A. Appendix Summary

In the appendix, we cover additional details of DDPO [5] and Reinforcement Learning Finetuning (RLFT) (Appendix B), training data details (Appendix C.1), training details (Appendix C.2), experiment details (Appendix C.3), user study details (Appendix C.4), additional Multi-view Reconstruction Consistency (MRC) metric experiments (Appendix D), ablation studies (Appendix E), additional related works (Appendix F), and broader impacts and future work (Appendix G).

B. Additional Details of DDPO and RLFT

B.1. Definitions

Following [5, 36, 45, 55], an *epoch* is defined as one round of data collection (*sampling*), which may consists multiple PPO [45] update steps (*training*), as discussed in Eq. (10) and Sec. 4.2. This definition of "epoch" is different from the meaning in supervised learning which usually refers to go through all data once. Since we opt for using pure on-policy training (vanilla policy gradient), as discussed in Sec. 4.2, we only do one training step per sampling step, and thus our sampling batch size and training batch size are equal.

B.2. Reward Normalization in DDPO

Specifically, the mean and standard deviation statistics of the rewards are tracked for each prompt c:

$$A_r(x_0, c) = \frac{r(x_0, c) - \mu_r(c)}{\sigma_r(c)} \tag{9}$$

DDPO's [5] reward-normalizing advantage replaces the value model that is more widely adopted in PPO-based [45] RLHF methods [36, 51, 55]. This is similar to the recent work [25], which shows that the value model creates unnecessary computation cost that can be replaced with a simpler advantage formulation.

B.3. DDPO$_{\text{IS}}$ Policy Gradient Function

By using the advantage term (Eq. (9)) in place of the reward, the DDPO$_{\text{IS}}$ policy gradient function is:

$$\begin{aligned}\hat{g}_{\text{IS}} = \mathbb{E}\Bigg[\sum_{t=0}^{T} &\frac{p_\theta(x_{t-1}|c,t,x_t)}{p_{\theta_{\text{old}}}(x_{t-1}|c,t,x_t)} \\ &\cdot \nabla_\theta \log p_\theta(x_{t-1}|c,t,x_t) A_r(x_0,c)\Bigg]\end{aligned} \tag{10}$$

where the expectation is taken over data generated by the policy $\pi_{\theta_{\text{old}}}$ with the parameters θ_{old}.

B.4. Computing KL Divergence

Following the widely adopted implementation in LLM RLHF [36, 55], we incorporate KL penalty into the reward function. Subtraction of the log probabilities is commonly used to approximate the full KL divergence [51, 55]:

$$\begin{aligned}&\text{KL}\left(\log p_\theta(x_0|c,T,x_T) || \log p_{\theta_{\text{base}}}(x_0|c,T,x_T)\right) \\ &= \sum_{t=0}^{T} \frac{\log p_\theta(x_{t-1}|c,t,x_t) - \log p_{\theta_{\text{base}}}(x_{t-1}|c,t,x_t)}{T+1}\end{aligned} \tag{11}$$

where $p_{\theta_{\text{base}}}$ is the base model. We will denote this approximated KL divergence term as $\text{KL}(x_0|c, x_T)$ for clarity in presentation.

B.5. Hypotheses on Stability and Sample Efficiency

Diverging from DDPO [5] and most Large Language Model (LLM) Reinforcement Learning from Human Feedback (RLHF) literature [2, 3, 36, 51, 55], we choose REINFORCE [54] (DDPO$_{\text{SF}}$) over PPO [45] (DDPO$_{\text{IS}}$) for its superior training stability. We provide two hypotheses behind our surprising finding.

(1) Training stability is more vital than sample efficiency when the task reward function is more challenging. When a reward function is more variant with respect to the model's output, it becomes more difficult for the model to discover the pattern of high-reward outputs and to improve its rewards. The high-variance prompt alignment reward curves in Fig. 5 of DDPO [5] indicates the challenging nature of the prompt alignment task as opposed to the smooth reward curves for the aesthetics and compressibility tasks in Fig. 4 of DDPO [5].

(2) The RL Finetuning (RLFT) sample efficiency is less important for a large model which requires less finetuning steps, as demonstrated in studies of LLM instruction finetuning [10]. Similarly, our RLFT on a 2.6B-parameter UNet from SDXL [39] only takes 55 epochs, as opposed to DDPO's [5] RLFT on a 860M-parameter UNet from SD 1.4 [42] using 200 epochs. Therefore, the potential sample efficiency gain provided by the multi-step update of PPO [45] gets outweighted by the training stability provided by REINFORCE [54].

The favorableness of REINFORCE [54] could apply to broader scenarios that fits these two conditions. We leave the verification of our hypotheses as future work.

