concept.

4 Method: Avatar Concept Slider

To customize a human avatar, previous text-driven 3D avatar editing methods [30, 46, 11] are limited in terms of precision. To overcome this limitation, this paper aims to achieve precise and fine-grained manipulation of 3D avatars towards a target concept in a convenient manner, akin to sliding a knob along a slider bar. However, achieving this goal is challenging due to the three reasons outlined in the introduction. To address the three challenges, we propose our Avatar Concept Slider (ACS), discussed in the following subsections. In Sec. 4.1, we address challenge ① by leveraging LDA and devising a sliding loss to enable slider-like fine-grained control of the editing process. To tackle challenge ②, discussed in Sec. 4.2, we augment the sliding loss with an attribute-preserving loss based on PCA, ensuring concept-specific editing that is disentangled from the identifying attributes to be preserved. Lastly, in Sec. 4.3, we introduce a concept-sensitivity-based primitive selection mechanism, facilitating faster editing via selective optimization of our avatar's 3DGS primitives, for handling challenge ③. Overall, our pipeline is built upon SDS optimization (introduced in Sec. 3), and an illustration of our full method is shown in Fig. 2.

4.1 Precise Slider-like Concept Editing

To precisely control the 3D avatar editing towards a specific degree of the target concept, the first challenge is to identify a precise and continuous axis connecting two opposing extremes of the concept. In particular, this can be challenging because the concepts are often abstract and high-level, making it difficult to pinpoint where exactly these opposing sides and their connecting axis reside in the large feature space. To overcome this, we leverage LDA to identify a continuous concept axis between the user-provided pair of descriptions for the two opposing sides, directly computing a 1D axis that is the most discriminative to link the opposing sides within the large feature space. Then, based on the identified concept axis, we introduce a Concept Sliding Loss ($\mathcal{L}_{\text{sliding}}$) to fine-tune an adapter, which enables fine-grained concept-specific editing via SDS optimization (Sec. 3). Below, we first describe how we identify a concept axis, before discussing the Concept Sliding Loss.

Concept Axis Identification with LDA. First, we aim to identify a continuous concept axis between the user-provided pair of descriptions that represent two opposing sides of the target concept, where each point on the axis represents an intermediate degree of expression of the concept. By identifying the concept axis, we can *precisely control the transitioning of the concept* from one end to the other, and facilitate the learning of sliding capability via the sliding loss, enabling users to select any intermediate point along the concept axis for precise editing. Yet, this task is not straight forward. For instance, a simple solution may be to interpolate the semantic embeddings of two concept extremes,

but such simple interpolation operations in a high-dimensional space often incurs high information redundancy and concept entanglement, incorporating many other unrelated concepts.

Therefore, to handle this challenge, we leverage two-class LDA. Specifically, LDA [13, 29] is a statistical technique that aims to identify the crucial information that can well-separate different groups/classes of data samples, even while using a low dimensional representation. Thus, by employing two-class LDA, we are able to directly compute a 1D axis (resembling a two-sided slider bar) that is the most crucial and discriminative to link the opposing concepts (and their corresponding sets of features) within the large feature space. More precisely, we harness LDA to exploit the latent space of the pre-trained text-to-image diffusion model used during SDS optimization (Sec. 3) to facilitate understanding of the target concept in a high-quality and consistent latent space.

Here, we present the details of how we identify a concept axis with two-cla ss LDA. At the start, the user provides a pair of text prompts representing the positive (c_p) and negative (c_n) sides of a concept. Then, based on these prompts, we sample two sets of latent features from the diffusion model ϵ_ϕ by running the generation process N_s times for each of the positive and negative sides, where we perform sampling over different diffusion timesteps using different seeds. This yields two sets of features $\{\mathbf{F}_{p,i,x,y}\}_{i\in[1,N_s],x\in[1,H],y\in[1,W]}$ and $\{\mathbf{F}_{n,i,x,y}\}_{i\in[1,N_s],x\in[1,H],y\in[1,W]}$ corresponding to the positive and negative sides, where H, W refer to the spatial dimensions of latent features. Each feature vector is D-dimensional, i.e., $\mathbf{F}_{p,i,x,y} \in \mathbb{R}^D$, $\mathbf{F}_{n,i,x,y} \in \mathbb{R}^D$. For clarity, we do not incorporate the diffusion time step in this formulation.

We then start the LDA analysis [13, 29] using these two feature sets for the two opposing sides of the target concept. Specifically, we first compute both the within-class scatter ($\mathbf{S}_w \in \mathbb{R}^{D\times D}$) and the between-class scatter ($\mathbf{S}_b \in \mathbb{R}^{D\times D}$), where $\mathbf{S}_w$ quantifies how much individual feature vectors deviate from their corresponding set centroids, while $\mathbf{S}_b$ measures the deviation between the centroids of both sides of the concept (more details in Supplementary). Then, using $\mathbf{S}_b$ and $\mathbf{S}_w$, the goal of LDA is to find a *concept axis* $\mathbf{b}_c \in \mathbb{R}^D$, a 1D axis such that the projection of each feature vector onto the concept axis $\mathbf{b}_c$ leads to maximal ratio of $\mathbf{S}_b$ to $\mathbf{S}_w$. Intuitively, this encourages the feature vectors from both sets to be separated and positioned far from each other along the concept axis, while the feature vectors from the same set become tightly clustered along the concept axis. By doing so, LDA effectively pinpoints the direction of the 1D concept axis, along which the opposing ends of the concept reside. Specifically, this process can be expressed as the following optimization problem:

$$\mathbf{b}_c = \arg\max_{\mathbf{w}} \frac{\mathbf{w}\mathbf{S}_b\mathbf{w}^T}{\mathbf{w}\mathbf{S}_w\mathbf{w}^T}, \text{s.t. } ||\mathbf{w}|| = 1, \tag{2}$$

where the numerator ($\mathbf{w}\mathbf{S}_b\mathbf{w}^T$) and denominator ($\mathbf{w}\mathbf{S}_w\mathbf{w}^T$) on the right hand side of Eq. 2 represent the projected scatters over the concept axis, and we want to find a concept axis $\mathbf{b}_c$ that maximises their ratio while constraining $\mathbf{w}$ to be a unit vector. Notably, the solution to Eq. 2 can be conveniently computed as the leading eigenvector of $\mathbf{S}_w^{-1}\mathbf{S}_b$ (refer to Supplementary).

Concept Sliding Loss. The above process identifies a concept axis which represents the continuous transition between the two opposing sides of the concept under consideration. Based on the identified concept axis, we introduce a Concept Sliding Loss to fine-tune an adapter to facilitate concept-specific editing of the 3D avatar. Specifically, since we adopt the SDS optimization pipeline (Sec. 3) to edit the 3D avatar via a pre-trained diffusion model, we aim to incorporate such concept-specific slider-like editing abilities into it. Hence, we propose a Concept Sliding Loss ($\mathcal{L}_{\text{sliding}}$) to impart such concept transitioning abilities to the adapter, enabling us to precisely control the editing through a scale factor α in Eq. 1.

Intuitively, our Concept Sliding Loss $\mathcal{L}_{sliding}$ facilitates the controlled movements of features along the concept axis towards a specified intermediate point, which is specified by the user as a *slider interpolation factor*. Specifically, to control such movements of features during editing, our loss fine-tunes the adapter ($\Delta\phi$) such that the scale factor α in Eq. 1 gains the ability as the *slider interpolation factor*. Thus, at deployment, with the user-provided interpolation factor $\hat{\alpha}$, we can set it as the scale factor (i.e., setting $\alpha = \hat{\alpha}$), thus allowing the adapted diffusion model to precisely locate the desired intermediate point on the concept axis. The Concept Sliding Loss $\mathcal{L}_{sliding}$ is defined as:

$$\mathcal{L}_{sliding}(\Delta\phi) = \mathbb{E}_{\alpha\in[-1,1]}\Big[||\frac{\mathbf{f}(\alpha, \Delta\phi, c_{nu}) \cdot \mathbf{b}_c}{\mathbf{b}_c \cdot \mathbf{b}_c}\mathbf{b}_c^T - (\frac{1+\alpha}{2}\boldsymbol{\mu}_p + \frac{1-\alpha}{2}\boldsymbol{\mu}_n)||\Big], \tag{3}$$

