3D-GOI: 3D GAN Omni-Inversion for Multifaceted and Multi-object Editing

Haoran Li^{1,2}[©], Long Ma^{1,2}[©], Haolin Shi^{1,2}[©], Yanbin Hao^{1,2}[©], Yong Liao^{1,2*} [©], Lechao Cheng³[©], and Peng Yuan Zhou^{4*}[©]

¹ University of Science and Technology of China
² CCCD Key Lab of Ministry of Culture and Tourism
{lhr123, longm, mar}@mail.ustc.edu.cn, haoyanbin@hotmail.com, yliao@ustc.edu.cn
³ Hefei University of Technology chenglc@hfut.edu.cn
⁴ Aarhus University pengyuan.zhou@ece.au.dk

Abstract. The current GAN inversion methods typically can only edit the appearance and shape of a single object and background while overlooking spatial information. In this work, we propose a 3D editing framework, 3D-GOI to enable multifaceted editing of affine information (scale, translation, and rotation) on multiple objects. 3D-GOI realizes the complex editing function by inverting the abundance of attribute codes (object shape/ appearance/ scale/ rotation/ translation, background shape/ appearance, and camera pose) controlled by GIRAFFE, a renowned 3D GAN. Accurately inverting all the codes is challenging, 3D-GOI solves this challenge following three main steps. First, we segment the objects and the background in a multi-object image. Second, we use a custom Neural Inversion Encoder to obtain coarse codes of each object. Finally, we use a round-robin optimization algorithm to get precise codes to reconstruct the image. To the best of our knowledge, 3D-GOI is the first framework to enable multifaceted editing on multiple objects. Both qualitative and quantitative experiments demonstrate that 3D-GOI holds immense potential for flexible, multifaceted editing in complex multi-object scenes. Our project and code are released at https://3d-goi.github.io.

1 Introduction

The development of generative 3D models has attracted increasing attention to automatic 3D objects and scene generation and edition. Most existing works are limited to a single object, such as 3D face generation [7] and synthesis of facial viewpoints [41]. There are few methods for generating multi-object 3D scenes while editing such scenes remains unexplored. In this paper, we propose 3D-GOI to edit images containing multiple objects with complex spatial geometric

^{*} Corresponding authors



Fig. 1: The first row shows the editing results of traditional 2D/3D GAN inversion methods on multi-object images. The second row showcases 3D-GOI, which can perform multifaceted editing on complex images with multiple objects. 'bg' stands for background. The red crosses in the upper right figures indicate features that cannot be edited with current 2D/3D GAN inversion methods.

relationships. 3D-GOI not only can change the appearance and shape of each object and the background, but also can edit the spatial position of each object and the camera pose of the image as shown by Figure 1.

Existing 3D multi-object scene generation methods can be mainly classified into two categories: those based on Generative Adversarial Networks (GANs) [10] and those [22] based on diffusion models [13], besides a few based on VAE or Transformer [3,39]. GAN-based methods, primarily represented by GIRAFFE [28] and its derivatives, depict complex scene images as results of multiple foreground objects, controlled by shape and appearance, subjected to affine transformations (scaling, translation, and rotation), and rendered together with a background, which is also controlled by shape and appearance, from a specific camera viewpoint. Diffusion-based methods [23] perceive scene images as results of multiple latent NeRF [24], which can be represented as 3D models, undergoing affine transformations, optimized with SDS [30], rendered from a specific camera viewpoint. Both categories represent scenes as combinations of multiple codes. To realize editing based on these generative methods, it's imperative to invert the complex multi-object scene images to retrieve their representative codes. After modifying these codes, regeneration can achieve diversified editing of complex images. Most inversion methods study the inversion of a single code based on its generation method. However, each multi-object image is the entangled result of multiple codes, thus inverting all codes from an image requires precise disentangling of the codes, which is extremely difficult and largely overlooked. Moreover, the prevailing inversion algorithms primarily employ optimization approaches. Attempting to optimize all codes simultaneously often leads to chaotic optimization directions and less accurate inversion outcomes.

Therefore, we propose 3D-GOI, a framework capable of inverting multiple codes to achieve a comprehensive inversion of multi-object images. Given current open-source 3D multi-object scene generation methods, we have chosen GI-RAFFE [28] as our generative model. In theory, our framework can be applied to other generative approaches as well. We address these challenges as follows.

First, we categorize different codes based on object attributes, background attributes, and pose attributes. Through qualitative verification, we found that segmentation methods can roughly separate the codes pertaining to different objects. For example, the codes controlling an object's shape, appearance, scale, translation, and rotation predominantly relate to the object itself. So, during the inversion process, we only use the segmented image of this object to reduce the impact of the background and other objects on its codes.

Second, we get the attributes' codes from the segmented image. Inspired by the Neural Rendering Block in GIRAFFE, we design a custom Neural Inversion Encoder network to coarsely disentangle and estimate the code values.

Finally, we obtain precise values for each code through optimization. We observed that optimizing all codes simultaneously tends to get stuck in local minima. Therefore, we propose a round-robin optimization algorithm that employs a ranking function to determine the optimization order for different codes. The algorithm enables a stable and efficient optimization process for accurate image reconstruction. Our contributions can be summarized as follows.

- To our best knowledge, 3D-GOI is the first multi-code inversion framework in generative models, achieving multifaceted editing of multi-object images.
- We introduce a three-stage inversion process: 1) separate the attribute codes of different objects via segmentation; 2) obtain coarse codes using a custom Neural Inversion Encoder; 3) optimize the reconstruction using a round-robin optimization strategy.
- Our method outperforms existing methods on both 3D and 2D tasks.

2 Related Work

2D/3D GANs. 2D GAN maps a distribution from the latent space to the image space using a generator and a discriminator and has been widely explored. For example, BigGAN [6] increases the batch size and uses a simple truncation trick to finely control the trade-off between sample fidelity and variety. Cycle-GAN [45] feeds an input image into the generator and loops the output back to the generator. It achieves style transfer by minimizing the consistency loss between the input and its result. StyleGAN [17] maps a latent code into multiple style codes, allowing for detailed style control of images. 3D GANs usually combine 2D GANs with some 3D representation, such as NeRF [25], and have demonstrated excellent abilities to generate complex scenes with multi-view consistency. Broadly, 3D GANs can be classified into explicit and implicit models. Explicit models like HoloGAN [26] enable explicit control over the object pose through rigid body transformations of the learned 3D features. BlockGAN [27] generates foreground and background 3D features separately, combining them into a complete 3D scene representation. On the other hand, implicit models generally perform better. Many of these models take inspiration from NeRF [25], representing images as neural radiance fields and using volume rendering to generate photorealistic images in a continuous view. EG3D [7] introduces an explicitimplicit hybrid network architecture that produces high-quality 3D geometries.



Fig. 2: Different GANs and GAN Inversion methods utilize codes differently. ω represents the latent code and c represents the camera pose.

