SYM3D: Canonicalizing Triplanes via Symmetry for Single-View 3D Learning

Jing Yang, Kyle Fogarty, Fangcheng Zhong, Cengiz Oztireli
University of Cambridge, UK

Abstract

Triplane is an effective 3D representation capable of generating high-quality 3D assets from 2D images by synthesizing rich details from multiple viewing directions. However, it relies heavily on multi-view images with camera annotations to achieve sufficient detail across all perspectives. The limited availability of calibrated multi-view image datasets, especially when compared to single-view images, has left triplane representation significantly underconstrained. In this work, we introduce SYM3D, which harnesses the common reflectional symmetry found in natural and man-made objects to regulate triplane learning. However, symmetry alone does not resolve the ambiguity in each plane's focus within the 3D space without camera annotations. To address this, we incorporate a view-aware spatial attention mechanism that aligns each plane of the triplane with a consistent focus in 3D space. We evaluate SYM3D on the triplane representation across various frameworks (3D-GAN and diffusion), tasks (reconstruction, unconditional generation, and text-to-3D), and datasets, including synthetic (ShapeNet Chairs, Cars, and Airplanes) and real-world data (ABO-Chair). The results demonstrate the effectiveness of integrating symmetry regularization. More details can be found at our project page: https://jingyang2017.github.io/sym3d.github.io/.

1. Introduction

Symmetry is a foundational trait in both natural and human-made objects, providing structural balance, aesthetic appeal, and functional coherence. Reflective symmetry, where one side of an object mirrors the other, is pervasive in diverse subjects, from organic structures like butterfly wings to man-made artifacts such as buildings, vehicles, and furniture [16]. In 3D modeling [15, 26, 27, 29], this symmetry acts as a useful prior, providing implicit information about an object's hidden views.

The triplane representation, introduced in EG3D [5], has proven highly effective in 3D generative modeling, offering a balance of memory efficiency, expressivity, and rendering quality. In the 3D-GAN framework, GET3D [10] creates textured meshes with separate triplanes for geometry and texture. More recent diffusion models, such as SSDNerf [7] and NFD [21], leverage multi-view images and camera annotations to model triplane distributions, generating 3D objects by progressively denoising sampled triplane codes. However, these models struggle with single-view training, where triplane codes become underdetermined, compromising accurate 3D generation.

Our core insight is that integrating a symmetry prior into the triplane representation can address the limitations of single-view training in 3D modeling. By enforcing reflectional symmetry constraints, the model can utilize symmetry to infer unseen details, resulting in more consistent and well-oriented 3D assets. This symmetry prior establishes a unified canonical orientation across objects, improving model accuracy and preserving essential structural features, both of which are vital for high-quality 3D asset generation.

To validate our approach, we select GET3D as our foundation model for its superior performance over other 3D-GAN methods [5, 18], its capability to generate explicit 3D meshes valuable for downstream applications requiring structured outputs, and its flexibility over diffusion-based methods, allowing training 3D representations without requiring ground-truth images or camera annotations for supervision. We test on symmetric datasets featuring centered objects in 2D images. This structured setup allows us to explore the potential of our idea in controlled conditions, while also highlighting its adaptability to more complex, real-world applications, such as text-to-3D generation. Unlike [22] trained on ImageNet with near-frontal single images from diverse categories, our approach focuses on learning detailed 3D representations for specific categories that can be rendered from complete 360° views.

In this work, we propose SYM3D, a symmetry-aware triplane model designed to learn from *single-view* images *without* camera pose annotations, enabling 3D textured mesh generation for symmetrical objects. Our framework incorporates view-aware spatial attention and symmetry regularization, enhancing the triplane model to better adhere to its intended structure: three axis-aligned planes designed to capture the top, bottom, and side views of an object. The view-aware spatial attention directs each geometry

Figure 1. Comparison of shapes generated by GET3D [10] and our SYM3D, rendered in Blender. SYM3D learns symmetric triplanes for improving 3D-awareness of GANs. Compared to GET3D, SYM3D can synthesize diverse objects with reasonable geometry and texture after training it on datasets with incomplete views. Refer Section 4.1 for dataset details.

plane to focus on specific viewpoints, while symmetry regularization enforces reflectional symmetry, reducing ambiguity and promoting consistency in the generated assets (see Figure 1 for a comparison between GET3D and SYM3D).

Our main contributions are summarized as follows:

- We introduce a symmetry-aware triplane representation that enhances 3D awareness in triplane-based models with minimal additional computational cost, effectively capturing both geometry and texture from single-view images of symmetric objects.
- We are the first to incorporate symmetry regularization into the triplane learning process, enabling complete representations of symmetric objects even when trained on datasets with incomplete views.
- Our experiments show the superior performance of the symmetry-enhanced triplane in generating high-quality 3D assets across various categories of symmetric objects (chair, car, airplane), diverse datasets (synthetic ShapeNet [6], real-world Amazon Berkeley Objects [9]), and tasks (unconditional generation, reconstruction, and text-to-3D), underscoring its potential for both synthetic and real-world applications.