C. Implementation Details

C.1. Training Data

An advantage of Reinforcement Learning Finetuning (RLFT) over Supervised Finetuning (SFT) is that, we can manually create a high-quality text prompts training set,

```python
def compute_mrc(ori_views, ori_cam_poses, lrm, lpips, resize_res):
    nerf = lrm(ori_views, ori_cam_poses)
    nerf_views = nerf.render(ori_cam_poses)
    square_bbox = compute_square_bbox(ori_views) # bounding box coordinates for each view
    x_min, y_min, x_max, y_max = square_bbox
    ori_views_bbox = [resize(o[:, y_min:y_max + 1, x_min:x_max + 1], resize_res) for o in
    ↪  ori_views]
    nerf_views_bbox = [resize(n[:, y_min:y_max + 1, x_min:x_max + 1], resize_res) for n in
    ↪  nerf_views]
    mrc = lpips(ori_views_bbox, nerf_views_bbox).mean()
    return mrc
```

Listing 1. Pseudo code for our MRC implementation. ori_views and ori_cam_poses are the multi-view images to be evaluated and their camera poses. lrm is the sparse-view LRM [17, 24]. lpips the the LPIPS [57] metric. resize_res is a fixed resolution to which we resize the bounding box patches.

while creating a dataset of diverse ground truth multi-view images for these high-quality text prompts is prohibitively expensive for SFT. By relying on samples generated by the model itself to compute the reward and the loss, RLFT can optimize a model beyond the limitation of a dataset and preserves the diversity and the style of the base model. In Carve3D, our training prompts preparation process involves two strategies.

Training Data Curation Instead of randomly sampling prompts from a dataset, we employ a data curation strategy where prompts with lowest rewards are selected. Specifically, we run inference of the base model on a prompt dataset, generating four results per prompt, compute the MRC rewards for each result, and sort the prompts according to their average reward. This is derived from observation that, for certain prompts, the model generates nearly optimal outputs with rewards close to the rewards of ground truth views of a 3D asset [12] (Fig. 8). Thus, the curated lowest-reward prompts have substantial room for 3D consistency improvement and prevent learning stagnation. This approach not only brings more efficient training but also provides a more generalized improvement in 3D consistency to the testing set.

Creating New Training Prompt Set The prompt dataset from DreamFusion [40], which contains 414 prompts and is commonly used as testing set. To employ the DreamFusion prompt set also as our testing set, we create a new prompt dataset with ChatGPT4 [35]. Following our training data curation strategy, we first sort the DreamFusion [40] prompts according to their rewards attained by the base Instant3D [24] model. We provide the sorted prompt set to ChatGPT4, and ask it to summarize the characteristics of the low-reward prompts by looking at the low-reward, median-reward, and high-reward prompts. ChatGPT4 summarizes low-reward prompts to possess properties of "complex and creative". We then ask it to generate 100 low-reward prompts that are both complex and creative, and another 100 low-reward prompts that are "complex but not too creative". For each set, again, we sort the prompts according to their rewards, and select those with the lowest rewards to be our training prompt set. Our best results are obtained with the "complex but not too creative" set.

C.2. Training Details

All of our RL finetuning experiments are run on 6 AWS EC2 P4de nodes with 8 NVIDIA A100-SXM4-80GB GPUs, a total of 48 GPUs. We use batch size of 768, which is 2x compared to that of DDPO. One experiment takes 16.5 hours to reach 55 epochs. The number of finetuning epochs is determined by our KL-divergence early-stopping mechanism, which we empirically choose to be $3.2e-4$ according to the level of reward overoptimization shown on qualitative results.

We use minibatches of size 8 during sampling and 4 during training due to the limited GPU memory. The total batch size of 768 is evenly distributed among each GPU, so that the per GPU batch size is 16. The model samples two minibatches of size 8 on all GPUs to reach the total batch size. Similarly, the model accumulates gradients over four minibatches of size 4 on all GPUs, before synchronizing the gradients and performing an optimizer step. We use a per prompt stat tracker with windows of size 76, so that it roughly tracks all the rewards per prompt ever 3 epochs. This is much larger than DDPO's default tracking window of size 32 for better training stability. The coefficients for the advantage terms in Eq. (8) are $\alpha = 1$ and $\beta = 0.2$.

The rest of our RL finetuning setup follows DDPO [4, 5]. We use the AdamW [29] optimizer with a fixed learning rate $3e-4$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1e-8$ and a weight decay of $1e-4$. The high learning rate is paired with Low Rank Adaptation (LoRA) [18] finetuning with rank 4, which significantly reduces the memory and computation requirements for finetuning. We freeze all networks in the base model and set their precision to fp16, and only finetune the LoRA weights of the unet under fp32 via mixed precision training.

Our base text-to-multiview diffusion model setup follows Instant3D [24], which uses the same architecure as SDXL [39]. It produces images of resolution 1024x1024, which contains four images of resolution 512x512 tiled in a 2-by-2 fashion. Instant3D requires 100 denoising steps during inference, doubling the required computation than the default 50 steps for SDXL. It uses Classifier Free Guidance [16] with a scale of 5.0

Our code is mainly based on DDPO's [5] official implementation, the ddpo-pytorch [4] Github repository, which uses Hugginface diffusers [50] and PyTorch [37] libraries. Our KL divergence regularization implementation is inspired by the codebases of DeepSpeedChat [55], TRL [51], and DPOK [15]. We thank the authors of these repositories for releasing the high-quality implementations and promoting open-sourced research. We are going to release the code for computing MRC and our improved DDPO implementation. However, due to the fact that Sparse View LRM and Instant3D do not plan to release their code, we have to leave these as empty, abstract functions in our released code.