where $\mathbf{f}(\cdot)$ represents the generation of latent features, c_{nu} is a neutral text prompt template, $|| \cdot ||$ represents L2 distance, $\boldsymbol{\mu}_p, \boldsymbol{\mu}_n \in \mathbb{R}^D$ are computed mean vectors for the positive and negative concept sides (see Supp for more details), and various α values are sampled within $[-1, 1]$ to

represent intermediate concepts. Intuitively, $\frac{\mathbf{f}(\alpha, \Delta\phi, c_{nu}) \cdot \mathbf{b}_c^T}{\mathbf{b}_c \cdot \mathbf{b}_c} \mathbf{b}_c^T$ is the projection of $\mathbf{f}(\alpha, \Delta\phi, c_{nu})$ onto the concept axis $\mathbf{b}_c$, and this loss tries to *match the projection's position with the desired location along the concept axis*, which is the interpolation between $\boldsymbol{\mu}_n$ and $\boldsymbol{\mu}_p$ as: $\left(\frac{1+\alpha}{2} \boldsymbol{\mu}_p + \frac{1-\alpha}{2} \boldsymbol{\mu}_n \right)$. Overall, this loss can fine-tune the adapter parameters $\Delta\phi$ such that the SDS optimization pipeline in Eq. 1 can be used to smoothly manipulate our avatar by adjusting the scale factor α according to the user-input slider interpolation factor. Note that, with this loss design, we only need to identify the concept axis once (via LDA), and it can be used for the entire adapter fine-tuning process.

4.2 Preserving of Identity-related Attributes

While the Concept Sliding Loss presented above facilitates learning and representing intermediate points between opposing sides of a concept, it may inadvertently alter other identifying attributes of the avatar that we want to retain. This challenging issue arises as the 3D avatars contain many elements (e.g., face shape, hairstyle and clothing type) that involve entangled concepts such as gender, race, and age, as highlighted in challenge ② in the introduction. Isolating and editing only the desired concept, while retaining the other identifying information of the avatar is challenging. To address this, we leverage Principal Component Analysis (PCA) to disentangle the concepts of the avatar and retain key identifying attributes. PCA helps to identify a small set of highly informative bases that are orthogonal to our concept axis, representing key identifying attributes unrelated to the target concept. Then, we propose an Attribute Preserving Loss $\mathcal{L}_{preserving}$ based on these key attribute bases found via PCA, enabling fine-tuning of the diffusion model adapter to preserve identifying attributes. Below, we first introduce the bases identification with PCA, followed by the attribute preserving loss.

Attribute Bases Identification with PCA. Firstly, we reuse the features $\mathbf{F}_{p,i,x,y}$ and $\mathbf{F}_{n,i,x,y}$ that were computed in Sec. 4.1, which represent the positive and negative sides of the target concept. Then, we run PCA analysis on these features to identify a small set of key attributes in the feature space that are not related to our target concept. This is achieved through finding the K principal components (i.e., orthogonal bases $\{\mathbf{b}_{a,k}\}_{k=1}^K$) which are orthogonal to our concept axis. Specifically, one way of performing PCA is through iteratively searching for the next principal component, which explains as much of the remaining variance of the data as possible, while being orthogonal to the previous identified principal components. Here, we recurrently employ this PCA technique to compute the main component vectors orthogonal to our concept axis, formulated as:

$$\mathbf{b}_{a,k} = \arg\max_{\mathbf{w}} \frac{\mathbf{w}\hat{\mathbf{F}}_k \hat{\mathbf{F}}_k^T \mathbf{w}^T}{\mathbf{w}\mathbf{w}^T}, \text{ where } \hat{\mathbf{F}}_k = \hat{\mathbf{F}} - \sum_{j=1}^{j-1} \hat{\mathbf{F}} \mathbf{b}_{a,j-1}^T \mathbf{b}_{a,k-1}, \tag{4}$$

where $\hat{\mathbf{F}} \in \mathbb{R}^{D \times (2 \cdot H \cdot W \cdot N_S)}$ is the combined merger of both positive and negative sets of features ($\{\mathbf{F}_{p,i,x,y}\}_{i\in[1,N_s],x\in[1,H],y\in[1,W]}$ and $\{\mathbf{F}_{n,i,x,y}\}_{i\in[1,N_s],x\in[1,H],y\in[1,W]}$), and $\mathbf{w}$ is constrained to be a unit vector. Crucially, by treating the first 'principal component' $\mathbf{b}_{a,0}$ as our concept axis $\mathbf{b}_c$ identified in Eq. 2 and iteratively solving Eq. 4 for $k = \{1, ..., K\}$, we can obtain K key attribute bases $\{\mathbf{b}_{a,k}\}_{k=1}^K$ that best explain the remaining variance in the features. Furthermore, these obtained bases are orthogonal to our concept axis $\mathbf{b}_c$, which means they are key attributes that are unrelated to the target concept. Note that, although the small set of K attribute bases has only a low rank, they capture the most important identifying information in the avatar that should be retained.

Attribute Preserving Loss. With the identified attribute bases above that represent important key identifying attributes which should be retained, we introduce our attribute preserving loss to fine-tune the adapter ($\Delta\phi$) to disentangle its editing effects, avoiding the editing of identifying attributes of the avatar. In other words, when users adjust the scale factor (α) to perform editing, we expect the other key attributes to remain unchanged, as if the adapter is not applied for those other attributes. Thus, the attribute preserving loss $\mathcal{L}_{preserving}$ can be defined as:

$$\mathcal{L}_{preserving}(\Delta\phi) = \mathbb{E}_{\alpha \in [-1,1]} \Big[\sum_{k=1}^K || \frac{\mathbf{f}(\alpha, \Delta\phi, c_{nu}) \cdot \mathbf{b}_{a,k}}{\mathbf{b}_{a,k} \cdot \mathbf{b}_{a,k}} \mathbf{b}_{a,k}^T - \frac{\mathbf{f}(\varnothing, c_{nu}) \cdot \mathbf{b}_{a,k}}{\mathbf{b}_{a,k} \cdot \mathbf{b}_{a,k}} \mathbf{b}_{a,k}^T || \Big], \tag{5}$$

where $\mathbf{f}(\varnothing, c_{nu})$ represents the original latent features without applying the adapter. Intuitively, this loss compares the features after adaptation with the features that have not undergone adaptation, comparing them along a few key orthogonal dimensions that represent the key attributes of the avatar which should be retained. Importantly, this loss helps to optimize the adapter parameters ($\Delta\phi$) to disentangle the effects of editing along the concept axis from the other main attributes and dimensions that are not related to the concept axis, encouraging the other attributes to be retained.

Overall, by applying both the sliding loss $\mathcal{L}_{sliding}$ (Eq. 3) and the attribute preserving loss $\mathcal{L}_{preserving}$ (Eq. 5) to fine-tune the adapter $\Delta\phi$ (see Fig. 2(a)), the model gains the ability to perform precise

concept-specifc editing via the SDS optimization pipeline in Eq. 1. Thus, when the user manipulates the scale factor α at deployment time, the model can edit the 3D avatar towards a precise point along the concept axis, while maintaining the key identifying features.

4.3 Concept Sensitivity-Aware 3DGS Primitive Selection

The previous subsections introduced two loss criteria to fine-tune a concept-specific diffusion adapter, enabling precise control of avatar concepts through the SDS optimization pipeline. However, as mentioned in challenge ③, applying this slider-like control in practical applications often requires fast feedback from the model, to enable users to iteratively edit avatars towards their desired results, which *necessitates high rendering and optimization speeds*.

Therefore, to improve the rendering and optimization speed of our editing pipeline, we explore the 3DGS representation for fast 3D avatar editing. 3DGS is known for its efficiency in rendering and optimization [26, 21, 32], and has recently been of great interest for generation of 3D objects [37, 44, 7] and 3D humans [26]. Thus, considering its efficiency, we explore using 3DGS representation for our 3D avatar editing process. Specifically, the 3DGS representation consists of a set of M Gaussian primitives which we denote as $\{\boldsymbol{\theta}_i\}_{i=1}^M$, where each i-th Gaussian primitive contains a set of parameters $\{\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i, \sigma_i, \mathbf{h}_i\}$ that determine their location ($\boldsymbol{\mu}_i$), shape ($\boldsymbol{\Sigma}_i$), opacity (σ_i), and color ($\mathbf{h}_i$). During rendering, the color of each pixel is computed based on all the parameters of all M primitives $\{\boldsymbol{\theta}_i\}_{i=1}^M$ in a differentiable manner, which facilitates the backpropagation of gradients.