GRAF [34] integrates shape and appearance coding within the generation process, which facilitates independent manipulation of the shape and appearance of the generated vehicle and furniture images. Moreover, the presence of 3D information provides additional control over the camera pose, contributing to the flexibility of the generated outputs. GIRAFFE [28] extends GRAF to multiobject scenes by considering an image as the composition of multiple objects in the foreground through affine transformation and the background rendered at a specific camera viewpoint. In this work, we select GIRAFFE as the 3D GAN model to be inverted.

2D/3D GAN Inversion. GAN inversion obtains the latent code of an input image under a certain generator and modifies the latent code to perform image editing operations. Current 2D GAN inversion methods can be divided into optimization-based, encoder-based, and hybrid methods. Optimization-based methods [1, 14, 44] directly optimize the initial code, requiring very accurate initial values. Encoder-based methods [29,31,37] can map images directly to latent code but generally cannot achieve full reconstruction. Hybrid-based methods [4, 43] combine these two approaches: first employ an encoder to map the image to a suitable latent code, then perform optimization. Currently, most 2D GANs only have one latent code to generate an image ⁵. Therefore, the 2D GAN inversion task can be represented as:

$$\boldsymbol{\omega}^* = \arg\min_{\boldsymbol{\omega}} \mathcal{L}(G(\boldsymbol{\omega}, \boldsymbol{\theta}), I), \tag{1}$$

where $\boldsymbol{\omega}$ is the latent component, G denotes the generator, θ denotes the parameters of the generator, I is the input image, and \mathcal{L} is the loss function measuring the difference between the generated and input image.

Typically, 3D GANs have an additional camera pose parameter compared to 2D GANs, making it more challenging to obtain latent codes during inversion. Current methods like SPI [41] use a symmetric prior for faces to generate images with different perspectives, while [19] employs a pre-trained estimator to achieve better initialization and utilizes pixel-level depth calculated from the NeRF parameters for improved image reconstruction.

Currently, there are only limited works on 3D GAN inversion [9,21,38] which primarily focus on creating novel perspectives of human faces using specialized

⁵ Although StyleGAN can be controlled by multiple style codes, these codes are all generated from a single initial latent code, indicating their interrelations. Hence only one encoder is needed to predict all the codes during inversion.

face datasets considering generally only two codes: camera pose code and the latent code. Hence its inversion task can be represented as:

$$\boldsymbol{\omega}^*, \boldsymbol{c}^* = \arg\min_{\boldsymbol{\omega}, \boldsymbol{c}} \mathcal{L}(G(\boldsymbol{\omega}, \boldsymbol{c}, \boldsymbol{\theta}), \boldsymbol{I}).$$
(2)

A major advancement of 3D-GOI is the capability to invert more independent codes compared with other inversion methods, as Figure 2 shows, in order to perform multifaceted edits on multi-object images.

3 Preliminary

GIRAFFE [28] represents individual objects as a combination of feature field and volume density. Through scene compositions, the feature fields of multiple objects and the background are combined. Finally, the combined feature field is rendered into an image using volume rendering and neural rendering.

For a coordinate \boldsymbol{x} and a viewing direction \boldsymbol{d} in scene space, the affine transformation T(s, t, r) (scale, translation, rotation) is used to transform them back into the object space of each individual object. Following the implicit shape representations used in NeRF, a multi-layer perceptron (MLP) h_{θ} is used to map the transformed \boldsymbol{x} and \boldsymbol{d} , along with the shape-controlling code z_s and appearance-controlling code z_a , to the feature field \boldsymbol{f} and volume density σ :

$$(T(s,t,r;\boldsymbol{x})), T(s,t,r;\boldsymbol{d})), \boldsymbol{z_s}, \boldsymbol{z_a}) \xrightarrow{h_{\theta}} (\sigma, \boldsymbol{f}).$$

$$(3)$$

Then, GIRAFFE defines a Scene Composite Operator: at a given x and d, the overall density is the sum of the individual densities (including the background). The overall feature field is represented as the density-weighted average of the feature field of each object:

$$C(\boldsymbol{x}, \boldsymbol{d}) = (\sigma, \frac{1}{\sigma} \sum_{i=1}^{N} \sigma_i \boldsymbol{f}_i), where \quad \sigma = \sum_{i=1}^{N} \sigma_i,$$
(4)

where N denotes the background plus (N-1) objects.

The rendering phase is divided into two stages. Similar to volume rendering in NeRF, given a pixel point, the rendering formula is used to calculate the feature field of this pixel point from the feature fields and the volume density of all sample points in a camera ray direction. After calculating all pixel points, a feature map is obtained. Neural rendering (Upsampling) is then applied to get the rendered image. Please refer to the Supplementary Material 1 for the detailed preliminary and formulas.

4 3D-GOI

4.1 Problem Definition

The problem we target is similar to the general definition of GAN inversion, with the difference being that we need to invert many more codes than existing methods (1 or 2) shown in Figure 2. The parameter W in GIRAFFE, which controls

6 H. Li et al.



Fig. 3: The overall framework of 3D-GOI. As shown in the upper half, the encoders are trained on single-object scenes, each time using L_{enc} to predict one $w, w \in W$, while other codes use real values. The lower half depicts the inversion process for the multi-object scene. We first decompose objects and background from the scene, then use the trained encoder to extract coarse codes, and finally use the round-robin optimization algorithm to obtain precise codes. The green blocks indicate required training and the yellow blocks indicate fixed parameters.

the generation, can be divided into object attributes, background attributes, and pose attributes, denoted by O, B, and C. Then, W can be expressed as follows:

$$W = \{O_i^{shape}, O_i^{app}, O_i^s, O_i^t, O_i^r, B^{shape}, B^{app}, C\}, \quad i = 1, ..., n,$$
(5)

where O_i^{shape} is the object shape latent code, O_i^{app} is the object appearance latent code, O_i^s is the object scale code, O_i^t is the object translation code, O_i^r is the object rotation code, B^{shape} is the background shape latent code, B^{app} is the background appearance latent code and C is the camera pose matrix. ndenotes the n objects. The reconstruction part can be expressed as:

$$W^* = \arg\min_{W} \mathcal{L}(G(W, \theta), I).$$
(6)

According to Equation 5, we need to invert a total of (5n + 3) codes. Then, we are able to replace or interpolate any inverted code(s) to achieve multifaceted editing of multiple objects.

4.2 Scene Decomposition

As mentioned, the GIRAFFE generator differs from typical GAN generators in that a large number of codes are involved and not a single code controls all the generated parts. Therefore, it is challenging to transform all codes using just one encoder or optimizer as in typical GAN Inversion methods. While a human can easily distinguish each object and some of its features (appearance, shape), a machine algorithm requires a large number of high-precision annotated samples to understand what code is expressed at what position in the image.





Fig. 4: Scene decomposition. (a) The input image. (b) The feature weight map of car A, where the redder regions indicate a higher opacity and the bluer regions lower opacity. (c) The feature weight map of car B. (d) The feature weight map of the background. By integrating these maps, it becomes apparent that the region corresponding to car A predominantly consists of the feature representation of cars A and B. The background's visible area solely contains the background's feature representation.