2. Related work

Triplane Representation for Generative models. Since EG3D [5] introduces the concept of triplane representation, this representation has become increasingly popular in 3D modeling for its efficiency and effectiveness. A line of works focuses on improving triplane representations, for example, He et al. [11] inject an extra axis to 2D triplane by preserving information associated with projection distance. Wu and Zheng [25] develop a multi-scale triplane in a hierarchical fashion that enables learning 3D shapes from a single example, progressively refining from coarse to detailed features. Another line of works distills the triplane representation to describe a single object. Trevithick et al. [23] learn a triplane representation to describe an unposed face from a pretrained EG3D, which facilitates real-time, photo-realistic 3D face rendering. Bhattarai et al. [4] improve GAN inversion by learning offsets to adjust the triplane representation. Our work improves triplane representation by introducing structure awareness, which enhances the triplane's ability to capture distinct viewpoints of an object. This enhancement is achieved in conjunction with symmetry regularization, allowing for a more holistic representation in training 3D-aware generators.

Reflectional Symmetry in Image Synthesis. Symmetric objects are widespread in nature, architecture, and art, and reflectional symmetry plays a key role in human visual perception [16]. This concept has been extensively utilized in computer vision research. A significant research direction in this domain focuses on applying reflectional symmetry principles to enhance the rendering of 2D images. Wu et al. [26] investigate the inference of 3D deformable objects from single images by employing symmetric structures to distinguish between depth, albedo, viewpoint, and illumination components. NeRD [27] presents a neural detector designed to identify 3D reflection symmetry in objects, estimating the normal vectors of their mirror planes. Yin et al. [29] have advanced 3D GAN inversion techniques by training with mirrored images, leveraging symmetry to enhance the quality of the results. SymmNeRF [15] represents an innovative strategy that explicitly incorporates symmetry into the training of neural radiance fields, utilizing both pixel-aligned image features and their symmetric analogs as additional training inputs. We introduce symmetry regular-

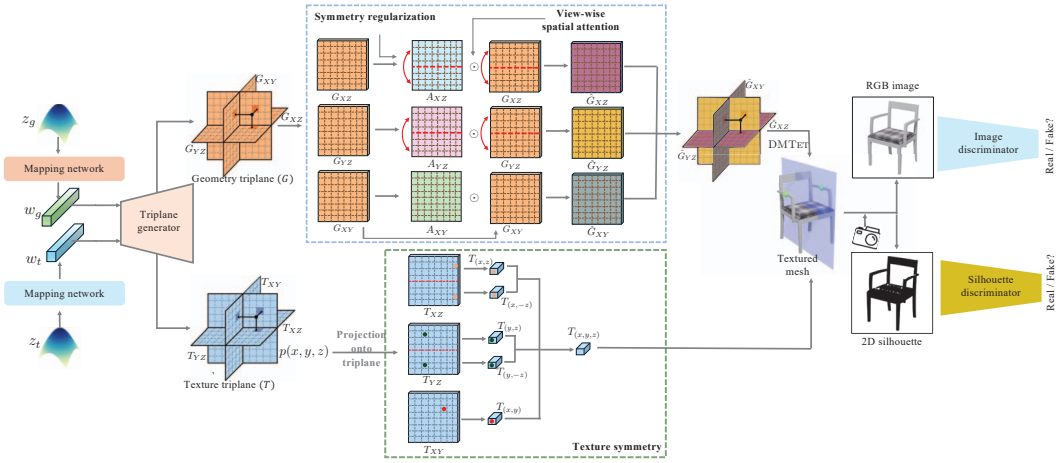

Figure 2. Overview of proposed SYM3D. Random input vectors z_g and z_t are first mapped to a latent space (w_g and w_t) and then fed into a shared generator to create the axis-aligned triplanes: geometry triplane G and texture triplane T. We assume that the shapes being modeled have a symmetry plane (XY) such that a subset of the axis-aligned planes (YZ, XZ) can be regularized to exploit such symmetry. We apply view-wise attention (Section 3.2) on geometry triplane, and regulate both geometry triplane and attention map with reflectional symmetry (Section 3.3). We use DMTet method [20] to extract a 3D mesh. We describe a surface point p with both the original and its reflective feature in texture triplane. Using differentiable rendering [14], we render RGB images and their silhouettes from different camera angles. We then use two discriminators to determine whether the RGB and silhouette images are real or fake, without requiring the camera pose of real images.

ization to both geometry and texture, ensuring more accurate and realistic 3D asset generation. Our insight is that the symmetry prior encodes information about the unseen view of an object when only a single view is accessible.

3. Method

In this section, we introduce the pipeline of our proposed 3D symmetry-aware textured mesh generation method (SYM3D), as shown in Figure 2.