However, we find that optimizing the entire 3DGS representation (i.e., all M primitives) yields sub-optimal results for 3D editing. This is because, not all primitives contribute to the final rendering equally [33], some of them are not so important, and so optimizing all primitives incurs high redundancy and harms optimization and editing efficiency, In addition, our concept-specific 3D editing task typically requires to modify only *a small subset of primitives* that contribute the most towards the target concept. Thus, we select and edit only a subset of primitives, as discussed below.

Concept-Sensitive Primitive Selection. We propose a primitive selection mechanism based on concept-sensitivity to further improve the efficiency of the 3DGS representation in our editing pipeline. Instead of editing all M 3DGS primitives, we only edit the primitives that contribute the most towards our target concept. To achieve this, our mechanism leverages the concept axis identified in Sec. 4.1 to quantify the contribution for each primitive, by computing a *Concept Sensitivity score* that measures how much each primitive is related to the target concept, i.e., how the changes in the primitive parameters affect the precise location along the concept axis. Then, by selecting primitives with a high Concept Sensitivity score and abediting only these primitives, we can effectively reduce the redundancy during the editing stage. We describe the details below.

Firstly, to compute the sensitivity of Gaussian primitives with respect to the target concept (i.e., the identified concept axis $\mathbf{b}_c$ in the diffusion model's latent space as discussed in Sec. 4.1), we need to express our 3D avatar in the same latent space. To do this, we first render our 3D avatar – which is represented by a set of primitives $\{\boldsymbol{\theta}_i\}_{i=1}^M$ – to V randomly sampled views, yielding a set of rendered images. These rendered images are then passed into the diffusion model, to extract a set of latent features as $\{\mathbf{z}_{v,x,y}\}_{v\in[1,V],x\in[1,H],y\in[1,W]}$, where $\mathbf{z}_{v,x,y} \in \mathbb{R}^D$. These latent features $\mathbf{z}_{v,x,y}$ belong to the same latent space as $\mathbf{F}_{p,i,x,y}$ and $\mathbf{F}_{n,i,x,y}$ in Sec. 4.1.

Next, we calculate the Concept Sensitivity score based on these latent features. Intuitively the Concept Sensitivity score measures the sensitivity of each primitive towards the target concept, i.e., how much movement there is along the identified concept axis $\mathbf{b}_c$ given a small change in the primitive's parameters. To compute this, we first compute the degree of alignment $\bar{C}$ between latent features $\mathbf{z}_{v,x,y}$ and concept basis $\mathbf{b}_c$ via the dot product as follows: $\bar{C} = \sum_{v=1}^V \sum_{x=1}^H \sum_{y=1}^W \mathbf{b}_c \mathbf{z}_{v,x,y}^T$. The higher the value of $\bar{C}$, the more aligned the 3D human avatar is towards the target concept. Hence, if we take a gradient of $\bar{C}$ with respect to the primitive parameters, it will *measure how much movement* there is along the concept axis, given a small change in the primitive, i.e., the sensitivity of the target concept with respect to each primitive's parameters. Specifically, for each i-th Gaussian primitive $\boldsymbol{\theta}_i$ with parameters $\{\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i, \boldsymbol{\theta}_i, \mathbf{h}_i\}$, we compute its Concept Sensitivity score S_i as:

$$S_i = \sum_{p_i \in \{\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i, \sigma_i, \mathbf{h}_i\}} \left| \frac{\partial \bar{C}}{\partial p_i} \right|, \tag{6}$$

which sums the magnitude of the gradients w.r.t. each parameter. The higher the Concept Sensitivity score S_i, the more movement there will be along the concept axis given a small change in the i-th Gaussian primitive, which means that primitive is more crucial to editing of the concept. To select the
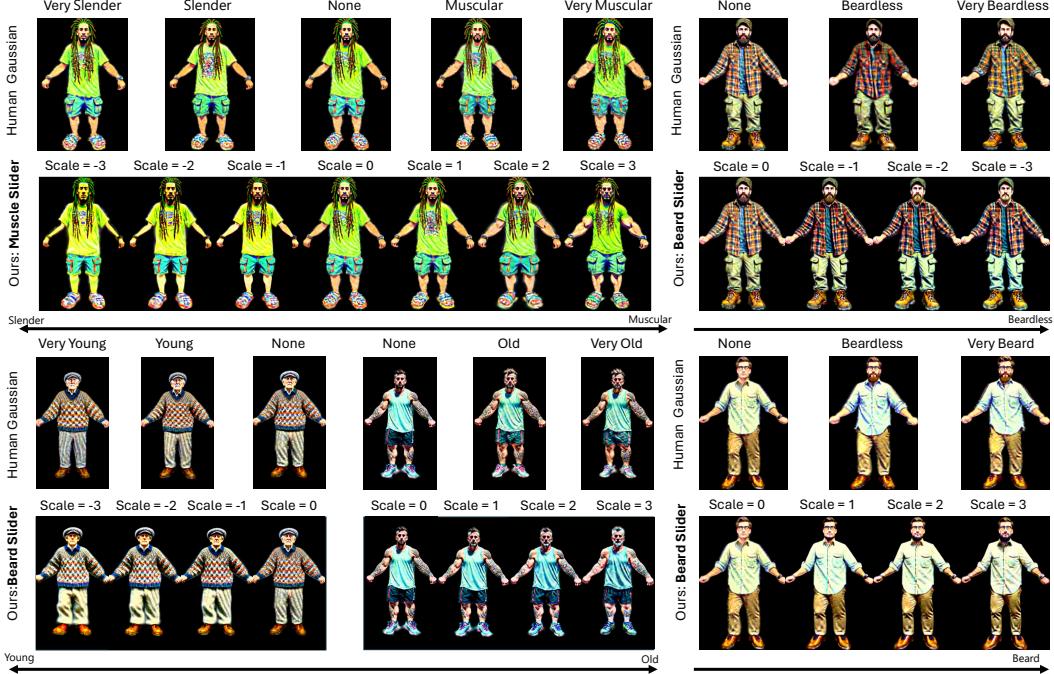
Figure 3: Qualitative Comparison of our method against existing SOTA [26].

most important primitives to edit, we pre-define a small fractional threshold γ, such that only the top γM primitives, i.e., $\{\theta_i\}_{i=1}^{\gamma M}$, are selected for SDS optimization, instead of all M primitives. Overall, this leads to much faster editing times, facilitating the fast feedback to users when using our ACS.

4.4 Implementation Details

Fine-tuning Details. We use the Depth-RGB dual branch diffusion model [26] as our diffusion prior. During fine-tuning, we weight the sliding loss and attribute preserving loss equally (coefficient value of 0.5 for each), and utilize LoRA adapter of rank 4. We set $N_s = 20$, and $K = 8$. The adapter is fine-tuned for 1000 steps with batch size 1, using AdamW optimizer (learning rate = $2e - 4$). For each concept-specific adapter, the whole fine-tuning process takes 1 hour and is executed once only.

Avatar Editing Details. The 3DGS implementation in our work is based on ThreeStudio framework [10] and the optimized renderer from [23]. To obtain source avatars to edit, we generate 3DGS avatars with [26]. Then, following the optimization settings of [26], we sample the camera distance from range $[1.5, 2.0]$, elevation from range $[40, 70]$, and azimuth from range $[-180, 180]$ with batch size of 4, for each editing update. The Adam optimizer is used for updating the mean, scale, rotation, color, and opacity of 3DGS with learning rate of $5e - 5, 1e - 3, 1e - 2, 1e - 2, 1e - 2$ respectively. We set $\gamma = 5$. The whole editing process undergoes 1200 updating steps (of SDS optimization), during which the densification and and pruning procedures are executed every 200 steps. The whole editing process for rendering resolution at 1024 costs 12 minutes on a single NVIDIA 4090 GPU.

5 Experiment Results

We compare our method with other text-based 3D human avatar generation and editing methods, such as [24, 19, 26]. For qualitative comparisons, we mainly compare with HumanGaussian [26] which is the state-of-the-art text-driven generation method for producing 3DGS avatars. We adopt their method while altering the text prompts to perform text-based editing.