A straightforward idea is that the attribute codes of an object will map to the corresponding position of the object in the image. For example, translation (O^t) and rotation (O^r) codes control the relative position of an object in the scene, scaling (O^s) and shape (O^{shape}) codes determine the contour and shape of the object, and appearance (O^{app}) codes control the appearance representation at the position of the object. The image obtained from segmentation precisely encompasses these three types of information, allowing us to invert it and obtain the five attribute codes for the corresponding object. Similarly, for codes (B^{shape}, B^{app}) that generate the background, we can invert them using the segmented image of the background. Note that obtaining C requires information from the entire rendered image.

We can qualitatively validate this idea. In Equation 3, we can see that an object's five attribute codes are mapped to the object's feature field and volume density through h_{θ} . As inferred from Equation 4, the scene's feature field is synthesized by weighting the feature fields of each object by density. Therefore, an object appears at its position because its feature field has a high-density weight at the corresponding location. Figure 4 displays the density of different objects at different positions during GIRAFFE's feature field composition process. The redder the higher the density, while the bluer the lower the density. As discussed, car A exhibits a high-density value within its area and near-zero density elsewhere - a similar pattern is seen with car B. The background, however, presents a non-uniform density distribution across the scene. we can consider that both car A and B and the background mainly manifest their feature fields within their visible areas. Hence, we apply a straightforward segmentation method to separate each object's feature field and get the codes. Segmenting each object also allows our encoder to pay more attention to each input object or background. As such, we can train the encoder on single-object scenes and then generalize it to multi-object scenes instead of directly training in multi-object scenes that involve more codes, to reduce computation cost.



Fig. 5: Neural Inversion Encoder. (a) The Neural Rendering Block in GIRAFFE [28], an upsampling process to generate image \hat{I} . (b) The Neural Inversion Encoder opposes (a), which is a downsampling process. I is the input image, H, W are image height and width. I_v is the heatmap of the image, H_v, W_v and M_f are the dimensions of I_v, w is the code to be predicted, and w_f is the dimension of w. Up/Down means upsampling/downsampling.

4.3 Coarse Estimation

The previous segmentation step roughly disentangles the codes. Unlike typical encoder-based methods, it's difficult to predict all codes using just one encoder. Therefore, we assign an encoder to each code, allowing each encoder to focus solely on predicting one code. Hence, we need a total of eight encoders. As shown in Figure 3, we input the object segmentation for the object attribute codes $(O^{shape}, O^{app}, O^s, O^t, O^r)$, the background segmentation for the background attribute codes (B^{shape}, B^{app}) , and the original image for pose attribute code. Different objects share the same encoder for the same attribute code.

We allocate an encoder called Neural Inversion Encoder with a similar structure to each code. Neural Inversion Encoder consists of three parts as Figure 5(b) shows. The first part employs a standard feature pyramid over a ResNet [12] backbone like in pSp [31] to extract the image features. The second part, in which we designed a structure opposite to GIRAFFE's Neural rendering Block based on its architecture as Figure 5(a) shows, downsamples the images layer by layer using a CNN and then uses skip connections [12] to combine the layers, yielding a one-dimensional feature. The third layer employs an MLP structure to acquire the corresponding dimension of different codes.

Training multiple encoders simultaneously is difficult to converge due to the large number of parameters. Hence, we use the dataset generated by GIRAFFE to retain the true values of each code and train an encoder for one code at a time, to keep the other codes at their true values, greatly smoothing the training.

During encoder training, we use the Mean Squared Error (MSE) loss, perceptual loss (LPIPS) [42], and identity loss (ID) [11] between the reconstructed image and the original image, to be consistent with most 2D and 3D GAN inversion training methodologies. When training the affine codes (scale *s*, translation *t*, rotation *r*), we find that different combinations of values produce very similar images, e.g., moving an object forward and increasing its scale yield similar re-

Algorithm 1: Round-robin Optimization

Data: all codes $w \in W$ predicted by encoders, fixed GIRAFFE generator G, input image I; 1 Initialize $lr \quad w = 10^{-3}, w \in W$; while any \overline{lr} $w > 10^{-5}$ do $\mathbf{2}$ foreach $w \in W$ do 3 Sample δw ; $\mathbf{4}$ Compute $\delta \mathcal{L}(w)$ using Eq. 8; $\mathbf{5}$ 6 end Compute rank list using Eq. 9; 7 for each $w \in rank_list and lr_w > 10^{-5} do$ 8 Optimization w with \mathcal{L}_{opt} in Eq. 10 of I and $G(W; \theta)$; 9 if the \mathcal{L}_{opt} ceases to decrease for five consecutive iterations then 10 11 $lr \ w = lr \ w/2;$ \mathbf{end} 12 end 13 14 end

sults. However, the encoder can only predict one value at a time, hence we add the MSE loss of the predicted s,t,r values, and their true values, to compel the encoder to predict the true value.

$$\mathcal{L}_{enc} = \lambda_1 L_2 + \lambda_2 L_{lpips} + \lambda_3 L_{id},\tag{7}$$

where $\lambda_i, i = 1, 2, 3$ represent the ratio coefficient between various losses. When training O^s, O^t, O^r code, the L_2 loss includes the MSE loss between the real values of O^s, O^t, O^r and their predicted values.

4.4 Precise Optimization

Pre-trained segmentation models have some segmentation errors and all encoderbased GAN inversion networks [31,35,36] usually cannot accurately obtain codes, necessitating refinements. Next, we optimize the coarse codes. Through experiments, we have found that using a single optimizer to optimize all latent codes tends to converge to local minima. Hence, we employ multiple optimizers, each handling a single code. The optimization order is crucial due to the variance of the disparity between the predicted and actual values across different encoders, and the different impact of code changes on the image, e.g., changes to B^{shape} and B^{app} codes controlling background generation mostly would have a larger impact on overall pixel values. Prioritizing the optimization of codes with significant disparity and a high potential for changing pixel values tends to yield superior results in our experiments. Hence, we propose an automated roundrobin optimization algorithm (Algorithm 1) to sequentially optimize each code based on the image reconstructed in each round.

Algorithm 1 aims to add multiple minor disturbances to each code, and calculate the loss between the images reconstructed before and after the disturbance

and the original image. A loss increase indicates that the current code value is relatively accurate, hence its optimization order can be postponed, and vice versa. For multiple codes that demand prioritized optimization, we compute their priorities using the partial derivatives of the loss variation and perturbation. We do not use backpropagation automatic differentiation here to ensure the current code value remains unchanged.

$$\delta \mathcal{L}(w) = \mathcal{L}(G(W - \{w\}, w + \delta w, \theta), I) - \mathcal{L}(G(W, \theta), I),$$
(8)

$$rank_list = F_{rank}(\delta \mathcal{L}(w), \frac{\delta \mathcal{L}(w)}{\delta w}), \tag{9}$$

where $w \in W$ is one of the codes and δw represents the minor disturbance of w. For the rotation angle r, we have found that adding a depth loss can accelerate its optimization. Thus, the loss \mathcal{L} during optimization can be expressed as:

$$\mathcal{L}_{opt} = \lambda_1 L_2 + \lambda_2 L_{lpips} + \lambda_3 L_{id} + \lambda_4 L_{deep}.$$
 (10)

This optimization method allows for more precise tuning of the codes for more accurate reconstruction and editing of the images.

