MVGBench: a Comprehensive Benchmark for Multi-view Generation Models

Supplementary Material

In this supplementary, we first discuss the implementation details, including our MVGBench metrics (Sec. 6.1) and evaluation experiment setups (Sec. 6.2). We then show additional evaluation result and analysis in Sec. 7, and conclude with discussion of limitations.

6. Implementation Details

We discuss the details of our metrics and experiment setups. Our benchmark suite and pre-trained models will be publicly released.

6.1. MVGBench Metric Implementation

View sets split. For 3D consistency metric, we split the generated multi-view images into two subsets and fit 3DGS separately to them. We allow small overlap when the total number of generated views is small. There are three different number of output views for all the methods we evaluated: 16, 18, 21, see Tab. 5. The view indices for the two subsets are: 1). Output 16 views: [0, 2, 4, 5, 6, 8, 9, 10, 11, 12, 14], [1, 3, 5, 6, 7, 9, 11, 12, 13, 14, 15]. 2). Output 18 views: [0, 1, 2, 4, 6, 8, 10, 12, 14, 16], [0, 1, 2, 3, 5, 7, 9, 11, 13, 15, 17]. 3). Output 21 views: the first (input) view is excluded and rest is divided into two non-overlap views, namely [0, 2, 4, 6, 8, 10, 12, 14, 16, 18], [1, 3, 5, 7, 9, 11, 13, 15, 17, 19].

3DGS optimization. We use the original version of 3DGS [27] for optimization. We explored more advanced version of 3DGS but found that they are either less accurate for object level multi-views [19, 41, 78] or the runtime is too long [79]. We hence stick to the original 3DGS version and randomly sample 100k points from unit cube [-1, 1] to initialize the Gaussians and optimize for 10k steps. White background is used during optimization as all methods generate images with white background.

Test view rendering. We render the optimized 3DGSs into RGB and depth images to compute the depth, cPSNR, cSSIM, cLPIPS metrics. To produce comparable numbers, the test views have to be the same for two 3DGSs and across all methods, for the same test object. The test views should be diverse so that is does not favor output elevation angles specific to some methods while it should also be close to the views used to fit 3DGS, otherwise the calculated scores are dominated by 3DGS fitting error instead of multi-view inconsistency. To this end, we use two setups to choose the views for rendering: a). Random views sampled from a fixed range, and b). Fixed views that differ 15 elevation degrees from generated multi-views. For both setup $K = 16$ views are used for rendering, and each object might have different test views but the same views are always used across methods for fair comparison.

Random test views are used for best setup performance, robustness w.r.t to lighting and azimuth conditions. As existing methods generate multi-view images with elevation angle ranging from 0 to 30 degrees (Tab. 5), we uniformly sample elevation from range [-15, 45], azimuth from [0, 360], and camera distance from [1.5, 1.9]. The field of view (Fov) is fixed at 42 degree such that it does not favor any of the methods evaluated.

Fixed test views are used for generalization to real images and robustness w.r.t to different elevation degrees. In these setups, the output elevation differ a lot and it is difficult to define a common range where 3DGS fitting also works well and we can sample elevation from. To this end, we take 8 views with equal azimuth distance from 8.5 to 360 degree and the elevation is 15 degree higher than the elevation of generated multi-views. The other 8 views have the same azimuth angles but the elevation is 15 degree lower than the output multi-view elevation. The fov and camera distance are fixed to 42 degree and 3.2m. We choose these azimuth, fov, and distance to not favor any methods. Note that this will lead to consistency scores that are not comparable across different output elevations, which address next.

Normalization of the consistency scores. The exact scores of our consistency metrics depend on the views used to render test images and the raw numbers are not directly comparable if the views are different. This is the case when we want to evaluate the robustness of a method w.r.t to different elevation angles (see discussion above). We hence propose to normalize the raw numbers using the upper bound scores obtained from ground truth multi-view images. Let $e_{\text{gt}}, e_{\text{mvg}}$ be the raw consistency score defined in Sec. 3.1 using MVG and GT images of the same camera views. The normalized error $e_{\text{mvg, n}}$ is computed as:

$$e^i_{\text{mvg, n}} = \begin{cases} \frac{e_{\text{mvg}}}{e_{\text{gt}}} & \text{if type}(e) \in \{\text{cPSNR, cSSIM}\}, \\ 1 - \frac{e_{\text{mvg}}}{\max e_{\text{mvg}}} & \text{if type}(e) \in \{\text{CD, depth, cLPIPS}\}. \end{cases} \tag{5}$$

here $\max e_{\text{mvg}}$ is the maximum error for this metric among our evaluated methods. This yields a score between 0 and 1 and it is always the higher the better. This normalization is also used to visualize the bottom plots in Fig. 1.