3.1. Triplane Representation of 3D Assets

Our representation is built upon GET3D [10] for its capability to produce high-quality textured meshes and its effective separation of geometry from texture through triplanes [5]. Following GET3D, we maintain two sets of triplanes, $G = \{G_{XY}, G_{XZ}, G_{YZ}\}$ for geometry and $T = \{T_{XY}, T_{XZ}, T_{YZ}\}$ for texture, effectively storing the shape and texture information of 3D objects. A triplane consists of three axis-aligned orthogonal feature planes, each with size $N \times N \times C$ where N is the spatial resolution and C is the number of channels. To create a 3D asset, GET3D incorporates DMTet [20] in the generator, which represents geometry as a signed distance field (SDF) defined on a deformable tetrahedral volume grid. For any vertex $p = (x, y, z)$ in the tetrahedral grid, we calculate its geometry feature by first projecting it onto XY, XZ and YZ planes based on its 2D coordinate $(x, y), (x, z), (y, z)$, and then querying and aggregating features from these projections as described by:

$$G_{(x,y,z)} = G_{(x,y)} + G_{(x,z)} + G_{(y,z)}. \quad (1)$$

The feature vector $G_{(x,y,z)}$ then represents the geometric features of p and is used to infer the SDF value and deformation of the tetrahedral volume grid. After computing SDF values and deformations, the differential marching tetrahedral algorithm extracts the explicit mesh. To shade a surface point, the texture feature $T_{(x,y,z)}$ is calculated through a comparable process to Eq 1 and is used to predict the RGB color. With a known mesh structure, it simplifies computations by only requiring surface point queries, significantly reducing computational complexity. To produce high-quality textured meshes, GET3D was trained with synthesized multi-view images rendered from various objects with camera pose annotations.

After training GET3D in single images, we observe that the geometry-based triplane representation, which uses three axis-aligned planes intended to capture the top, bottom, and side views of an object, often results in planes that are highly similar. The difficulty in training a triplane to have factorization features along axes in a 3D GAN setup arises during the optimization phase. Here, the generator focuses solely on creating a realistic 2D image from a specific view, overlooking the creation of a consistent high-quality 3D shape (see Figure 7).

To refine the triplane representation, we establish two primary goals: first, ensuring that each plane maintains a consistent focal region; and second, enforcing symmetry in at least two of the three planes to accurately reflect the inherent symmetry of the object. To achieve these objectives, we incorporate view-wise spatial attention, which directs each plane to focus on specific scene regions from distinct

viewpoints, and reflectional symmetry, which aligns the feature planes with the natural symmetry [16] of the modeled object. [16] of the object being modeled. This dual approach enables triplane learning to simultaneously deceive the discriminator and maintain a cohesive, accurate 3D representation, ultimately enhancing both the effectiveness and realism of generated 3D models.

3.2. View-wise Spatial Attention

Due to the lack of multi-view supervision when learning triplane representations, each feature piece ($N \times N \times 1$) in the triplane struggles to maintain a consistent focus, often resulting in high similarity of features across different views. To enhance the robustness and spatial distribution of features over the three views, we introduce view-wise spatial attention (VSA) for each of the three planes (Figure 3). This mechanism enables each plane to better capture distinct perspectives, promoting more accurate and diverse feature representation across viewpoints(Figure 3).

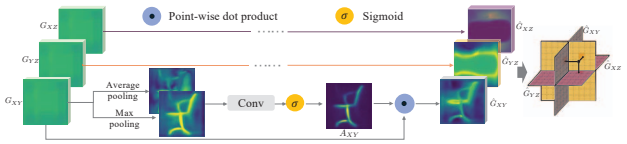

Figure 3. Illustration of proposed view-wise spatial attention (VSA) module. This module analyzes each plane individually, utilizing spatial features as guidance for attention.

The concept of attention in this context is well-documented in prior studies [13, 24]. Given that our planes are naturally divided into three distinct groups, we concentrate on learning attention that is specific to each plane, rather than applying the same attention across all planes. This view-specific attention tailors the focus to the unique aspects of each plane. For example, in Figure 3, our aim is to ensure the XY plane to capture an object's side view.

For each plane, this is achieved by initially aggregating the plane feature set along the channel axis via using two pooling (max, average) operations, generating two 2D maps. Subsequently, a localized convolutional layer is employed to derive the attention on the concatenation of two maps, which is then mapped to a weight value in the interval $[0, 1]$ via a Sigmoid function:

$$A_{XY} = \sigma\left(\text{Conv}\left(\text{mean}(\hat{G}_{XY}) \oplus \max(\hat{G}_{XY})\right)\right). \quad (2)$$

To produce the attention-aware feature planes, we apply element-wise multiplication between the attention map and the original feature plane:

$$\hat{G}_{XY} = A_{XY} \odot G_{XY}. \quad (3)$$

We apply VSR to each feature plane within G, and finally we get a new set of triplanes:

$$\hat{G} = \{\hat{G}_{XY}, \hat{G}_{YZ}, \hat{G}_{XZ}\}. \quad (4)$$

3.3. Reflectional Symmetry Regularization

Reflectional symmetry is a common trait in many object categories[16], including those under our study. To leverage this characteristic, we impose a reflection-symmetry regularization on the geometry triplane G. Consider a chair as an example, in its triplane representation, we impose reflectional symmetry along the XY-plane. We propose two variants to enforce such symmetry:

Feature symmetry: For the plane features G_{YZ} and G_{XZ}, which correspond to the front and down views, respectively, we enforce reflectional symmetry:

$$\mathcal{R}(G) = \|G_{YZ} - \mathit{flip}(G_{YZ})\|^2 + \|G_{XZ} - \mathit{flip}(G_{XZ})\|^2. \quad (5)$$

Attention symmetry: For the plane attention A_{YZ} and A_{XZ}, again corresponding to the front and bottom views, we enforce reflectional symmetry as follows:

$$\mathcal{R}(A) = \|A_{YZ} - \mathit{flip}(A_{YZ})\|^2 + \|A_{XZ} - \mathit{flip}(A_{XZ})\|^2. \quad (6)$$

While geometric symmetry is more naturally applicable to some categories, texture symmetry can also be beneficial [15]. We implement a methodology where each pixel's features are combined with those of its symmetrical counterpart:

$$\mathcal{R}(T_{(x,y,z)}) = T_{(x,y)} + \frac{T_{(y,z)} + T_{(y,-z)}}{2} + \frac{T_{(x,z)} + T_{(x,-z)}}{2}. \quad (7)$$

This technique is designed to efficiently capture and represent the object's features. Rather than enforcing symmetry across all regions, we focus specifically on the vertices of the generated mesh, ensuring that attention is not unnecessarily diverted to unrelated areas.

3.4. Training Objectives

We follow GET3D [10] and use the standard adversarial loss for training following:

$$\begin{aligned} L_D &= -\mathbb{E}[\log(1 - \mathrm{D}(I_f))] - \mathbb{E}[\log(D(I_r))] \\ &\quad + \lambda \mathbb{E}\left[\|\nabla_{I_r} D(I_r)\|^2\right], \\ L_G &= -\mathbb{E}[\log(D(I_f))] + \alpha\mathcal{R}(G) + \beta\mathcal{R}(A), \end{aligned} \quad (8)$$

where I_r represents real data while I_f denotes data produced by the generator (*i.e.* RGB image, silhouettes). The third element in Eq 8 is the gradient penalty, with λ as the weighting coefficient for this term. The hyperparameters α and β control the symmetry constraint on the geometry triplane and the attention maps, respectively. We have also adopted the shape regularization utilized in GET3D, which helps eliminate internal floating faces. Our discriminator does not require the camera pose of real images, which is inaccessible in most real-world datasets. Overall, by combining symmetry prior with other losses, our method produces realistic images, not just strictly symmetrical ones, making it suitable for a wide range of categories.

Dataset	Method	COV (%, ↑)		MMD (↓)		FID (↓)
		LFD	CD	LFD	CD	3D
Chair-S1	OP3D [11]	24.4	0.09	4183	34.05	60.82
	GET3D [10]	**66.31**	15.03	3412	14.98	55.17
	SYM3D	64.58	**58.53**	**3227**	**4.42**	**38.23**
Chair-S2	OP3D [11]	32.18	0.18	4764	11.8	74.83
	GET3D [10]	**67.29**	15.38	3754	14.92	63.35
	SYM3D	67.02	**56.32**	**3563**	**4.74**	**51.18**
Chair-S3	OP3D [11]	27.76	0.09	4853	15.50	76.40
	GET3D [10]	**63.00**	16.00	3837	15.52	66.51
	SYM3D	62.86	**53.14**	**3661**	**4.94**	**56.59**

Dataset	Method	COV (%, ↑)		MMD (↓)		FID (↓)
		LFD	CD	LFD	CD	3D
Car-S1	OP3D [11]	29.24	0.07	2593	13.48	34.23
	GET3D [10]	58.30	28.25	1461	1.42	29.69
	SYM3D	**63.35**	**36.13**	**1284**	**1.21**	**23.07**
Car-S2	OP3D [11]	15.69	0.07	3404	18.5	41.49
	GET3D [10]	55.96	19.88	1731	1.77	34.60
	SYM3D	**65.46**	**32.86**	**1708**	**1.50**	**31.35**
Car-S3	OP3D [11]	12.43	0.13	3737	20.67	48.19
	GET3D [10]	47.79	15.86	1747	1.80	36.39
	SYM3D	**50.13**	**19.88**	**1709**	**1.69**	**32.81**

Table 1. Quantitative results on ShapeNet-Chair and ShapeNet-Car. The best result is shown in **bold**, and MMD-CD scores are multiplied by 10^3.