5 Implementation

Neural Inversion Encoder. The first part of our encoder uses ResNet50 to extract features. In the second part, we downsample the extracted features (512-dimensional) and the input RGB image (3-dimensional) together. The two features are added together through skip connections, as shown in Figure 3. In the downsampling module, we use a 2D convolution with a kernel of 3 and a stride of 1, and the LeakyReLU activation function, to obtain a 256-dimensional intermediate feature. For object shape/appearance attributes, the output dimension is 256, and we use four Fully Connected Layers $\{4 \times FCL(256, 256)\}$ to get the codes. For background shape/appearance attributes, the output dimension is 128, we use $\{FCL(256, 128) + 3 \times FCL(128, 128)\}$ to get the codes. For object scale/translation attributes, the output dimension is 3, and we use the network $\{FCL(2^i, 2^{i-1}) + FCL(8, 3), i = 8, ..., 4\}$ to get the codes. For camera pose and rotation attributes, the output dimension is 1, and we use a similar network $\{FCL(2^i, 2^{i-1}) + FCL(8, 1), i = 8, ..., 4\}$ to get the codes.

Training and Optimization are carried out on a single NVIDIA A100 SXM GPU with 40GB of memory, using the Adam optimizer. The initial learning rate is set to 10^{-4} and 10^{-3} , respectively. Encoder training employs a batch size of 50. Each encoder took about 12 hours to train, and optimizing a single image of a complex multi-object scene took about 1 minute. For rotation features, it is difficult for the encoder to make accurate predictions for some images. Therefore, we uniformly sampled 20 values in the range of $[0, 360^{\circ}]$ for the

rotation parameters with large deviations. We selected the value that minimizes the loss in Equation 7 as the initial value for the optimization stage.

For LPIPS loss, we employ a pre-trained AlexNet [20]. For ID calculation, we employ a pre-trained Arcface [8] model in human face datasets and a pre-trained ResNet-50 [33] model in the car dataset. For depth loss, we use the pre-trained Dense Prediction Transformer model. We set $\lambda_1 = 1$, $\lambda_2 = 0.8$, and $\lambda_3 = 0.2$ in Equation 7, as well as in Equation 10, in which $\lambda_4 = 1$.

6 Experiment

Datasets. To obtain the true values of the 3D information in GIRAFFE for stable training performance, we use the pre-trained model of GIRAFFE on CompCars [40] and Clevr [15] dataset to generate training datasets. For testing datasets, we also use GIRAFFE to generate images for multi-car datasets denoted as G-CompCars (CompCars is a single car image dataset) and use the original Clevr dataset for multi-geometry dataset (Clevr is a dataset that can be simulated to generate images of multiple geometries). We follow the codes setup in GIRAFFE. For CompCars, we use all the codes from Equation 5. For Clevr, we fixed the rotation, scale, and camera pose codes of the objects. For experiments on facial data, we utilized the FFHQ [17] dataset for training and the CelebA-HQ [16] dataset for testing.



Fig. 6: Single-object editing on *G-CompCars* dataset. Co-R: coarse reconstruction. Pre-R: precise reconstruction.



Fig. 7: Single-object editing on *Clevr* dataset.

Baselines. In the comparative experiments for our Neural Inversion Encoder, we benchmarked encoder-based inversion methods such as e4e [35] and pSp [31], which use the 2D GAN StyleGAN2 [18] as the generator, and E3DGE [21] and



Fig. 8: Multi-object editing on G-CompCars dataset.

 (a) Input, Co-R, Pre-R
 (b) Edit Appearance
 (c) Edit Translation
 (d) Add/Remove Objects

Fig. 9: Multi-object editing on Clevr dataset.

TriplaneNet [5] that employ the 3D GAN EG3D [7] as the generator, on the generator of GIRAFFE. Additionally, we compared our encoder on StyleGAN2 with SOTA inversion methods HyperStyle [2] and HFGI [36] for StyleGAN2. Metrics. We use Mean Squared Error (MSE), perceptual similarity loss (LPIPS) [42], and identity similarity (ID) to measure the quality of image reconstruction.

6.1 3D GAN Omni-Inversion

Single-object Multifaceted Editing. In Figure 6 and Figure 7, (a) depict the original images, the coarsely reconstructed images produced by the Neural Inversion Encoder, and the precisely reconstructed images obtained via round-robin optimization. As Figure 7 shows, the simple scene structure of the Clevr dataset allows us to achieve remarkably accurate results using only the encoder (Co-Recon). However, for car images in Figure 6, predicting precise codes using the encoder only becomes challenging, necessitating the employment of the round-robin optimization algorithm to refine the code values for precise reconstruction (Pre-Recon). Figure 6 (b)-(h) and Figure 7 (b)-(d) show the editing results for different codes. As noted in Section 4.3, moving an object forward and increasing its scale yield similar results. Please refer to the Supplementary Material 3.1 for more results like camera pose and shape editing.

Multi-object Multifaceted Editing. We notice that the prediction for some object parameters (O^{shape} , O^{app} , O^s , O^t) are quite accurate. However, the prediction for the background codes deviates significantly. We speculate this is due to the significant differences in segmentation image input to the background encoder between multi-object scenes and single-object scenes. Therefore, background reconstruction requires further optimization. Figure 8 and Figure 9 depict the multifaceted editing outcomes for two cars and multiple Clevr objects, respectively. The images show individual edits of two objects in the left and middle images and collective edits at the right images in Figure 8 (b-c) and (f-h).



(a) Reconstruction results of different GAN inversion encoders using the generator of GI-RAFFE.

(b) Reconstruction results of different GAN inversion encoders using the generator of Style-GAN2.

Fig. 10: Reconstruction quality of different GAN inversion encoders.

As shown in Figure 8, the predictive discrepancy between the car's background and rotation angle on the left is considerable, requiring adjustments through the round-robin optimization. As illustrated in Figure 1, 2D/3D GAN inversion methods can not inverse multi-object scenes. More images pertaining to multi-object editing can be found in Supplementary Material 3.2.

6.2 Comparison Experiment of Neural Inversion Encoder

For fair comparison and to eliminate the impact of the generator on the quality of the inverted image generation, we trained the encoders from the baseline methods by connecting them to the GIRAFFE generator using our Neural Inversion Encoder training approach and compared them with our Neural Inversion Encoder. At the same time, we also connected our encoder to StyleGAN2 and compared it with inversion methods based on StyleGAN2, thereby demonstrating the efficiency of our encoder design. Table 1 and Figure 10 quantitatively and qualitatively displays the comparison results on both the GIRAFFE and StyleGAN2 generators respectively. The results show that our Neural Inversion Encoder consistently outperforms baseline methods.