Qualitative Results. First, we conduct qualitative evaluations on controllability editing and quality, as shown in Fig. 3. For fair comparison, source 3DGS avatars for both our method and the baseline are generated using the pipeline of [26]. To evaluate the capabilities of our ACS, we edit the source avatars to get results at various points on the slider, equally spaced apart. For the baseline method [26] (text-based only), we modify the text prompts by adding concept-related words (e.g., young, old) and adding adverbs to express degrees of expression, e.g., 'very'. Thus, for each text prompt, we can obtain 2 edited avatars from the baseline, which expresses the specified concepts to different degrees. Crucially, unlike the text-driven baseline, our method offers *precise controllability*, enabling users to select a slider factor to edit the avatar concept to a desired degree. Besides, our edited outcomes also tend to *better align with the provided descriptions*.
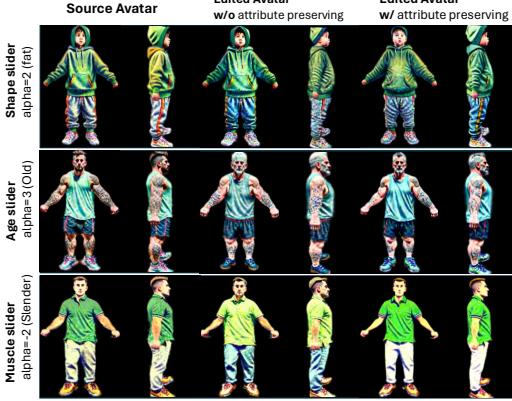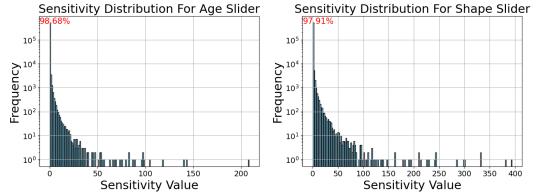
Figure 4: Impact of Attribute Preserving Loss

Figure 5: Distribution of Concept Sensitivity score for all primitives for 'Age' and 'Shape' concepts

Setting	Time Taken to converge
NeRF	91 min
3DGS	42 min
Ours	12 min

Table 3: Efficiency comparisons of our method.

Quantitative Results. We also evaluate our ACS in terms of editing quality with Frechet Inception Distance (FID) [14, 43, 15] which compares the rendered images against a real-world full-body human dataset (SHHQ [8]), and also the semantic similarity between edited avatars and corresponding prompts with the CLIP score [35]. Specifically, to compute *FID score*, we first use our ACS to edit 40 avatars with 10 sliders at 5 equally spaced slider factors, then render each edited avatar from 5 randomly sampled views, resulting in 10k ($40 \times 10 \times 5 \times 5$) images. For the baseline [24] and two other text-based generation methods [24] [19], we generate same number of avatars based on prompts with concepts adjusted to varying degrees via adverbs (e.g., 'moderately', 'very'), and render them from multiple views to obtain 10k images. Then, we compare these rendered images against 10k images from SHHQ [8] with the FID metric. To compute the *CLIP score* of the text-based methods [24] [24] [19], we augment each image's corresponding prompts with viewpoint information (front, side, back), resulting in 10,000 image-text pairs. The CLIP score is then computed as the cosine similarity score between the image and text pairs in CLIP space. We follow the same procedure to compute the CLIP score for our method, where we edit the text prompts to include adverb corresponding to our scale factor α, e.g.,'moderately' for $\alpha = 2$, and 'very' for $\alpha = 3$. The quantitative results in Tab. 1, show that our ACS produces high-quality avatars that semantically align with the desired target, without compromising quality.

Table 1: Quantitative comparison.

Settings	FID Score ↓	CLIP Score ↑
DreamHuman [24]	320	0.25
DreamWaltz [19]	257	0.33
Baseline [26]	269	0.31
Ours	**239**	**0.37**

User Study. We conduct a user study to show the superiority of our ACS for manipulating the degree of concepts. We invited 30 individuals from the general public to compare our method with the baseline [26]. Each participant was shown 16 sets of edited avatars, with each set containing results from both methods edited to various degrees of a concept. Participants answered three questions regarding editing quality, relevance and controllability: (1) Which image shows better editing quality? (2) Which image shows more relevant results to the editing target? (3) Which image better shows the concept changing from source to target? A copy of the survey is included in Supp. As shown in Fig. 3, users significantly favour our ACS over the baseline.

Table 2: Results of user study.

Setting	User Preference Rate (%)		
	Q1	Q2	Q3
Baseline [26]	26.04	15.62	16.67
Ours	73.96	84.38	83.33

6 Ablation Studies

Impact of Attribute Preserving Loss. In Sec. 4.2, we proposed the attribute preserving loss to better retain identity information during editing. We visualize the effect of our attribute preserving loss in Fig. 4, where we observe that it helps to better retain identify information (e.g., clothing, hair).

Analysis of Concept-Sensitive Primitive Selection. In our ACS, we edit the 3D avatar in 3DGS representation for fast editing feedback, and introduce the concept sensitivity-aware selection mechanism to edit only a small fraction of primitives based on the Concept Sensitivity score. Fig. 5 visualizes the distribution of the Concept Sensitivity score over all primitives for two concepts. We observe that most primitives (over 95%) have small sensitivity values (i.e., close to zero), indicating that they do not contribute to our target concept. Therefore, we set the fractional threshold γ to 5%, significantly improving efficiency while maintaining editing quality. Moreover, Tab. 3 compares the editing speed of our method against a NeRF-based representation [47] and 3DGS baseline [26], using the same diffusion prior. Results show that our proposed method is much more efficient than the baselines.

7 Conclusion

In this work, we introduce our novel ACS for 3D avatar editing, enabling users to edit their 3D avatar precisely towards the desired degree of expression of a given concept. Our ACS includes a Concept Sliding Loss based on LDA and Attribute Preserving Loss based on PCA, while leveraging 3DGS representation with our concept-sensitivity selection mechanism for high editing efficiency. Results show that ACS enables fine-grained and high-quality concept-specific avatar editing.

References

[1] Md Shahinur Alam, Jason Lamberton, Jianye Wang, Carly Leannah, Sarah Miller, Joseph Palagano, Myles de Bastion, Heather L. Smith, Melissa Malzkuhn, and Lorna C. Quandt. Asl champ!: A virtual reality game with deep-learning driven sign recognition, 2023.

[2] Shivangi Aneja, Justus Thies, Angela Dai, and Matthias Nießner. Clipface: Text-guided editing of textured 3d morphable models. In ACM SIGGRAPH 2023 Conference Proceedings, SIGGRAPH 2023, Los Angeles, CA, USA, August 6-10, 2023, pages 70:1–70:11. ACM, 2023.

[3] Tim Brooks, Aleksander Holynski, and Alexei A. Efros. Instructpix2pix: Learning to follow image editing instructions. In IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023, pages 18392–18402. IEEE, 2023.

[4] Yukang Cao, Yan-Pei Cao, Kai Han, Ying Shan, and Kwan-Yee K. Wong. Dreamavatar: Text-and-shape guided 3d human avatar generation via diffusion models. CoRR, abs/2304.00916, 2023.

[5] Yiwen Chen, Zilong Chen, Chi Zhang, Feng Wang, Xiaofeng Yang, Yikai Wang, Zhongang Cai, Lei Yang, Huaping Liu, and Guosheng Lin. Gaussianeditor: Swift and controllable 3d editing with gaussian splatting. CoRR, abs/2311.14521, 2023.

[6] Zilong Chen, Feng Wang, and Huaping Liu. Text-to-3d using gaussian splatting. CoRR, abs/2309.16585, 2023.

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

[8] Jianglin Fu, Shikai Li, Yuming Jiang, Kwan-Yee Lin, Chen Qian, Chen Change Loy, Wayne Wu, and Ziwei Liu. Stylegan-human: A data-centric odyssey of human generation. In European Conference on Computer Vision, pages 1–19. Springer, 2022.

[9] Rohit Gandikota, Joanna Materzynska, Tingrui Zhou, Antonio Torralba, and David Bau. Concept sliders: Lora adaptors for precise control in diffusion models, 2023.

[10] Yuan-Chen Guo, Ying-Tian Liu, Ruizhi Shao, Christian Laforte, Vikram Voleti, Guan Luo, Chia-Hao Chen, Zi-Xin Zou, Chen Wang, Yan-Pei Cao, and Song-Hai Zhang. threestudio: A unified framework for 3d content generation. `https://github.com/threestudio-project/threestudio`, 2023.