Prompt templates for VLM based metrics. We propose four metrics based on the pretrained 73B InternVL2.5 VLM [8]. The same model is used to obtain the reference attributes (Sec. 3.2) via three-round prompts given multi-view images. The three sequential prompts are: 1). "Here are images of a daily object, what is the appearance style of this object? Ignore the background, focus on the appearance, style and design instead of describing the object type, return the appearance style only and in less than 5 words.", 2). "Which object it is? Just return the class name, do not repeat question. Use daily used common words. If there are multiple possibilities, return like this: classname 1 or classname2 or classname3...", 3). "What is the main color(s) of this object? simply answer the color(s), summarize to less than 4 colors."

We then use the answers from these prompts as the reference attributes and evaluate the semantic consistency using the following templates: 1). `class`: "Is [obj cls] presented in this image? just answer yes or no." 2). `color`: "Does the object (possibly [obj cls]) shown in this image have the color(s): [color]? just answer yes or no. 3). "Is the appearance style of the object (possibly [obj cls]): [style]? just answer yes or no."

We also use the same model to asses the image quality (IQ-vlm) which we find align well with human perception. The prompt template is: "Is this image an overall high-quality image with good overall structure, good visual quality, nice color harmony, clear object and free of strange artifacts and distortions? just answer yes or no.".

Runtime performance. The most compute expensive steps in our evaluation pipeline are 3DGS optimization and VLM assessment, which takes around 76s (two subsets) and 12s per input image to finish on L40s GPU. In total it takes around 2.7 hours to evaluate 100 objects which is still reasonable. More advanced techniques such as better 3DGS initialization [9] or VLM inference via API call could be adopted to speed up evaluation. We leave these for future works.

6.2. Experiment Setups

User study. We conduct user studies to verify our oFID score and VLM based metrics, each with 400 questions answered by 10 users. As 400 questions are too many for one single user study survey, we divide it into 8 smaller surveys, each with 50 questions. We then recruit 10 users to finish each survey and no overlap is allowed for different surveys. Hence in total we have 80 different users to participate one user study. This ensures sufficient diversity and statistically meaningful results. We show example questions from our user studies in Fig. 7 and Fig. 8.

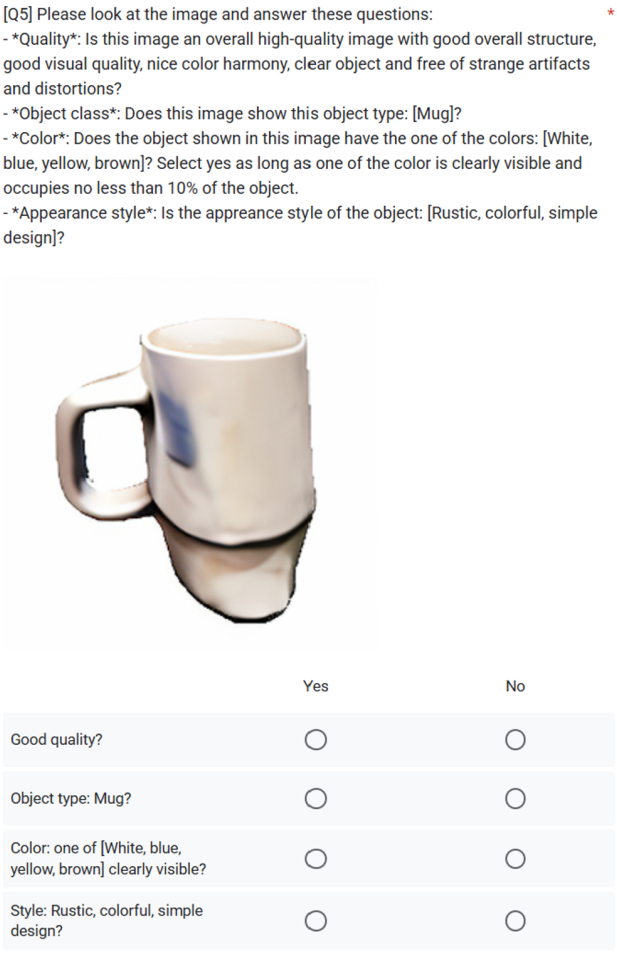

Figure 7. Example question from our user study on the alignment between our VLM based metrics and human preference.