6.3 Ablation Study

We conducted ablation experiments separately for the proposed Neural Inversion Encoder and the Round-robin Optimization algorithm. Table 2 displays the average ablation results of the Neural Inversion Encoder on various attribute codes, where NIB refers to Neural Inversion Block (the second part of the encoder) and MLP is the final part of the encoder. The results clearly show that our encoder structure is extremely effective and can predict code values more accurately. Please find the complete results in the Supplementary Material.

For the Round-robin optimization algorithm, we compared it with three fixed optimization order algorithms on both single-object and multi-object scenarios. The three fixed sequences are as follows:

 $Order1: B^{shape}, B^{app}, \{O^{r}_{i}, O^{t}_{i}, O^{s}_{i}\}_{i=1}^{N}, \{O^{shape}_{i}, O^{app}_{i}\}_{i=1}^{N}, C^{shape}_{i}, O^{app}_{i}\}_{i=1}^{N}, C^{shape}_{i}, O^{app}_{i}\}_{i=1}^{N}, C^{shape}_{i}, O^{shape}_{i}, O^{shape}_{i}\}_{i=1}^{N}, C^{shape}_{i}, O^{shape}_{i}, O^{shape}_{i}\}_{i=1}^{N}, C^{shape}_{i}, O^{shape}_{i}\}_{i=1}^{N}, C^{shape}_{i}, O^{shape}_{i}\}_{i=1}^{N}, C^{shape}_{i}, O^{shape}_{i}\}_{i=1}^{N}, O^{shape}_{i}\}_{i=1}^{N}, C^{shape}_{i}\}_{i=1}^{N}, C^{sha$

Table 1: Reconstruction quality of different GAN inversion encoders using the generator of GIRAFFE and StyleGAN2. \downarrow indicates the lower the better and \uparrow indicates the higher the better.

Method	GIRAFFE for Generator		StyleG	eGAN2 for Generator		
Method	$\mathrm{MSE}\downarrow$	$\text{LPIPS}\downarrow$	$\mathrm{ID}\uparrow$	$\mathrm{MSE}\downarrow$	$\text{LPIPS}\downarrow$	ID↑
e4e [35]	0.031	0.306	0.867	0.052	0.200	0.502
pSp [31]	0.031	0.301	0.877	0.034	0.172	0.561
HyperStyle [2]	-	-	-	0.019	0.091	0.766
HFGI [36]	-	-	-	0.023	0.124	0.705
TriplaneNet [5]	0.029	0.296	0.870	-	-	-
E3DGE [21]	0.031	0.299	0.881	-	-	-
3D-GOI(Ours)	0.024	0.262	0.897	0.017	0.098	0.769

Table 2: Ablation Study of the Neural Inversion Encoder.

Method	$\mathrm{MSE}\downarrow$	LPIPS↓	$\text{ID}\uparrow$
w/o NIB	0.023	0.288	0.856
w/o MLP	0.015	0.183	0.878
3D-GOI	0.010	0.141	0.906

Table 3: The quantitative metricsof ablation study of the Round-robinOptimization algorithm.

Method	$\mathrm{MSE}\downarrow$	$\text{LPIPS}\downarrow$	$ID\uparrow$
Order1	0.016	0.184	0.923
Order2	0.019	0.229	0.913
Order3	0.019	0.221	0.911
3D-GOI	0.008	0.128	0.938

$$\begin{array}{l} Order2: \{O_{i}^{r}, O_{i}^{t}, O_{i}^{s}\}_{i=1}^{N}, \{O_{i}^{shape}, O_{i}^{app}\}_{i=1}^{N}, B^{shape}, B^{app}, C \\ Order3: C, \{O_{i}^{shape}, O_{i}^{app}\}_{i=1}^{N}, \{O_{i}^{r}, O_{i}^{t}, O_{i}^{s}\}_{i=1}^{N}, B^{shape}, B^{app}, C \\ Order3: C, \{O_{i}^{shape}, O_{i}^{app}\}_{i=1}^{N}, \{O_{i}^{r}, O_{i}^{t}, O_{i}^{s}\}_{i=1}^{N}, B^{shape}, B^{app}, C \\ Order3: C, \{O_{i}^{shape}, O_{i}^{app}\}_{i=1}^{N}, \{O_{i}^{shape}, O_{i}^{shape}\}_{i=1}^{N}, B^{shape}, B^{app}, C \\ Order3: C, \{O_{i}^{shape}, O_{i}^{app}\}_{i=1}^{N}, \{O_{i}^{shape}, O_{i}^{shape}\}_{i=1}^{N}, B^{shape}, B^{app}, C \\ Order3: C, \{O_{i}^{shape}, O_{i}^{app}\}_{i=1}^{N}, \{O_{i}^{shape}, O_{i}^{shape}\}_{i=1}^{N}, B^{shape}, B^{app}, C \\ Order3: C, (O_{i}^{shape}, O_{i}^{shape})_{i=1}^{N}, C \\ Order3: C, (O_{i}^{shape}, O_{i}^{shape})_{i=1}^{N}, C \\ Order3: C, (O_{i}^{shape}, O_{i}^{shape})_{i=1}^{N}, C \\ Order$$

 $\{\}_{i=1}^{N}$ indicates that the elements inside $\{\}$ are arranged in sequence from 1 to N. There are many possible sequence combinations, and here we chose **the three with the best results** for demonstration. As Table 3 shows, our method achieves the best results on all metrics, demonstrating the effectiveness of our Round-robin optimization algorithm. As mentioned in Section 4.4, optimizing features like the background first can enhance the optimization. Hence, Order1 performs much better than Order2 and Order3. Please see the Supplementary Material 3.5 for qualitative comparisons of these four methods on images.

7 Conclusion

This paper introduces a 3D GAN inversion method, 3D-GOI, that enables multifaceted editing of scenes containing multiple objects. By using a segmentation approach to separate objects and background, then carrying out a coarse estimation followed by a precise optimization, 3D-GOI can accurately obtain the codes of the image. These codes are then used for multifaceted editing. To the best of our knowledge, 3D-GOI is the first method to attempt multi-object & multifaceted editing. We anticipate that 3D-GOI holds immense potential for future applications in fields such as VR/AR, and the Metaverse. Acknowledgements: This work was supported by the National Key Research and Development Program of China (2022YFB3105405, 2021YFC3300502).