4. Experiment

4.1. Settings

Datasets. We conduct experiments on the synthetic ShapeNet [6] dataset and the real-world Amazon Berkeley Objects (ABO) [9] dataset. Following GET3D [10], we focus on car and chair categories from ShapeNet, where each object is represented by 24 different views. As suggested by [10], we split each category into training (70%), validation (10 %), and testing (20%) sets. To simulate a more realistic training scenario similar to natural images, our approach only uses a *single view* of each object in the training set. We define three scenarios based on the range of azimuth angles for the selected view: Scenario 1 (S1) spans 0-360 degrees, Scenario 2 (S2) covers 0-180 degrees, and Scenario 3 (S3) encompasses 0-120 degrees. For S2 and S3, we adopt random view flips to enhance view diversity. For ABO datasets, we run experiments on its chair category, which has 1158 objects. All experiments are conducted at a resolution of 1024×1024.

Competitors. To the best of our understanding, SYM3D marks the initial attempt to integrate a symmetry prior into a triplane representation. While, GET3D is unique in its approach to separate geometry and texture during the training of a 3D generative model, making it the benchmark in this field. As such, GET3D serves as our primary reference point. We also draw comparisons with another SOTA work in 3D-aware image synthesis, OrthoPlanes for 3D (OP3D) [11], which enhances the triplane representation by maintaining information related to the projection distance.

Implementations. In real-world settings, accurately determining camera poses can be difficult. As a result, we adopt a strategy of training the discriminator without camera pose condition, opting instead for a fixed camera distribution approach, as demonstrated to be effective in previous studies [10, 18]. For all our experiments, we utilize the camera distribution defined in GET3D. It's important to mention that the camera distribution is not completely covered in S3. For hyper-parameters, we set $\alpha = 100$, $\beta = 10$ in the experiments. Additionally, we adopt the same setup of [10] including the training configuration as given in its open source code[1]. Experiments are done on 8 A100 GPUS.

Metrics. To assess the quality of the generated objects, we examine both their geometry and texture. For geometric, we use Coverage (COV) and Minimum Matching Distance (MMD) [1] metrics, which assess the average quality and diversity of the shapes. The comparison between two shapes is using the Chamfer Distance (CD) and the Light Field Distance (LFD) [8], measuring their similarity. For OP3D [11], we use marching cubes to extract the underlying geometry. For texture, we utilize the Fréchet Inception Distance (FID) [12], which is applied to 2D rendered images of the objects.

4.2. Main Results

Quantitative Results. The quantitative results for the ShapeNet chairs and cars are presented in Table 1. Our observations are as follows: **(1)** In comparison to OP3D [11], which utilizes a single triplane representation to encode both shape and texture, GET3D [10] and our SYM3D, which separate shape and texture learning, exhibit improved generation in quality and diversity. **(2)** Models trained on various splits show that S1 outperforms S2, which in turn outperforms S3. Specifically, S1 encompasses a full 360-degree view of an object category, S2 achieves a 360-degree view through image flipping, while S3 is limited to a 240-degree view. The results further suggest that incorporating views from various angles, including those from other objects, enhances the learning of a 3D representation. **(3)** All models are trained without conditioning discriminator on camera poses. Our approach, which incorporates symmetry assumed in a canonical space, yields superior results in both COV and MMD metrics. This indicates that the shapes

[1] https://github.com/nv-tlabs/GET3D

Figure 4. Qualitative comparison of SYM3D against OP3D and GET3D on generated images. SYM3D produces images with sharp details and a high diversity of shapes.

(a) Trained on ShapeNet Chair-S1 (b) Trained on ShapeNet Car-S1

Figure 5. Rendered RGB images across different camera views.

learned by SYM3D closely resemble those in a canonicalized form. **(4)** SYM3D outperforms GET3D in majority metrics, demonstrating the benefits of integrating symmetry regularization and view-wise spatial attention into 3D representations. Consequently, SYM3D proves to be an effective approach that enhances 3D representation.

Qualitative Comparisons. Figure 4 provides qualitative comparisons against competitors in terms of generated 2D images. In general, SYM3D achieves a more realistic appearance across different settings. In the more challenging settings S2, S3, and real-world ABO-chair, OP3D struggles to generate a complete scene, and GET3D tends to generate distortions. Figure 5 provides comparisons across various camera positions. GET3D and SYM3D are able to generate view-consistent images across different camera views but OP3D fails to do that. This demonstrates that decoupling shape and texture benefits the 3D-aware generation.

Since both GET3D and SYM3D generate textured meshes, we export their shapes into Blender and show their comparison in Figure 6. SYM3D significantly outperforms GET3D in creating textured meshes. It consistently delivers more regular and higher fidelity representations across various objects. While GET3D struggles with generating a complete shape, often producing armchairs with missing halves, uneven chair backs, broken carriage, and other irregularities. In contrast, our method provides uniform and complete shapes. This advantage becomes even more apparent when working with incomplete views (S3); GET3D is prone to creating fragmented shapes due to the absence of certain viewpoints. SYM3D shows that with symmetry as a structural prior, the generator can learn to produce complete and accurate shapes, even when trained on datasets with limited views. Additionally, when applied to real-world datasets, the shortcomings of GET3D become evident through the creation of chairs with unrealistic features, such as five legs or irregularly shaped supports, highlighting the effectiveness of our method.