Evaluation setup for existing methods. We show the input and output setups for all the methods we evaluate in the **best-setup performance** experiment in Tab. 5. We use ambience light of 1.0 and zero azimuth angle for this setup. The rendering setup is the same for **robustness evaluation** except for the attribute we want to evaluate (elevation, light intensity, and azimuth angle). For **generalization to real images**, we cannot control the rendering anymore hence we use the same input image crop for different methods, which has 0.2 margin from the object bounding box to image boarder. The number of output views of each method remain the same as in best-setup evaluation.

MVG design choice experiments. We use the 150k kiui objects filtered by LGM [52] as our training data. Following same camera parameters in SV3D [54], we render each object from 84 views and randomly pick 21 views at training time. We adopt the dynamic orbit rendering from SV3D which adds perturbations of azimuth and elevation angles to the equally distributed static orbit. We pre-compute the la-

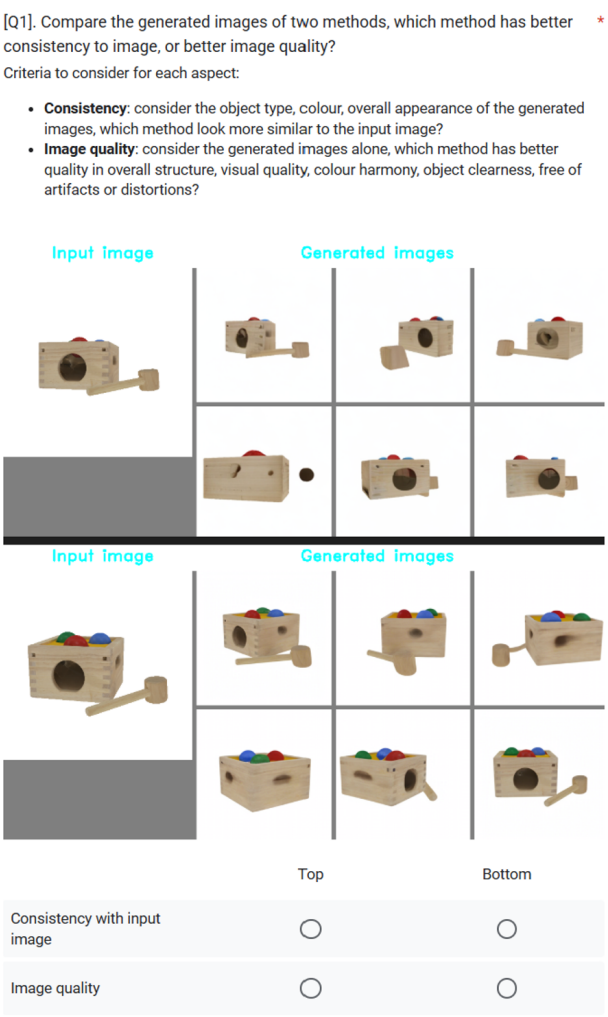

Figure 8. Example question from our user study on the alignment between our oFID score and human preference.

tent features of CLIP [44], SVD [46], DINOV2 [42] and ConvNextV2 [59] to speed up training. We use batch size of 64, learning rate of 2e-5 for all the experiments. The total training steps is 26k for camera embedding experiments and 50k for all other experiments.

7. Additional Results and Analysis

Validation of VLM based metrics.

Full evaluation results. We show all scores of our MVGBench from all evaluated methods on four datasets in Tab. 6 (GSO [12]), Tab. 7 (Omni3D [63]), Tab. 8 (CO3D [45]), and Tab. 9 (MVImgnet [77]). It can be seen that our method achieves the best overall 3D consistency and on par performance on image quality and semantic consistency.