References

- Abdal, R., Qin, Y., Wonka, P.: Image2stylegan: How to embed images into the stylegan latent space? In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 4432–4441 (2019)
- Alaluf, Y., Tov, O., Mokady, R., Gal, R., Bermano, A.: Hyperstyle: Stylegan inversion with hypernetworks for real image editing. In: Proceedings of the IEEE/CVF conference on computer Vision and pattern recognition. pp. 18511–18521 (2022)
- Arad Hudson, D., Zitnick, L.: Compositional transformers for scene generation. Advances in Neural Information Processing Systems 34, 9506–9520 (2021)
- Bau, D., Zhu, J.Y., Wulff, J., Peebles, W., Strobelt, H., Zhou, B., Torralba, A.: Seeing what a gan cannot generate. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 4502–4511 (2019)
- Bhattarai, A.R., Nießner, M., Sevastopolsky, A.: Triplanenet: An encoder for eg3d inversion. arXiv preprint arXiv:2303.13497 (2023)
- Brock, A., Donahue, J., Simonyan, K.: Large scale gan training for high fidelity natural image synthesis. arXiv preprint arXiv:1809.11096 (2018)
- Chan, E.R., Lin, C.Z., Chan, M.A., Nagano, K., Pan, B., De Mello, S., Gallo, O., Guibas, L.J., Tremblay, J., Khamis, S., et al.: Efficient geometry-aware 3d generative adversarial networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 16123–16133 (2022)
- Deng, J., Guo, J., Xue, N., Zafeiriou, S.: Arcface: Additive angular margin loss for deep face recognition. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 4690–4699 (2019)
- Deng, Y., Wang, B., Shum, H.Y.: Learning detailed radiance manifolds for highfidelity and 3d-consistent portrait synthesis from monocular image. arXiv preprint arXiv:2211.13901 (2022)
- Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial networks. Communications of the ACM 63(11), 139–144 (2020)
- He, K., Fan, H., Wu, Y., Xie, S., Girshick, R.: Momentum contrast for unsupervised visual representation learning. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 9729–9738 (2020)
- He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
- Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. Advances in neural information processing systems 33, 6840–6851 (2020)
- Huh, M., Zhang, R., Zhu, J.Y., Paris, S., Hertzmann, A.: Transforming and projecting images into class-conditional generative networks. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16. pp. 17–34. Springer (2020)
- Johnson, J., Hariharan, B., Van Der Maaten, L., Fei-Fei, L., Lawrence Zitnick, C., Girshick, R.: Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2901–2910 (2017)

- 16 H. Li et al.
- Karras, T., Aila, T., Laine, S., Lehtinen, J.: Progressive growing of gans for improved quality, stability, and variation. arXiv preprint arXiv:1710.10196 (2017)
- Karras, T., Laine, S., Aila, T.: A style-based generator architecture for generative adversarial networks. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 4401–4410 (2019)
- Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., Aila, T.: Analyzing and improving the image quality of stylegan. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 8110–8119 (2020)
- Ko, J., Cho, K., Choi, D., Ryoo, K., Kim, S.: 3d gan inversion with pose optimization. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 2967–2976 (2023)
- Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Communications of the ACM 60(6), 84–90 (2017)
- Lan, Y., Meng, X., Yang, S., Loy, C.C., Dai, B.: Self-supervised geometry-aware encoder for style-based 3d gan inversion. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 20940–20949 (2023)
- 22. Li, H., Shi, H., Zhang, W., Wu, W., Liao, Y., Wang, L., Lee, L.h., Zhou, P.: Dreamscene: 3d gaussian-based text-to-3d scene generation via formation pattern sampling. arXiv preprint arXiv:2404.03575 (2024)
- Lin, Y., Bai, H., Li, S., Lu, H., Lin, X., Xiong, H., Wang, L.: Componerf: Text-guided multi-object compositional nerf with editable 3d scene layout. arXiv preprint arXiv:2303.13843 (2023)
- 24. Metzer, G., Richardson, E., Patashnik, O., Giryes, R., Cohen-Or, D.: Latentnerf for shape-guided generation of 3d shapes and textures. arXiv preprint arXiv:2211.07600 (2022)
- Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. Communications of the ACM 65(1), 99–106 (2021)
- Nguyen-Phuoc, T., Li, C., Theis, L., Richardt, C., Yang, Y.L.: Hologan: Unsupervised learning of 3d representations from natural images. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 7588–7597 (2019)
- Nguyen-Phuoc, T.H., Richardt, C., Mai, L., Yang, Y., Mitra, N.: Blockgan: Learning 3d object-aware scene representations from unlabelled images. Advances in neural information processing systems 33, 6767–6778 (2020)
- Niemeyer, M., Geiger, A.: Giraffe: Representing scenes as compositional generative neural feature fields. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11453–11464 (2021)
- Perarnau, G., Van De Weijer, J., Raducanu, B., Álvarez, J.M.: Invertible conditional gans for image editing. arXiv preprint arXiv:1611.06355 (2016)
- Poole, B., Jain, A., Barron, J.T., Mildenhall, B.: Dreamfusion: Text-to-3d using 2d diffusion. arXiv preprint arXiv:2209.14988 (2022)
- Richardson, E., Alaluf, Y., Patashnik, O., Nitzan, Y., Azar, Y., Shapiro, S., Cohen-Or, D.: Encoding in style: a stylegan encoder for image-to-image translation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 2287–2296 (2021)
- Roich, D., Mokady, R., Bermano, A.H., Cohen-Or, D.: Pivotal tuning for latentbased editing of real images. ACM Transactions on Graphics (TOG) 42(1), 1–13 (2022)
- 33. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual recognition challenge. International journal of computer vision 115, 211–252 (2015)

- Schwarz, K., Liao, Y., Niemeyer, M., Geiger, A.: Graf: Generative radiance fields for 3d-aware image synthesis. Advances in Neural Information Processing Systems 33, 20154–20166 (2020)
- Tov, O., Alaluf, Y., Nitzan, Y., Patashnik, O., Cohen-Or, D.: Designing an encoder for stylegan image manipulation. ACM Transactions on Graphics (TOG) 40(4), 1–14 (2021)
- Wang, T., Zhang, Y., Fan, Y., Wang, J., Chen, Q.: High-fidelity gan inversion for image attribute editing. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11379–11388 (2022)
- Wei, T., Chen, D., Zhou, W., Liao, J., Zhang, W., Yuan, L., Hua, G., Yu, N.: E2style: Improve the efficiency and effectiveness of stylegan inversion. IEEE Transactions on Image Processing **31**, 3267–3280 (2022)
- Xie, J., Ouyang, H., Piao, J., Lei, C., Chen, Q.: High-fidelity 3d gan inversion by pseudo-multi-view optimization. arXiv preprint arXiv:2211.15662 (2022)
- Yang, H., Zhang, Z., Yan, S., Huang, H., Ma, C., Zheng, Y., Bajaj, C., Huang, Q.: Scene synthesis via uncertainty-driven attribute synchronization. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5630–5640 (2021)
- Yang, J., Li, H.: Dense, accurate optical flow estimation with piecewise parametric model. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1019–1027 (2015)
- Yin, F., Zhang, Y., Wang, X., Wang, T., Li, X., Gong, Y., Fan, Y., Cun, X., Shan, Y., Oztireli, C., et al.: 3d gan inversion with facial symmetry prior. arXiv preprint arXiv:2211.16927 (2022)
- 42. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 586–595 (2018)
- Zhu, J., Shen, Y., Zhao, D., Zhou, B.: In-domain gan inversion for real image editing. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVII 16. pp. 592–608. Springer (2020)
- 44. Zhu, J.Y., Krähenbühl, P., Shechtman, E., Efros, A.A.: Generative visual manipulation on the natural image manifold. In: Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part V 14. pp. 597–613. Springer (2016)
- 45. Zhu, J.Y., Park, T., Isola, P., Efros, A.A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. In: Proceedings of the IEEE international conference on computer vision. pp. 2223–2232 (2017)