4.3. Properties of Learned Triplane

View-wise Triplanes. We use a similarity matrix to measure difference among feature maps in the geometry triplane. The geometry triplane is a $N \times N \times 3C$ feature tensor, which we flatten to a 2D tensor $3C \times N^2$.

Subsequently, we compute the similarity matrix for this 2D tensor. Each entry within this matrix is a cosine similarity between the two channels. The rendered 2D images, similarity matrix, along with 3 selected feature maps from XY, YZ, and XZ planes from both GET3D (*i.e.* G_{XY}, G_{YZ}, G_{XZ}) and SYM3D ($\hat{G}_{XY}$,$\hat{G}_{YZ}$,$\hat{G}_{XZ}$), are visualized in Figure 7. This comparison reveals that our method produces planes more specific to each view, demonstrating increased similarity within each plane but high discrimina-

Figure 6. Comparison on shapes generated by GET3D and SYM3D rendered in Blender. SYM3D significantly outperforms GET3D in creating textured meshes. It consistently delivers more regular and higher fidelity representations across various objects. In contrast, GET3D often produces armchairs missing halves, uneven chair back, broken carriage, and other irregularities.

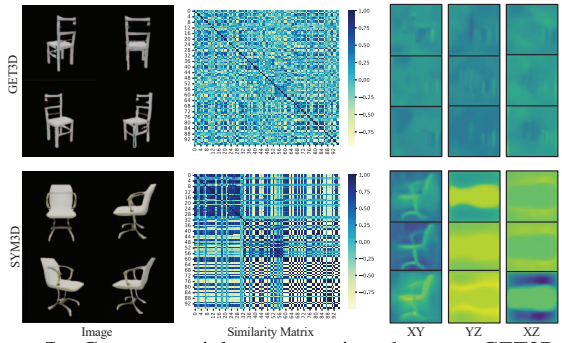

Figure 7. Geometry triplane comparison between GET3D and SYM3D. From left to right: rendered 2D image, similarity matrix across channels, feature maps from each plane. We note that SYM3D displays enhanced view-wise properties.

tion across different planes. Furthermore, the XY plane in our method clearly exhibits a side view pattern, offering a more distinct representation. Although the YZ, XZ planes do not capture front and down views in a precise way, they still adhere to the symmetry assumption. We argue that the enforced symmetry regularization plays a crucial role in driving a clearer representation in the XY plane.

Robustness to Biased Views. Figure 1 shows the results from models trained on the incomplete view dataset: ShapeNet Chair-S3. Overall, SYM3D shows its superiority in generating complete objects.

Consistent Camera Orientations. The fixed camera distribution strategy ensures that all objects produced by both GET3D and SYM3D are uniformly positioned and oriented. By applying symmetry regularization, our method secures a consistent orientation for objects in the chair category with respect to the symmetry plane (Figure 8).

Figure 8. Images rendered from different models given a fixed camera view.

4.4. Further Analysis

The effect of different components. We validate the design of our framework by ablating four components using the ShapeNet Chair-S1 dataset (in Table 2). SVE introduces view-wise spatial enhancement in Section 3.2, $\mathcal{R}(G)$ adds symmetry in feature maps as in Eq. 5, $\mathcal{R}(A)$ adds symmetry in attention maps as in Eq. 6, $\mathcal{R}(T)$ supplements a point's feature with its symmetric counterpart as in Eq. 7. Our findings indicate that each component independently improves the quality of generation, with the collective implementation of all elements resulting in the most effective model.

VSA	$\mathcal{R}(G)$	$\mathcal{R}(A)$	Tex	FID
✗	✗	✗	✗	55.17
✗	✓	✗	✓	47.87
✓	✗	✓	✓	43.43
✓	✓	✓	✗	41.09
✓	✓	✓	✓	**38.23**

Table 2. ablation studies

With window self-attention. To show the effectiveness of the proposed attention module, we compare SYM3D against a variant that uses window self-attention [17] across

three views, without employing attention symmetry. Considering memory limitations, we chose two window sizes: Wins=4 and Wins=8. Table 3 indicates that SYM3D, while adding negligible increases in parameters and computational cost, results in significant improvements.

Method	Added Params	Added Flops	FID
GET3D [10]	0	0	55.17
WinS=4	4616	1.0G	49.83
WinS=8	6024	1.6G	46.73
SYM3D	294	19.3M	**38.23**

Table 3. Comparison using the self-attention module, tested on ShapeNet Chair-S1.

On a more complex category. For evaluating the generality of proposed method on more complex categories, we test on airplane dataset in ShapeNet [6]. We render 24 views for each object and select one view as training set. As shown in Table 4, we obtain consistent results as on ShapeNet-Chair and ShapeNet-Car in Table 1.