[11] Xiao Han, Yukang Cao, Kai Han, Xiatian Zhu, Jiankang Deng, Yi-Zhe Song, Tao Xiang, and Kwan-Yee K. Wong. Headsculpt: Crafting 3d head avatars with text. CoRR, abs/2306.03038, 2023.

[12] Ayaan Haque, Matthew Tancik, Alexei A. Efros, Aleksander Holynski, and Angjoo Kanazawa. Instruct-nerf2nerf: Editing 3d scenes with instructions. CoRR, abs/2303.12789, 2023.

[13] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. The elements of statistical learning: data mining, inference, and prediction, volume 2. Springer, 2009.

[14] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. Advances in neural information processing systems, 30, 2017.

[15] Fangzhou Hong, Zhaoxi Chen, Yushi Lan, Liang Pan, and Ziwei Liu. Eva3d: Compositional 3d human generation from 2d image collections, 2022.

[16] Fangzhou Hong, Mingyuan Zhang, Liang Pan, Zhongang Cai, Lei Yang, and Ziwei Liu. Avatarclip: zero-shot text-driven generation and animation of 3d avatars. ACM Trans. Graph., 41(4):161:1–161:19, 2022.

[17] 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. In The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022. OpenReview.net, 2022.

[18] Huajian Huang, Longwei Li, Hui Cheng, and Sai-Kit Yeung. Photo-slam: Real-time simultaneous localization and photorealistic mapping for monocular, stereo, and RGB-D cameras. CoRR, abs/2311.16728, 2023.

[19] Yukun Huang, Jianan Wang, Ailing Zeng, He Cao, Xianbiao Qi, Yukai Shi, Zheng-Jun Zha, and Lei Zhang. Dreamwaltz: Make a scene with complex 3d animatable avatars. CoRR, abs/2305.12529, 2023.

[20] Ruixiang Jiang, Can Wang, Jingbo Zhang, Menglei Chai, Mingming He, Dongdong Chen, and Jing Liao. Avatarcraft: Transforming text into neural human avatars with parameterized shape and pose control. CoRR, abs/2303.17606, 2023.

[21] Kai Katsumata, Duc Minh Vo, and Hideki Nakayama. An efficient 3d gaussian representation for monocular/multi-view dynamic scenes, 2023.

[22] Nikhil Varma Keetha, Jay Karhade, Krishna Murthy Jatavallabhula, Gengshan Yang, Sebastian A. Scherer, Deva Ramanan, and Jonathon Luiten. Splatam: Splat, track & map 3d gaussians for dense RGB-D SLAM. CoRR, abs/2312.02126, 2023.

[23] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. ACM Trans. Graph., 42(4):139:1–139:14, 2023.

[24] Nikos Kolotouros, Thiemo Alldieck, Andrei Zanfir, Eduard Gabriel Bazavan, Mihai Fieraru, and Cristian Sminchisescu. Dreamhuman: Animatable 3d avatars from text. CoRR, abs/2306.09329, 2023.

[25] 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/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023, pages 300–309. IEEE, 2023.

[26] Xian Liu, Xiaohang Zhan, Jiaxiang Tang, Ying Shan, Gang Zeng, Dahua Lin, Xihui Liu, and Ziwei Liu. Humangaussian: Text-driven 3d human generation with gaussian splatting. CoRR, abs/2311.17061, 2023.

[27] Timo Lüddecke and Alexander S. Ecker. Image segmentation using text and image prompts. In IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022, pages 7076–7086. IEEE, 2022.

[28] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. CoRR, abs/2308.09713, 2023.

[29] Geoffrey J McLachlan. Discriminant analysis and statistical pattern recognition. John Wiley & Sons, 2005.

[30] Mohit Mendiratta, Xingang Pan, Mohamed Elgharib, Kartik Teotia, Mallikarjun B. R., Ayush Tewari, Vladislav Golyanik, Adam Kortylewski, and Christian Theobalt. Avatarstudio: Text-driven editing of 3d dynamic human head avatars. ACM Trans. Graph., 42(6):226:1–226:18, 2023.

[31] 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 Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part I, volume 12346 of Lecture Notes in Computer Science, pages 405–421. Springer, 2020.

[32] Arthur Moreau, Jifei Song, Helisa Dhamo, Richard Shaw, Yiren Zhou, and Eduardo Pérez-Pellitero. Human gaussian splatting: Real-time rendering of animatable avatars, 2023.

[33] Simon Niedermayr, Josef Stumpfegger, and Rüdiger Westermann. Compressed 3d gaussian splatting for accelerated novel view synthesis, 2023.

[34] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. In The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023. OpenReview.net, 2023.

[35] 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. In Marina Meila and Tong Zhang, editors, Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event, volume 139 of Proceedings of Machine Learning Research, pages 8748–8763. PMLR, 2021.

[36] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022, pages 10674–10685. IEEE, 2022.

[37] Jiaxiang Tang, Jiawei Ren, Hang Zhou, Ziwei Liu, and Gang Zeng. Dreamgaussian: Generative gaussian splatting for efficient 3d content creation. CoRR, abs/2309.16653, 2023.

[38] Kartik Teotia, Mallikarjun B. R., Xingang Pan, Hyeongwoo Kim, Pablo Garrido, Mohamed Elgharib, and Christian Theobalt. Hq3davatar: High quality controllable 3d head avatar. CoRR, abs/2303.14471, 2023.

[39] Qian Wan and Zhicong Lu. Investigating vtubing as a reconstruction of streamer self-presentation: Identity, performance, and gender, 2023.

[40] 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. CoRR, abs/2305.16213, 2023.

[41] Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4d gaussian splatting for real-time dynamic scene rendering. CoRR, abs/2310.08528, 2023.

[42] Yichao Yan, Zanwei Zhou, Zi Wang, Jingnan Gao, and Xiaokang Yang. Dialoguenerf: Towards realistic avatar face-to-face conversation video generation, 2023.

[43] Zhuoqian Yang, Shikai Li, Wayne Wu, and Bo Dai. 3dhumangan: 3d-aware human image generation with 3d pose mapping, 2023.

[44] Taoran Yi, Jiemin Fang, Junjie Wang, Guanjun Wu, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Qi Tian, and Xinggang Wang. Gaussiandreamer: Fast generation from text to 3d gaussians by bridging 2d and 3d diffusion models, 2023.

[45] Taoran Yi, Jiemin Fang, Guanjun Wu, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Qi Tian, and Xinggang Wang. Gaussiandreamer: Fast generation from text to 3d gaussian splatting with point cloud priors. CoRR, abs/2310.08529, 2023.

[46] Hao Zhang, Yao Feng, Peter Kulits, Yandong Wen, Justus Thies, and Michael J. Black. Text-guided generation and editing of compositional 3d avatars. CoRR, abs/2309.07125, 2023.

[47] Huichao Zhang, Bowen Chen, Hao Yang, Liao Qu, Xu Wang, Li Chen, Chao Long, Feida Zhu, Kang Du, and Min Zheng. Avatarverse: High-quality & stable 3d avatar creation from text and pose. CoRR, abs/2308.03610, 2023.

[48] Jianfeng Zhang, Xuanmeng Zhang, Huichao Zhang, Jun Hao Liew, Chenxu Zhang, Yi Yang, and Jiashi Feng. Avatarstudio: High-fidelity and animatable 3d avatar creation from text. CoRR, abs/2311.17917, 2023.

[49] Lvmin Zhang and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. CoRR, abs/2302.05543, 2023.

[50] Jingyu Zhuang, Chen Wang, Liang Lin, Lingjie Liu, and Guanbin Li. Dreameditor: Text-driven 3d scene editing with neural fields. In June Kim, Ming C. Lin, and Bernd Bickel, editors, SIGGRAPH Asia 2023 Conference Papers, SA 2023, Sydney, NSW, Australia, December 12-15, 2023, pages 26:1–26:10. ACM, 2023.

[51] Wojciech Zielonka, Timur M. Bagautdinov, Shunsuke Saito, Michael Zollhöfer, Justus Thies, and Javier Romero. Drivable 3d gaussian avatars. CoRR, abs/2311.08581, 2023.

8 Supplemental Material

8.1 More Details about LDA

In Sec. 4.1 of the main paper, we present how we leverage Linear Discriminant Analysis (LDA) to identify a concept axis to link the opposite sides of a concept. Here, we discuss more regarding the details of LDA.