Supplementary Material

8 Preliminary

NeRF [25] is a recently rising approach for 3D reconstruction tasks that employs a neural radiance field to represent a scene. It allows for mapping highdimensional positional codes from any viewing direction d and spatial coordinates x to color c and opacity values σ and then synthesizes images corresponding to the specified view using a volume rendering equation. We use Equation 11 to succinctly describe this process:

$$\begin{array}{l} (\gamma(\boldsymbol{x}), \gamma(\boldsymbol{d})) \xrightarrow{f_{\boldsymbol{\theta}}} (\sigma, \boldsymbol{c}) \\ \mathbb{R}^{L_{\boldsymbol{x}}} \times \mathbb{R}^{L_{\boldsymbol{d}}} \xrightarrow{f_{\boldsymbol{\theta}}} \mathbb{R}^{+} \times \mathbb{R}^{3} \end{array}$$
(11)

where γ represents the positional encoding function utilized to incorporate highdimensional information into \boldsymbol{x} and \boldsymbol{d} and obtained the output $\gamma(\mathbf{x})$, $\gamma(\boldsymbol{d})$ of dimension L_x, L_d , respectively. γ is typically represented using trigonometric functions, such as $\gamma(t, L) = (sin(2^0t\pi), cos(2^0t\pi), ..., sin(2^{L-1}t\pi), cos(2^{L-1}t\pi))$. θ represents the parameters of the mapping function f.

Equation 12 delineates the volume rendering formula that predicts color $C(\mathbf{r})$ for a camera ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$ within the near and far bounds tn and tf. Here, T(t) signifies the cumulative transmittance along the ray from tn to t.

$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(r(t))c(r(t),d)dt,$$

$$where \quad T(t) = exp(-\int_{t_n}^t \sigma(r(s))ds)$$
(12)

GRAF [34] is a generative neural radiance field adding additional latent codes like object shape z_s and appearance z_a to NeRF, allowing control not only the shape and appearance of the object but also the camera pose of the image. $z_s, z_a \sim \mathcal{N}(0, I)$ and the mapping function g_θ of the radiance field of GRAF can be expressed as follows:

$$(\gamma(\boldsymbol{x}), \gamma(\boldsymbol{d}), \boldsymbol{z}_{\boldsymbol{s}}, \boldsymbol{z}_{\boldsymbol{a}}) \xrightarrow{g_{\boldsymbol{\theta}}} (\sigma, \boldsymbol{c})$$
$$\mathbb{R}^{L_{x}} \times \mathbb{R}^{L_{d}} \times \mathbb{R}^{M_{s}} \times \mathbb{R}^{M_{a}} \xrightarrow{g_{\boldsymbol{\theta}}} \mathbb{R}^{+} \times \mathbb{R}^{3},$$
(13)

where Ms and Ma are the dimensions of z_s and z_a , respectively. GRAF renders images using a volume rendering formula similar to that of NeRF.

GIRAFFE [28] perceives an image scene as a composition of the background and multiple foreground objects, each subjected to affine transformations. Each object can be manipulated and placed at a specific location k(x) in the image through operations of scaling S, translation t, and rotation R:

$$k(\boldsymbol{x}) = \boldsymbol{R} \cdot \boldsymbol{S} \cdot \boldsymbol{x} + \boldsymbol{t}, \tag{14}$$

where \mathbf{x} is the spatial coordinate in the object space.

To better compose scenes, GIRAFFE replaces the three-dimensional color output in GRAF's Equation 13 with a high-dimensional feature field. GIRAFFE renders in scene space and evaluates the feature field in the object space. Hence, the mapping function of radiance field h_{θ} of GIRAFFE in object space can be expressed as follows:

$$(\gamma(k^{-1}(\boldsymbol{x})), \gamma(k^{-1}(\mathbf{d})), \boldsymbol{z_s}, \boldsymbol{z_a}) \xrightarrow{h_{\theta}} (\sigma, \boldsymbol{f})$$
$$\mathbb{R}^{L_x} \times \mathbb{R}^{L_d} \times \mathbb{R}^{M_s} \times \mathbb{R}^{M_a} \xrightarrow{h_{\theta}} \mathbb{R}^+ \times \mathbb{R}^{M_f},$$
(15)

where k^{-1} is the inverse function of k, M_f is the dimension of the feature field f.

In the construction of multi-object scenes, GIRAFFE employs a compositing operation C to merge the feature fields of multiple objects and the background together. The features at (x, d) can be expressed as:

$$C(\boldsymbol{x}, \boldsymbol{d}) = (\sigma, \frac{1}{\sigma} \sum_{i=1}^{N} \sigma_i \boldsymbol{f}_i), where \quad \sigma = \sum_{i=1}^{N} \sigma_i,$$
(16)

where N is the number of objects plus one (the background), σ_i and f_i represent the density value and feature field of the i - th object (or the background).

The rendering process of GIRAFFE can be divided into two stages. In the first stage, feature fields are used instead of color for volume rendering like in NeRF to get a low-resolution feature map:

$$\boldsymbol{f} = \sum_{i=1}^{N_s} \tau_j \alpha_j \boldsymbol{f}_{\boldsymbol{j}}, \quad \tau_j = \prod_{k=1}^{j-1} (1 - \alpha_k), \quad \alpha_j = 1 - e^{-\sigma_j \delta_j}, \quad (17)$$

where α_j is the alpha value of the coordinates x_j , τ_j represents the transmittance, and $\delta_j = ||x_{j+1} - x_j||_2$ is the distance between the neighboring sampled points x_{j+1} and x_j . The second stage is called neural rendering, which transforms low-resolution feature maps into high-resolution images through an upsampling network.

9 Implementation

The round-robin optimization algorithm works well when the discrepancy between the coarse estimation of the Neural Inversion Encoder and the actual results is not too large. This is because in the presence of a slight perturbation in the codes, an increase in the loss of Equation 8 in the paper doesn't necessarily conclude that the code has reached its true value. Otherwise, if the encoder cannot make a rough prediction of the code, or if one wishes to forgo using the encoder and rely solely on the optimization method, we offer a program for manually selecting the current optimization code interactively. This allows the image to be manually optimized to a certain degree of difference from the original image before using the round-robin optimization algorithm for automatic optimization.

Table 4: Architecture comparison for different GAN inversion methods. SG2 indicates StyleGAN2. "2D/3D" indicates whether 2D or 3D editing is possible. "object" indicates whether the method can edit a single object or multiple objects. "code" indicates the number of codes that the method can invert.

Method	Generator	$2\mathrm{D}/3\mathrm{D}$	Object	Code
e4e [35]	SG2	2D	single	1
pSp [31]	SG2	2D	single	1
PTI [32]	SG2	2D	single	1
HyperStyle [2]	SG2	2D	single	1
HFGI [36]	SG2	2D	single	1
TriPlaneNet [5]	EG3D	3D	single	2
E3DGE [21]	EG3D	3D	single	2
SPI [41]	EG3D	3D	single	2
3D-GOI	GIRAFFE	$2\mathrm{D}/3\mathrm{D}$	$\operatorname{single}/\operatorname{multi}$	5n+3

10 Additional Results

Baselines. We added another 2D GAN inversion method PTI [32] based on StyleGAN2 [18], and a 3D GAN inversion method SPI [41] based on EG3D [7], to validate the performance of our method in the novel viewpoint synthesis task. Table 4 compares the structures and capabilities of various GAN Inversion methods.