Method	COV (%, ↑)		MMD (↓)		FID (↓)
	LFD	CD	LFD	CD	3D
GET3D [10]	53.73	7.17	4980	5.37	40.66
SYM3D	**57.85**	**49.07**	**4074**	**1.15**	**32.61**

Table 4. Generality on a more complex category. Experiments are done on ShapeNet Airplane.

5. Applications

Applied on the text-to-3D task. Current text-to-3D task usually utilize text-to-image generation models as guidance. The training process distills one image at a time, facing challenges in maintaining consistent views and precise geometry, leading to texture misalignments, asymmetry, incoherent appearances, or severe "Janus effect" issues where features like faces or eyes appear repeatedly and unnaturally [3] in generated object. Symmetry assumption encodes information from unseen views, providing a global constraint that defines a canonical frame, crucial for accurate model alignment and generation.

To demonstrate the generalizability of our method, we have applied our proposed symmetry regularization to this task. MTN [28] introduces a multi-scale triplane representation for the text-to-3D task. We apply symmetric regularization (see Eq 5) to these multi-scale triplanes. We develop two versions: W/O SYM and W SYM, whose results are in Figure 9. Our observations show that W/O SYM often generates salient artifacts, such as extra or malformed legs and ears when modeling a cat, and a dog with three forelegs. Conversely, W SYM produces cats and dogs without these defects, proving that symmetric priors effectively eliminate such artifacts in 3D model creation.

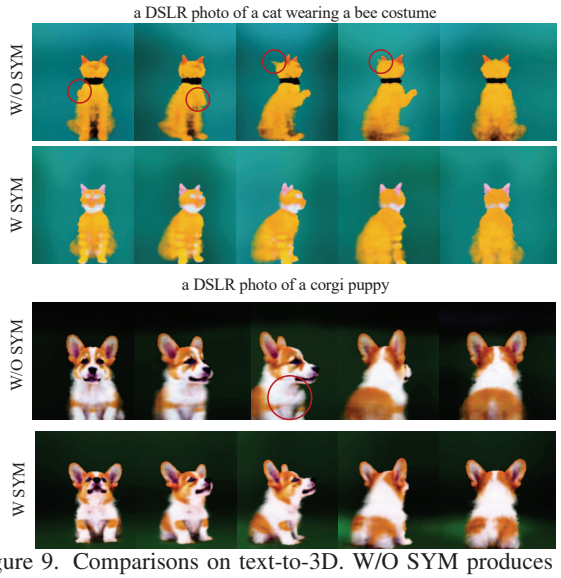

Figure 9. Comparisons on text-to-3D. W/O SYM produces artifacts in rendered images, while W SYM with symmetry regularization loss avoids this.

Applied to SSDNerf To demonstrate the versatility of our proposed method, we apply it to SSDNerf, a diffusion framework that also employs the triplane representation for 3D modeling. We follow the public repository to train vanilla SSDNerf and symmetric SSDNerf on random sampled single view from each scene. This framework requires supervised training with both camera poses and corresponding rendered images. Table 5 shows that the symmetry prior remains beneficial in this setting, yielding improved performance on both generation and single-view reconstruction tasks.

Method	Generation		1-View Reconstruction		
	FID↓	KID↓	PSNR↑	SSIM↑	LPIPS↓
SSDNerf	54.50	100.77	15.19	0.773	0.217
SSDNerf+SYM	**31.24**	**57.84**	**17.44**	**0.818**	**0.165**

Table 5. Comparison between SSDNerf and SSDNerf+SYM after training on single-view images, evaluated on Generation and Reconstruction tasks.

6. Conclusion

We present SYM3D for learning symmetric triplanes for improving 3D-awareness of GANs when trained on single images without camera pose annotation. While effective it has several **limitations** that future work should investigate. Global reflectional symmetry is not always satisfied and some objects satisfy other geometric transformations symmetry such as reflections, translations, rotations, or combinations in local parts (*i.e.* car tires, chair legs). In future work, we plan to extend our approach to create a large real dataset of common object categories and combat the canonicalization issue as in [2] and symmetry issue as in [19].

References

[1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3d point clouds. In *ICML*, 2018. 5

[2] Rohith Agaram, Shaurya Dewan, Rahul Sajnani, Adrien Poulenard, Madhava Krishna, and Srinath Sridhar. Canonical fields: Self-supervised learning of pose-canonicalized neural fields. In *CVPR*, 2023. 8

[3] Raphael Bensadoun, Yanir Kleiman, Idan Azuri, Omri Harosh, Andrea Vedaldi, Natalia Neverova, and Oran Gafni. Meta 3d texturegen: Fast and consistent texture generation for 3d objects. *arXiv preprint arXiv:2407.02430*, 2024. 8

[4] Ananta R. Bhattarai, Matthias Nießner, and Artem Sevastopolsky. Triplanenet: An encoder for eg3d inversion. In *WACV*, 2024. 2