First, we provide more details regarding the within-class scatter ($\mathbf{S}_w$) and the between-class scatter ($\mathbf{S}_b$) which are used in Eq. 2. As discussed in the main paper, the within-class scatter ($\mathbf{S}_w$) quantifies how much individual feature vectors deviate from their corresponding set centroids, while the between-class scatter ($\mathbf{S}_b$) measures the deviation between the centroids of both sides of the concept.

More precisely, given the two sets of features $\{\mathbf{F}_{p,i,x,y}\}_{i\in[1,N_s],x\in[1,H],y\in[1,W]}$ and $\{\mathbf{F}_{n,i,x,y}\}_{i\in[1,N_s],x\in[1,H],y\in[1,W]}$, the within-class scatter $\mathbf{S}_w$ is calculated as the summation of the variance for each set as:

$$
\begin{aligned}
\mathbf{S}_w = & \sum_{i=1}^{N_s}\sum_{x=1}^{H}\sum_{y=1}^{W}(\mathbf{F}_{p,i,x,y} - \boldsymbol{\mu}_p)(\mathbf{F}_{p,i,x,y} - \boldsymbol{\mu}_p)^T \\
& + \sum_{i=1}^{N_s}\sum_{x=1}^{H}\sum_{y=1}^{W}(\mathbf{F}_{n,i,x,y} - \boldsymbol{\mu}_n)(\mathbf{F}_{n,i,x,y} - \boldsymbol{\mu}_n)^T,
\end{aligned}
\tag{7}
$$

where $\boldsymbol{\mu}_p, \boldsymbol{\mu}_n \in \mathbb{R}^D$ are computed mean vectors for each concept side, i.e., $\boldsymbol{\mu}_p = \frac{1}{N_s HW}\sum_{i=1}^{N_s}\sum_{x=1}^{H}\sum_{y=1}^{W}\mathbf{F}_{p,i,x,y}$ and $\boldsymbol{\mu}_n = \frac{1}{N_s HW}\sum_{i=1}^{N_s}\sum_{x=1}^{H}\sum_{y=1}^{W}\mathbf{F}_{n,i,x,y}$. At the same time, we can calculate the between-class scatter S_b as:

$$
\mathbf{S}_b = (\boldsymbol{\mu}_p - \boldsymbol{\mu}_n)^T(\boldsymbol{\mu}_p - \boldsymbol{\mu}_n) \text{where } \bar{\mu} = (\boldsymbol{\mu}_p + \boldsymbol{\mu}_n)/2
\tag{8}
$$

Then, based on the computed $\mathbf{S}_w$ and $\mathbf{S}_b$, we seek to identify the concept basis $\mathbf{b}_c$, such that the projection of each feature vector onto the concept axis $\mathbf{b}_c$ leads to maximal ratio of $\mathbf{S}_b$ to $\mathbf{S}_w$. Therefore, as mentioned in Eq. 2 of main paper, this process can be expressed as the following optimization problem:

$$
\mathbf{b}_c = \arg\max_{\mathbf{w}} \frac{\mathbf{w}\mathbf{S}_b\mathbf{w}^T}{\mathbf{w}\mathbf{S}_w\mathbf{w}^T}, \text{s.t. } ||\mathbf{w}|| = 1,
\tag{9}
$$

Notably, it can be shown that the solution to Eq. 9 can be conveniently computed as the leading eigenvector of $\mathbf{S}_w^{-1}\mathbf{S}_b$, which we provide the derivation below.

First of all, we define $\mathcal{J}(\mathbf{w}) = \frac{\mathbf{w}\mathbf{S}_b\mathbf{w}^T}{\mathbf{w}\mathbf{S}_w\mathbf{w}^T}$. Then, we take the derivative of $\mathcal{J}(\mathbf{w})$ with respect to $\mathbf{w}$ and set it to zero, so that we can have:

$$
\begin{aligned}
\frac{d\mathcal{J}(\mathbf{w})}{d\mathbf{w}} &= \frac{(\frac{d}{d\mathbf{w}}\mathbf{w}\mathbf{S}_b\mathbf{w}^T)\mathbf{w}\mathbf{S}_w\mathbf{w}^T - (\frac{d}{d\mathbf{w}}\mathbf{w}\mathbf{S}_w\mathbf{w}^T)\mathbf{w}\mathbf{S}_b\mathbf{w}^T}{(\mathbf{w}\mathbf{S}_w\mathbf{w}^T)^2} \\
&= \frac{(2\mathbf{S}_b\mathbf{w}^T)\mathbf{w}\mathbf{S}_w\mathbf{w}^T - (2\mathbf{S}_w\mathbf{w}^T)\mathbf{w}\mathbf{S}_b\mathbf{w}^T}{(\mathbf{w}\mathbf{S}_w\mathbf{w}^T)^2} = 0
\end{aligned}
\tag{10}
$$

Then, we simplify it further as follows:

$$
\begin{aligned}
(2\mathbf{S}_b\mathbf{w}^T)\mathbf{w}\mathbf{S}_w\mathbf{w}^T - (2\mathbf{S}_w\mathbf{w}^T)\mathbf{w}\mathbf{S}_b\mathbf{w}^T &= 0 \\
\mathbf{S}_b\mathbf{w}^T - \frac{\mathbf{w}\mathbf{S}_b\mathbf{w}^T(\mathbf{S}_w\mathbf{w}^T)}{\mathbf{w}\mathbf{S}_w\mathbf{w}^T} &= 0 \\
\mathbf{S}_w^{-1}\mathbf{S}_b\mathbf{w}^T &= \mathcal{J}\mathbf{w}^T
\end{aligned}
\tag{11}
$$

Therefore, by applying the definition of eigenvectors, we find that the stationary points of $\mathcal{J}(\mathbf{w})$ only occur when $\mathbf{w}$ is an eigenvector of $\mathbf{S}_w^{-1}\mathbf{S}_b$. In other words, over all the stationary points, $\mathcal{J}(\mathbf{w})$ takes values corresponding to all the eigenvalues of $\mathbf{S}_w^{-1}\mathbf{S}_b$. As a result, the maximum value of $\mathcal{J}(\mathbf{w})$ is equivalent to the largest eigenvector, and happens when $\mathbf{w}$ is the leading eigenvector of $\mathbf{S}_w^{-1}\mathbf{S}_b$. Hence, this shows that Eq. 9 can be solved by simply setting $\mathbf{w}$ to be the leading eigenvector of $\mathbf{S}_w^{-1}\mathbf{S}_b$.

Algorithm 1 Training of Slider Adapter

Require: $c_p, c_n, c_{ne}, \boldsymbol{\epsilon}_{\boldsymbol{\phi_0}}, \boldsymbol{\epsilon}_{\hat{\boldsymbol{\phi}}}, T_{DDIM}, N_s, S$