Methods struggle with fine-grained details. We rank the input images based on the 3D consistency score (cPSNR)

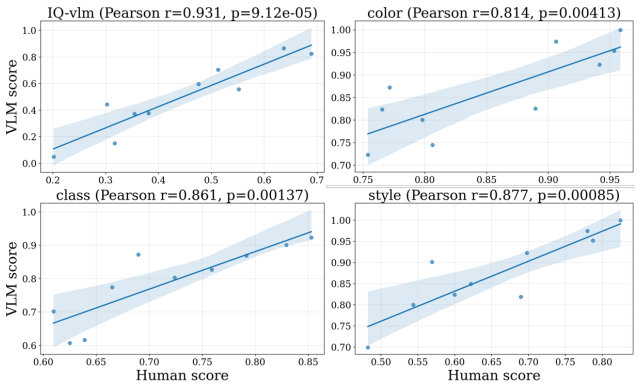

Figure 9. **Validating vision language model (VLM) based metrics.** Our VLM metrics strongly correlate with human perception (Pearson coefficient confidence interval: 0.95).

Methods	Fov	Elev.	Dist.(m)	#Out views
SyncDreamer based [17, 21, 38, 74, 82]	49.1	30	1.5	16
V3D [9]	60	0	2.0	18
SV3D [54], ours	33.8	12.5	2.35	21
Zero123 based [2, 11, 28, 30, 76]	49.1	0	1.85	21

Table 5. The input rendering (Fov, camera elevation degree and distance) and number of output views of each method for the best performance evaluation.

from different methods and visualize the 10 common inputs that have the worst scores in Fig. 10. It can be seen that the common challenging images are the objects with complex and fine-grained geometric structures and textures such as bicycle, flowers, text boxes. Diving further into this problem, we find that the autoencoder used in all MVGs already destroys the high frequency structures after one single pass through the autoencoder. We show two examples in Fig. 11. To further understand the effect on 3D consistency, we send the ground truth multi-view images of 30 GSO objects [28] through the autoencoder of SV3D [54]. The consistency scores before and after the autoencoder are (CD / depth / cPSNR / cSSIM / cLPIPS): 2.58 / 9.82 / 31.94 / 0.94 / 0.03 (before), 2.69 / 9.05 / 30.29 / 0.92 / 0.04 (after). It can be clearly seen that the 3D texture consistency scores already degrade after single feedforward through the autoencoder. Interestingly, the depth error, sensitive to inconsistency in edges, decreases which indicates the images are indeed smoother with high-frequency details lost.

8. Limitations and Future Work

We present the first comprehensive benchmark to evaluate 3D consistency of object multi-view generation models. Despite robust to various settings, there are still limitations of our benchmark. First, our method cannot evaluate methods that generate very few views (<10) as the 3DGS

Method	Geometry consistency		Texture consistency			Image quality		Semantic consistency		
	CD ↓	depth ↓	PSNR ↑	SSIM ↑	LPIPS ↓	FID ↓	IQ-vlm ↑	class ↑	color ↑	style ↑
Ours	3.15	**14.20**	**28.93**	**0.90**	**0.05**	20.46	0.82	0.86	0.94	0.93
SyncDreamer[38]	**2.99**	17.29	26.83	0.87	0.07	22.72	0.53	0.84	0.96	0.94
SV3D-tune	3.34	16.49	27.71	0.88	0.06	19.06	0.80	0.89	0.95	**0.96**
SV3D[54]	3.47	19.65	26.75	0.86	0.07	21.31	0.77	0.85	0.92	0.93
Hi3D[74]	3.29	21.69	24.60	0.84	0.09	18.68	**0.87**	**0.89**	0.95	0.95
V3D[9]	4.25	28.08	23.84	0.81	0.12	21.20	0.77	0.86	0.96	0.91
EscherNet[28]	4.34	20.61	23.89	0.79	0.11	24.71	0.57	0.77	0.90	0.88
MVDFusion[17]	4.77	38.74	21.44	0.76	0.15	25.60	0.48	0.88	0.94	0.94
ViewFusion[76]	5.33	40.20	22.34	0.80	0.14	22.03	0.63	0.82	0.92	0.92
EpiDiff[21]	5.77	50.65	20.28	0.72	0.19	**16.53**	0.77	0.89	**0.97**	0.94
Free3D[82]	6.03	44.27	20.26	0.77	0.18	27.30	0.73	0.78	0.82	0.90
Vivid123[30]	7.57	43.97	21.74	0.81	0.18	38.91	0.63	0.66	0.78	0.80
Zero123[2]	10.99	63.72	17.37	0.67	0.29	21.35	0.73	0.82	0.90	0.93
Zero123-xl[11]	15.40	68.13	17.10	0.66	0.30	20.72	0.72	0.83	0.91	0.94

Table 6. Best setup performance on the GSO [12] dataset.