10.1 Single-object multifaceted editing

Figure 11 and 12 depict the additional results of our multifaceted edits on a single object.



(i) Edit Camera Pose

Fig. 11: Single-object editing performance on G-CompCars dataset.

10.2 Multi-object multifaceted editing

As shown in the Figure 13 and 14, we demonstrate the additional results of our multifaceted edits on multiple objects.



Fig. 12: Single-object editing performance on Clevr dataset.



Fig. 13: Multi-object editing performance on G-CompCars dataset.



 ${\bf Fig. 14:} \ {\rm Multi-object} \ {\rm editing} \ {\rm performance} \ {\rm on} \ {\rm Clevr} \ {\rm dataset}.$



Fig. 15: Novel views synthesis for human faces of different GAN inversion methods.



Fig. 16: The figure of the reconstruction result of inaccurate segmentation.

10.3 Novel views synthesis for human faces

We also test the synthesis of novel views of the face, which is a minor ability of 3D-GOI yet the major ability of existing 3D GAN inversion methods. Figure 15 shows that our method has better performance than the latest 3D inversion method SPI [41] and some advanced 2D inversion methods that can generate novel views such as PTI [32] and SG2(StyleGAN2) [18].

10.4 Inaccurate segmentation

Figure 16 shows the reconstruction result of 3D-GOI with inaccurate segmentation. Both accurate and inaccurate segmentation can reconstruct the original image well with only minor differences, which demonstrates the robustness of our model.

10.5 Ablation Study

Table 5 shows the results of the ablation experiments on each attribute encoder. It shows that our added NIB structure can greatly improve the prediction ac-

Method	attribute codes	$\mathrm{MSE}\downarrow$	LPIPS↓	ID ↑
	O^{shape}	0.046	0.412	0.811
Method 3D-GOI (w/o NIB) 3D-GOI (w/o MLP) 3D-GOI	O^{app}	0.006	0.092	0.907
	O^s	0.025	0.269	0.856
3D-GOI	O^t	0.036	0.340	0.848
(w/o NIB)	O^r	0.031	0.343	0.805
	B^{shape}	0.030	0.400	0.812
	B^{app}	0.009	0.155	0.881
	C	0.001	0.289	0.929
	average	0.023	0.288	0.856
	O^{shape}	0.030	0.286	0.850
	O^{app}	0.004	0.075	0.916
	O^s	0.012	0.157	0.889
3D-GOI	O^t	0.016	0.199	0.877
(w/o MLP)	O^r	0.025	0.280	0.827
	B^{shape}	0.022	0.316	0.837
	B^{app}	0.006	0.120	0.898
	C	0.001	0.029	0.929
	average	0.015	0.183	0.878
	O^{shape}	0.008	0.116	0.913
	O^{app}	0.005	0.084	0.931
	O^s	0.005	0.084	0.924
2D COI	O^t	0.010	0.138	0.905
3D-GOI	O^r	0.022	0.257	0.855
	B^{shape}	0.021	0.332	0.853
	B^{app}	0.005	0.116	0.922
	C	0.001	0.002	0.941
	average	0.010	0.141	0.906

 Table 5: Ablation Study of the Neural Inversion Encoder of different attribute codes.

curacy, and that ${\cal O}^{shape},\,{\cal B}^{shape}$ and ${\cal O}^r$ are more difficult to predict than other codes.

Figure 17 shows the result of using only one optimizer for all codes. For a single object image, even though our encoder can estimate the codes more accurately as shown in Figure 10 in the paper, the optimizer is still unable to reconstruct the image accurately, which is even more obvious for multi-object codes that require more codes to be controlled.

Figure 18 is a comparison of the four methods. As shown, our method achieves the best results on all metrics, demonstrating the effectiveness of our round-robin optimization algorithm. Figure 18 clearly shows that using a fixed order makes it difficult to optimize back to the image, especially in multi-object images. As mentioned in Section 4.4, optimizing features like the image background first can enhance the optimization results. Hence, Order1 performs much better than Order2 and Order3.



Fig. 17: The result of optimizing all codes using only one optimizer.

 Table 6: The comparison of encoder-based 3D inversion methods for computational costs.

Method	parameter numbers	FLOPs	time(s)
E3DGE (single encoder)	90M	50G	0.07
TriplaneNet (single encoder)	247M	112G	0.11
3D-GOI(multi encoders)	169M	165G	0.08

10.6 Computational costs

We believe it is reasonable that for editing images with multiple objects in a multifaceted manner, the computational cost is positively correlated with the number of objects in the image. Furthermore, in tasks of reconstructing single objects, all our Neural Inversion encoders indeed incur more computational cost compared to the baselines E3DGE [21] and TriplaneNet [5] as shown in Table 6. That is due to our goal of editing multiple objects diversely so it necessitates separate encoding predictions for various attributes of objects and backgrounds in the image, especially for affine transformation attributes, which most inversion works fail to achieve. In practice, in our experiments, the time consumed for encoding is minimal, with all codes outputted within 0.1 second. Our main time consumption is in the optimization part, but since we optimize all codes directly, even using a per-code round-robin optimization strategy is faster than the current mainstream algorithms SPI [41] and PTI [32] that require optimization of generator parameters as shown in Table 7.

11 Limitations

Despite the impressive generative capabilities of GIRAFFE, we encountered several notable issues in the tests. Notably, there was a gap between the data distribution generated by GIRAFFE and that of the original datasets, which is the main problem faced by current complex scene generation methods, making it difficult to inverse in-the-wild images. Additionally, we observed interaction effects among different codes in some of the GIRAFFE-generated images, which further complicated our inversion targets.

PTI 55 SPI 550 3D-GOI 30

 Table 7: The comparison of hybrid-based 3D inversion methods for time costs.

Method time(s)

Fig. 18: The figure of ablation study of the round-robin Optimization algorithm.

We believe that with the advancement of complex multi-object scene generation methods, our editing method 3D-GOI will hold immense potential for future 3D applications such as VR/AR and Metaverse.

12 Futuer work

As the first work in this new field, our current primary focus is on the accuracy of reconstruction. Our present encoding and optimization strategies are mainly aimed at achieving more precise reconstruction, while we have not given enough consideration to computational cost. Moving forward, we will continue to design the structure of the encoder to enable it to predict codes more quickly and accurately. Additionally, we need to address the entanglement issue in GIRAFFE, allowing each code to independently control the image, which may simplify our entire method process. Lastly, we need to solve the generalization issue in GAN inversion, which may require training on more real-world datasets.

13 Ethical considerations

Generative AI models in general, including our proposal, face the risk to be used for spreading misinformation. The authors of this paper do not condone such behaviors.