1: $t = 1; i = 0$
2: **while** $t <= T_{DDIM}$ **do**   ▷ Bases Analysis
3:   $\{\mathbf{F}_{p,i,x,y,t}\}_{i=1,x=1,y=1}^{N_s,H,W}, \boldsymbol{\mu}_{p,t} \leftarrow Sampling(\boldsymbol{\epsilon}_{\boldsymbol{\phi_0}}, c_p, t)$
4:   $\{\mathbf{F}_{n,i,x,y,t}\}_{i=1,x=1,y=1}^{N_s,H,W}, \boldsymbol{\mu}_{n,t} \leftarrow Sampling(\boldsymbol{\epsilon}_{\boldsymbol{\phi_0}}, c_n, t)$
5:   $\mathbf{b}_{c,t} \leftarrow LDA(\{\mathbf{F}_{p,i,x,y}\}_{i=1,x=1,y=1}^{N_s,H,W}, \{\mathbf{F}_{n,i,x,y}\}_{i=1,x=1,y=1}^{N_s,H,W})$
6:   $\{\mathbf{b}_{a,k,t}\}_{k=1}^{K} \leftarrow PCA(\{\mathbf{F}_{p,i,x,y}\}_{i=1,x=1,y=1}^{N_s,H,W}, \{\mathbf{F}_{n,i,x,y}\}_{i=1,x=1,y=1}^{N_s,H,W})$
7:   $t = t + 1$
8: **end while**
9: **while** $i < S$ **do**   ▷ LoRA Finetuning
10:   $t = 1; Z_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
11:   $T_2 \sim Uniform(\{1, ..., T_{DDIM}\})$
12:   $\alpha \sim Uniform([-1, 1])$
13:   **while** $t < T_2$ **do**
14:    $Z_{T-(t+1)} = \text{StopGrad}(\text{denoising}(\boldsymbol{\epsilon}_{\boldsymbol{\phi_0}}(Z_{T-t}, T - t, c_{ne})))$
15:    $t = t + 1$
16:   **end while**
17:   $f(\Delta\phi, \alpha, c_{ne}, T_2) \leftarrow \boldsymbol{\epsilon}_{\hat{\boldsymbol{\phi}}}(Z_{T-T_2}, T - T_2, c_{ne}, \alpha, \Delta\phi)$
18:   $f(\varnothing, c_{ne}, T_2) \leftarrow \boldsymbol{\epsilon}_{\boldsymbol{\phi_0}}(Z_{T-T_2}, T - T_2, c_{ne})$ ▷ Extract feature from middle layer.
19:   $\mathcal{L}_1 = \mathcal{L}_{sliding}(f(\Delta\phi, \alpha, c_{ne}, T_2), \boldsymbol{\mu}_p, \boldsymbol{\mu}_n, \alpha, \mathbf{b}_{c,T_2})$
20:   $\mathcal{L}_2 = \mathcal{L}_{preserving}(f(\Delta\phi, \alpha, c_{ne}, T_2), f(\varnothing, c_{ne}, T_2), \alpha, \{\mathbf{b}_{a,k,T_2}\}_{k=1}^{K})$
21:   $(\mathcal{L}_1 + \mathcal{L}_2).backward()$; Update $\Delta\phi$
22:   $i = i + 1$
23: **end while**

8.2 More Implementation Details

8.2.1 Fine-tuning of the Adapter

At the fine-tuning stage, when given a concept via a pair of opposing text prompts c_p, c_n, we start by randomly generating $N_s = 20$ images for each concept using LoRA-unwrapped diffusion model $\boldsymbol{\epsilon}_{\phi_0}$ based on DDIM scheduler of $T_{DDIM} = 50$ time steps. We keep the intermediate features from the middle layer at each time step t as $\{\mathbf{F}_{p,i,x,y,t}\}_{i=1,x=1,y=1}^{N_s,H,W}$ and $\{\mathbf{F}_{n,i,x,y,t}\}_{i=1,x=1,y=1}^{N_s,H,W}$, which are subsequently processed with LDA and iterative PCA analysis procedure to obtain the time-dependent concept axis $\mathbf{b}_{c,t}$ and attribute bases $\{\mathbf{b}_{a,k,t}\}_{k=1}^{K}$. Meanwhile, we compute the mean vectors $\boldsymbol{\mu}_{n,t}$ and $\boldsymbol{\mu}_{p,t}$ for each set. Next, we follow similar process as [9] to fine-tuning our adapter. We sample a pair of intermediate features conditioned on neutral concept prompt c_{ne} as $f(\Delta\phi, \alpha, c_{ne}, T_2)$ and $f(\varnothing, c_{ne}, T_2)$ from LoRA-wrapped $\boldsymbol{\epsilon}_{\hat{\phi}}(\Delta\phi, \alpha)$ and unwrapped diffusion model $\boldsymbol{\epsilon}_{\phi_0}(\varnothing)$ with T_2 denoising process respectively. The T_2 and α are uniformly sampled from $\{1, ..., T_{DDIM}\}$ and $[-1, 1]$ at beginning of each train step. It is worth noting that, despite fine-tuning with alpha from [-1, 1], at editing stage, we can set slider factor going beyond the range as shown in Fig. 3. We freeze the parameters of original diffusion model and train the parameters of adapter for $S = 1000$ steps based on the $\mathcal{L}_{sliding}$ and $\mathcal{L}_{preserving}$ loss criterions. The detailed fine-tuning process is illustrated as pseudocode in Alg. 1.

8.2.2 Editing Process

At the editing stage, we take the original source avatar as input and follow the SDS optimization pipeline introduced in Sec. 3 to optimize the 3D GS parameters for 1200 steps. Because the generated avatar is fully densified and consists of large number of primitives, to prevent the optimization from consuming too much memory because of further densification, we start the editing without densification procedure in the first 600 optimization steps and only allow prune-only [32] procedure at every 200 steps. In the remaining 600 steps, we resume the densification process every 200 steps to refine the Avatar. Notably, to adapt the [26] for editing an avatar, akin to our pipeline, we take the source avatar as the initial point of SDS-based editing and start the prune-only stage first, to avoid the over densification.

Figure 6: Comparison with the text embedding interpolation baseline. Our method is significantly superior in terms of precisely controlling the editing process.

8.3 More Ablation Studies

8.3.1 Comparison with Simple Text Embedding Interpolation

In Sec. 4.1, we propose to leverage LDA mechanism to discover the continuous axis allowing us to precisely specify the intermediate concept. A straightforward and intuitive solution to find such an axis is to linearly interpolate the two sides of textual prompts in embedding space. So, here, we compare our LDA-based method with the straightforward solution at higher scale precision here. The results are illustrated in Fig. 6, which indicates that our LDA-based solution allows us precisely edit avatar to an intermediate degree of concept with smooth interpolation between different scales. These results show the superiority of our method in terms of precise control of the editing process.

8.3.2 Ablation of Primitive Selection

In Sec. 4.3, we introduce a concept-sensitivity-aware primitive selection mechanism to reduce redundancy and enhance editing efficiency by selecting a small fraction of primitives. We visualize the edited avatar with and without this selection mechanism at their median and final editing steps. Notably, our editing process with the selection mechanism converges in about 1200 steps, whereas the method without it takes approximately 3000 steps. The results, illustrated in Fig. 7, demonstrate that our selection mechanism accelerates the convergence process significantly, without negatively affecting the final quality.

8.3.3 Details about User Study

We conduct a user study to qualitatively evaluate our method against the baseline. Each respondent is given a questionnaire with 16 randomly sampled editing results. Respondents are asked three questions regarding the quality, editing relevance and editing controllability as noted in 6 for each editing example and they can vote for better their preferred results. One example questionnaire is shown in Fig. 8 and Fig. 9. Note that in the real questionnaire, the results of baseline and our method are shuffled for fair comparison. However, for illustration purposes, the first row in each cell in Fig. 8 and Fig. 9 shows the baseline results while the second row shows our ACS results.
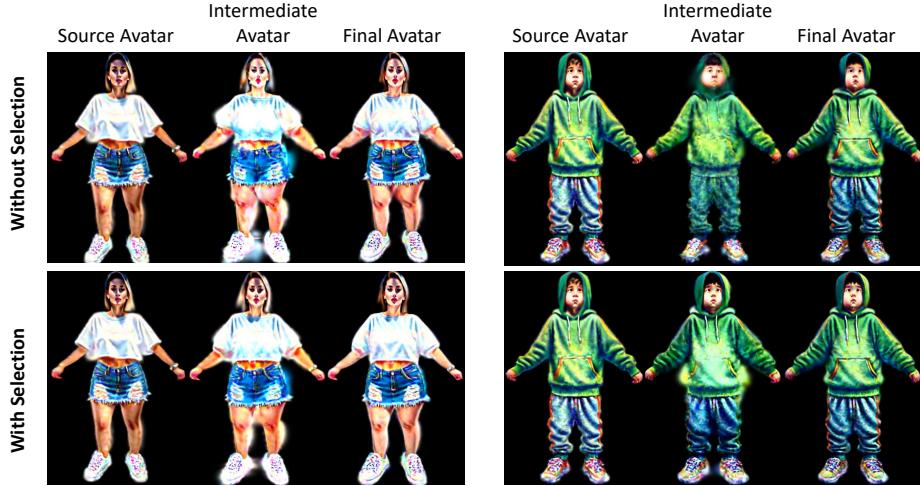
15

Figure 7: Ablation of Primitive Selection. By using our Primitive Selection mechanism, we can increase efficiency significantly, without compromising on quality.

8.4 Limitations and Impacts

8.4.1 Limitations and Future Works

Our ACS utilizes the text-image diffusion model as a 2D prior for avatar editing. However, this model may not comprehend abstract or uncommon terms like 'Star Wars' or 'Naruto' leading to difficulties in learning the correct adapter. Additionally, the quality of generation and editing can vary based on the quality of the diffusion prior. Further research may focus on enhancing the model's ability to understand prompts when learning the adapter and explore various types of diffusion priors to improve quality and diversity.