Method	Geometry consistency		Texture consistency			Image quality		Semantic consistency		
	CD ↓	depth ↓	PSNR ↑	SSIM ↑	LPIPS ↓	oFID ↓	IQ-vlm ↑	class ↑	color ↑	style ↑
Ours	2.98	**11.63**	**29.09**	**0.92**	**0.04**	15.47	**0.54**	0.77	0.85	**0.88**
SyncDreamer[38]	**2.93**	13.60	27.24	0.89	0.06	5.94	0.24	0.64	0.85	0.79
Hi3D[74]	3.13	17.63	25.25	0.88	0.08	16.07	0.55	0.74	0.87	0.84
SV3D-tune	3.16	14.11	27.69	0.91	0.05	15.00	0.51	0.79	0.88	0.89
SV3D[54]	3.46	19.46	26.02	0.88	0.07	17.60	0.50	0.69	0.85	0.85
V3D [9]	4.51	23.62	23.01	0.85	0.12	17.70	0.46	0.70	0.84	0.85
EscherNet[28]	5.01	23.25	21.87	0.77	0.14	22.39	0.41	0.60	0.80	0.79
MVDFusion[17]	5.67	47.96	19.04	0.76	0.19	26.89	0.21	0.69	0.83	0.82
EpiDiff[21]	6.78	57.31	18.37	0.73	0.21	**14.61**	0.52	**0.79**	**0.89**	0.88
ViewFusion[76]	7.88	54.32	17.90	0.73	0.24	16.96	0.44	0.68	0.85	0.87
Free3D[82]	8.02	52.58	16.97	0.72	0.25	23.78	0.50	0.61	0.77	0.79
Zero123-xl[11]	13.67	70.17	13.64	0.60	0.39	17.86	0.51	0.67	0.84	0.86
Zero123[2]	14.17	70.32	14.15	0.62	0.38	17.62	0.51	0.69	0.84	0.88
Vivid123[30]	14.31	56.07	17.80	0.76	0.26	27.98	0.50	0.56	0.74	0.74

Table 7. Best setup performance on the Omni3D [63] dataset.

Method	Geometry consistency		Texture consistency			Image quality		Semantic consistency		
	CD ↓	depth ↓	PSNR ↑	SSIM ↑	LPIPS ↓	FID ↓	IQ-vlm ↑	class ↑	color ↑	style ↑
Ours	3.10	16.94	**25.99**	**0.88**	**0.06**	23.40	0.29	0.80	**0.86**	0.82
SyncDreamer[38]	**3.04**	**13.48**	25.30	0.88	0.06	30.96	0.12	0.69	0.83	0.70
SV3D-tune	3.43	19.99	24.32	0.85	0.08	21.71	0.26	0.82	0.84	0.83
SV3D[54]	3.48	25.80	23.72	0.87	0.13	24.19	0.29	0.76	0.87	0.78
EscherNet[28]	5.14	26.46	20.34	0.71	0.14	28.54	0.26	0.71	0.79	0.72
Hi3D[74]	5.60	31.09	20.92	0.81	0.12	25.51	0.35	0.75	0.82	0.75
MVDFusion[17]	5.77	47.43	17.50	0.71	0.20	27.16	0.19	0.75	0.82	0.78
EpiDiff[21]	7.71	58.58	15.66	0.64	0.26	**20.58**	0.31	0.84	0.86	0.82
ViewFusion[76]	7.75	49.76	16.49	0.77	0.29	22.10	0.33	0.82	0.85	0.82
Vivid123[30]	9.81	56.38	15.31	0.69	0.29	35.89	**0.49**	0.70	0.76	0.72
V3D[9]	10.45	58.71	16.39	0.71	0.26	28.76	0.32	0.72	0.85	0.77
Free3D[82]	11.15	60.95	14.42	0.76	0.33	32.84	0.32	0.71	0.75	0.75
Zero123[2]	12.06	64.74	13.16	0.55	0.38	21.22	0.38	0.84	0.87	**0.86**
Zero123-xl[11]	12.58	66.99	12.97	0.54	0.38	20.83	0.34	**0.85**	0.86	0.84