[5] Eric R Chan, Connor Z Lin, Matthew A Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas J Guibas, Jonathan Tremblay, Sameh Khamis, et al. Efficient geometry-aware 3d generative adversarial networks. In *CVPR*, 2022. 1, 2, 3

[6] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv*, 2015. 2, 5, 8

[7] Hansheng Chen, Jiatao Gu, Anpei Chen, Wei Tian, Zhuowen Tu, Lingjie Liu, and Hao Su. Single-stage diffusion nerf: A unified approach to 3d generation and reconstruction. In *ICCV*, 2023. 1

[8] Yanqin Chen, Xin Jin, and Qionghai Dai. Distance measurement based on light field geometry and ray tracing. *Optics express*, 2017. 5

[9] Jasmine Collins, Shubham Goel, Kenan Deng, Achleshwar Luthra, Leon Xu, Erhan Gundogdu, Xi Zhang, Tomas F Yago Vicente, Thomas Dideriksen, Himanshu Arora, Matthieu Guillaumin, and Jitendra Malik. Abo: Dataset and benchmarks for real-world 3d object understanding. In *CVPR*, 2022. 2, 5

[10] Jun Gao, Tianchang Shen, Zian Wang, Wenzheng Chen, Kangxue Yin, Daiqing Li, Or Litany, Zan Gojcic, and Sanja Fidler. Get3d: A generative model of high quality 3d textured shapes learned from images. In *NeurIPS*, 2022. 1, 2, 3, 4, 5, 8

[11] Honglin He, Zhuoqian Yang, Shikai Li, Bo Dai, and Wayne Wu. Orthoplanes: A novel representation for better 3d-awareness of gans. In *ICCV*, 2023. 2, 5

[12] 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. In *NeurIPS*, 2017. 5

[13] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *CVPR*, 2018. 4

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

[15] Xingyi Li, Chaoyi Hong, Yiran Wang, Zhiguo Cao, Ke Xian, and Guosheng Lin. Symmnerf: Learning to explore symmetry prior for single-view view synthesis. In *ACCV*, 2022. 1, 2, 4

[16] Niloy J Mitra, Mark Pauly, Michael Wand, and Duygu Ceylan. Symmetry in 3d geometry: Extraction and applications. In *Computer Graphics Forum*, 2013. 1, 2, 4

[17] Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jon Shlens. Stand-alone self-attention in vision models. *NeurIPS*, 32, 2019. 7

[18] Katja Schwarz, Yiyi Liao, Michael Niemeyer, and Andreas Geiger. Graf: Generative radiance fields for 3d-aware image synthesis. In *NeurIPS*, 2020. 1, 5

[19] Ahyun Seo, Byungjin Kim, Suha Kwak, and Minsu Cho. Reflection and rotation symmetry detection via equivariant learning. In *CVPR*, 2022. 8

[20] Tianchang Shen, Jun Gao, Kangxue Yin, Ming-Yu Liu, and Sanja Fidler. Deep marching tetrahedra: a hybrid representation for high-resolution 3d shape synthesis. In *NeurIPS*, 2021. 3

[21] J Ryan Shue, Eric Ryan Chan, Ryan Po, Zachary Ankner, Jiajun Wu, and Gordon Wetzstein. 3d neural field generation using triplane diffusion. In *CVPR*, 2023. 1

[22] Ivan Skorokhodov, Aliaksandr Siarohin, Yinghao Xu, Jian Ren, Hsin-Ying Lee, Peter Wonka, and Sergey Tulyakov. 3d generation on imagenet. In *ICLR*, 2023. 1

[23] Alex Trevithick, Matthew Chan, Michael Stengel, Eric Chan, Chao Liu, Zhiding Yu, Sameh Khamis, Manmohan Chandraker, Ravi Ramamoorthi, and Koki Nagano. Real-time radiance fields for single-image portrait view synthesis. *ACM TOG*, 2023. 2

[24] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module. In *ECCV*, 2018. 4

[25] Rundi Wu and Changxi Zheng. Learning to generate 3d shapes from a single example. *ACM TOG*, 2022. 2

[26] Shangzhe Wu, Christian Rupprecht, and Andrea Vedaldi. Unsupervised learning of probably symmetric deformable 3d objects from images in the wild. In *CVPR*, 2020. 1, 2

[27] Yifan Xu, Tianqi Fan, Yi Yuan, and Gurprit Singh. Ladybird: Quasi-monte carlo sampling for deep implicit field based 3d reconstruction with symmetry. In *ECCV*, 2020. 1, 2

[28] Han Yi, Zhedong Zheng, Xiangyu Xu, and Tat-seng Chua. Progressive text-to-3d generation for automatic 3d prototyping. *arXiv*, 2023. 8

[29] Fei Yin, Yong Zhang, Xuan Wang, Tengfei Wang, Xiaoyu Li, Yuan Gong, Yanbo Fan, Xiaodong Cun, Ying Shan, Cengiz Oztireli, et al. 3d gan inversion with facial symmetry prior. In *CVPR*, 2023. 1, 2