8.4.2 Potential Impacts

Our ACS holds significant potential for positive impact, enabling more personalized and diverse avatar creation in gaming, virtual reality, and digital art. Users can create avatars that better reflect their identities or creative visions. However, there are also potential negative impacts, such as reinforcing harmful stereotypes or biases, and the ethical concerns of manipulating realistic human likenesses. It is crucial to approach these developments with sensitivity and a focus on inclusive, responsible use.
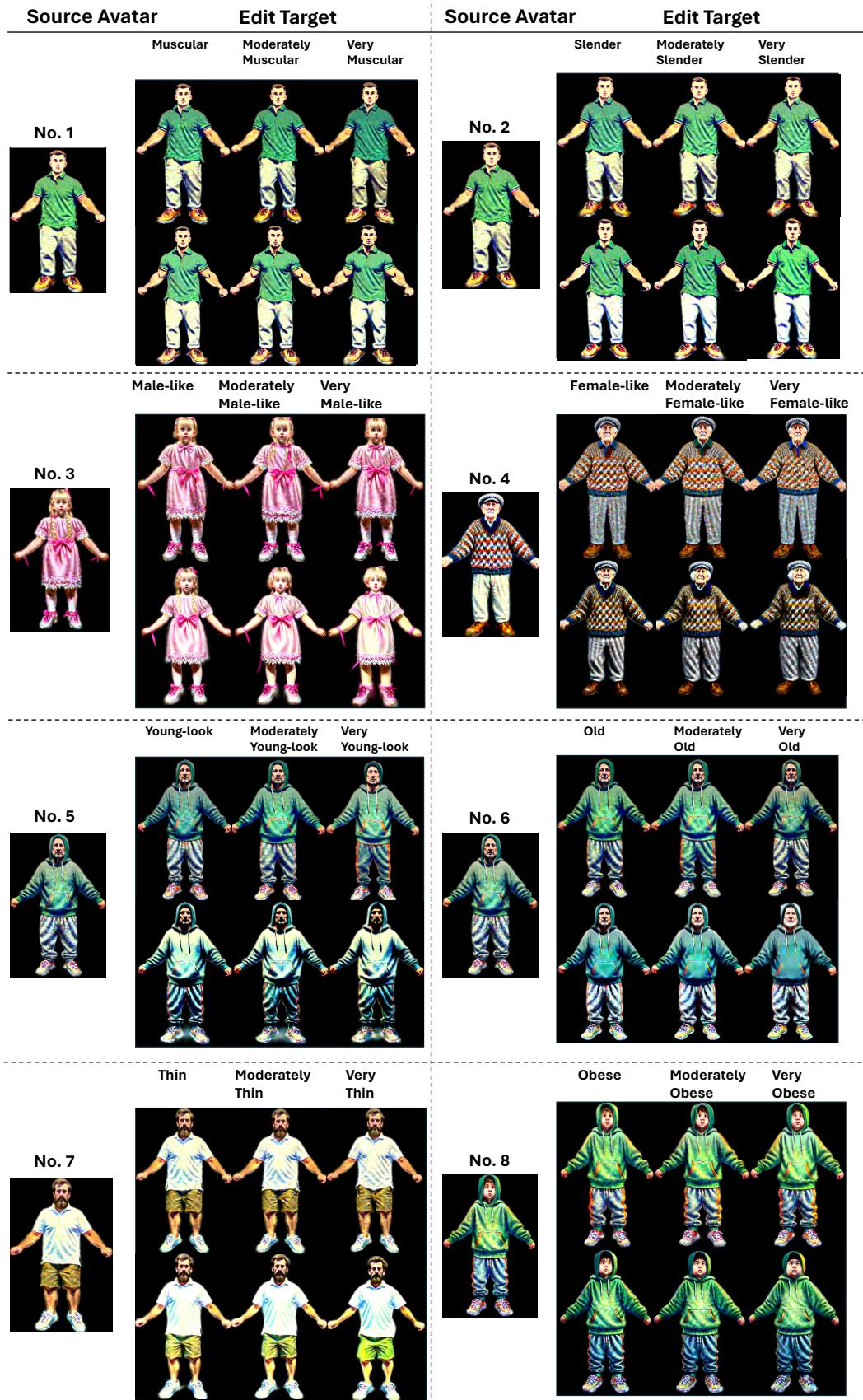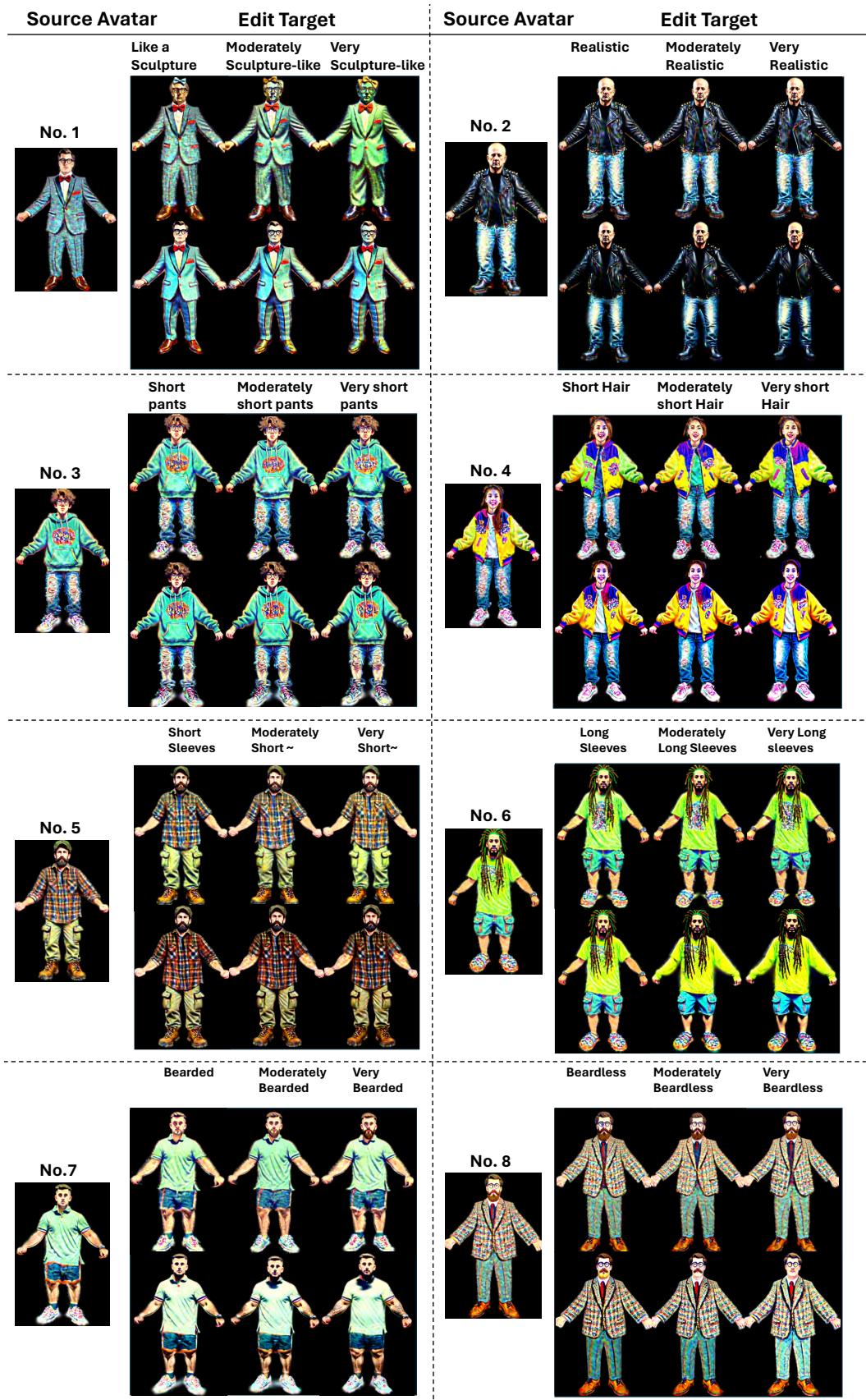
Figure 8: User Study Examples

Figure 9: User Study Examples