Table 8. Evaluation results on the CO3D[45] dataset with manually selected front view and annotated elevation angles.

fitting is very inaccurate and fitting error instead of multi-view inconsistency dominates our consistency scores. One

Method	Geometry consistency		Texture consistency			Image quality		Semantic consistency		
	CD ↓	depth ↓	PSNR ↑	SSIM ↑	LPIPS ↓	FID ↓	IQ-vlm ↑	class ↑	color ↑	style ↑
Ours	3.04	17.58	**26.43**	**0.88**	**0.06**	22.10	**0.37**	**0.74**	**0.88**	0.84
SyncDreamer[38]	**2.87**	**15.35**	25.44	0.88	0.06	30.64	0.17	0.59	0.84	0.79
SV3D-tune	3.37	21.94	24.97	0.85	0.08	22.04	0.34	0.72	0.88	0.82
SV3D[54]	3.39	26.17	23.99	0.83	0.09	22.07	0.33	0.71	0.87	0.79
EscherNet[28]	5.35	29.55	20.31	0.72	0.15	25.78	0.32	0.66	0.82	0.78
Hi3D[74]	6.24	33.72	21.42	0.81	0.12	25.77	0.41	0.63	0.83	0.78
MVDFusion[17]	6.29	50.54	17.27	0.70	0.22	28.45	0.26	0.62	0.80	0.78
EpiDiff[21]	8.05	61.91	15.85	0.64	0.27	**20.21**	0.42	0.74	0.86	0.84
ViewFusion[76]	8.26	54.27	16.14	0.63	0.28	21.77	0.39	0.72	0.84	0.84
Free3D[82]	10.64	60.00	14.77	0.65	0.33	33.58	0.38	0.60	0.70	0.75
Vivid123[30]	10.67	58.06	15.81	0.69	0.30	35.84	**0.56**	0.56	0.68	0.75
V3D [9]	10.74	65.40	16.30	0.71	0.26	27.89	0.40	0.65	0.79	0.79
Zero123-xl[11]	12.04	67.08	13.45	0.55	0.38	20.51	0.41	0.74	0.85	**0.87**
Zero123[2]	12.11	66.76	13.61	0.56	0.38	20.83	0.42	0.74	0.85	0.86

Table 9. Evaluation results on the MVImgNet [77] dataset with manually selected front view and annotated elevation angles.

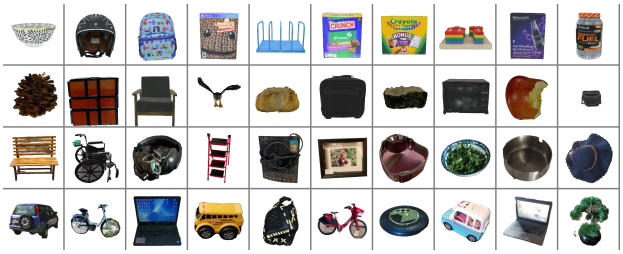

Figure 10. **The most challenging test images** from GSO[12], Omni3D[63], MVImgnet[77] and CO3D[45]. Methods produce the most 3D inconsistent images for these inputs due to their complex geometric structure or high frequency details.

Figure 11. Degradation of image quality after passing through the autoencoder of SV3D [54]. Clearly the high frequency details are destroyed by the autoencoder.

possible solution is to replace 3DGS fitting with pre-trained models that can take few-views as input and directly regress 3DGS, such as LGM [52]. This however requires the model to be robust to diverse camera setups which is still an ongoing research. Second, we curated four datasets which covers mainly daily objects, and most of them are indoor. It would be interesting to also consider outdoor objects such as buildings, statues or complex compositional shapes such as human-human or human-object interactions. Last but not least, we evaluate the robustness w.r.t lighting, elevation and azimuth angles. Real life objects have much more attributes that can affect the performance, such as the material, shading condition, specific object categories. One can do more comprehensive analysis could be done using our proposed metrics to understand the progress of SoTA methods. We leave these for future works.