C.3. Experiment Details

All quantitative and qualitative experiments and the user study uses the results from evaluating each model on the DreamFusion [40] testing prompt set, containing 415 prompts, for 4 times, which equals to 1660 results per model.

For fair qualitative comparisons, results from each model are generated from the the same initial noise. Since Instant3D-20K and -100K and Carve3D are all finetuned from Instant3D-10K, their output tend to represent the same object when given the same initial noise (e.g. Figs. 1, 4 and 11).

Since Zero123++ [47] and SyncDreamer [28] are image-to-multi-view diffusion models, we let them take one of Carve3D's output image and their input image-conditioning. Therefore, their output has the same level of prompt alignment, texture details, and diversity as Carve3D.

C.4. User Study Details

To run the study we *randomly* selected 20 unseen testing prompts. For each text prompt, we generated a pair of data from both the base and the finetuned models with the same initial noise. Then, we provided both the tiled 4-view image and the turntable video of the reconstructed NeRF to participants and asked them the following two questions: (1) Which result is more 3D-consistent? and (2) Which result is better aligned with the prompt?

D. Additional MRC Metric Experiments

Distortion Types Here, we show the full results for the metric experiments for the inpainting distortion (Fig. 14) discussed in Sec. 3.3 and Figs. 7 and 8. We also conduct metric experiments with other distortions types: azimuth rotation (Fig. 15, and elevation rotation (Fig. 16). In azimuth and elevation rotation, for one out of the four views, we rotate the object with an azimuth or elevation rotation by 3.6 or 4 degrees, before rendering that view, and also use the original camera extrinsic matrix as the input to Sparse View LRM [17, 24]. The quantitative results matches our expectations, where MRC, i.e. with LPIPS, monotonically decreases as we intentionally add more distortion.

LPIPS vs. Other Image Similarity Metrics Here, we compare substituting LPIPS [57] with L1, L2, PSNR, and SSIM in the Multi-view Reconstruction Consistency (MRC) metric experiments on all distortion types. In the inpainting distortion experiments (Fig. 14), which is the most representative of diffusion model's inconsistencies, LPIPS is more linear than other pixel level image metrics. In azimuth and elevation distortion experiments (Figs. 15 and 16), all image metrics shows monotonically decreasing pattern, while pixel-level image metrics are more linear. This is expected as the distortion is pixel-aligned and more structured.

E. Ablation Studies

Bounding Box Normalization As shown in Fig. 9, when the bounding box normalization is removed from MRC, the model would trivially increase the reward by reducing the size of the foreground object on the white background. This would lead to the model generating images containing only the white background, after longer finetuning. With bounding box normalization, the model would learn the harder task of improving the reconstruction consistency of the multi-view images.

KL Divergence Regularization As shown in Fig. 10 Our KL divergence regularization does not sacrifice the model's efficiency on improving its reward. Without KL divergence regularization, the KL divergence grows much faster. As discussed in Sec. 4.2, this leads to degraded object identity and loss of texture details.

F. Additional Related Work

F.1. 3D Generation with 2D Diffusion Models

3D models can be derived from either single or multi-view images by optimizing the Score Distillation Sampling (SDS) loss [40, 52]. However, the optimization process is

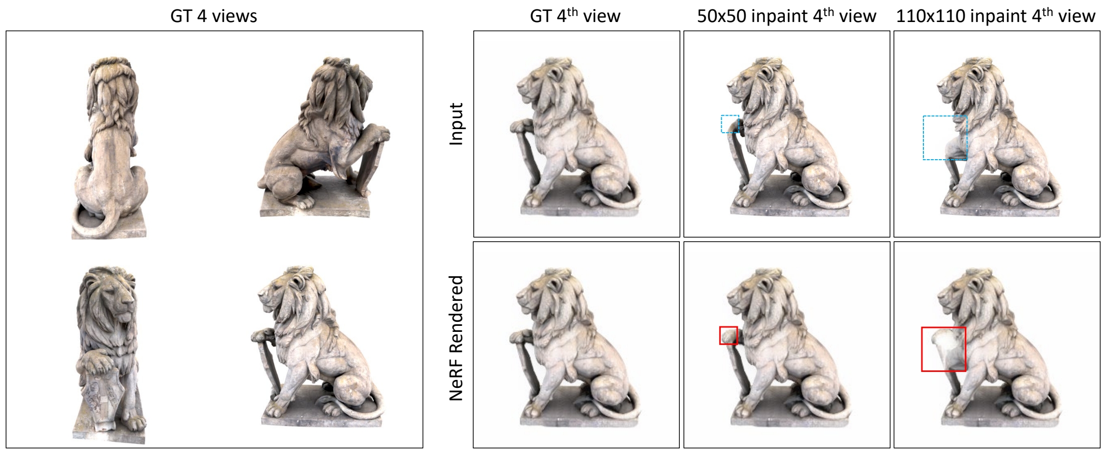

Figure 7. Qualitative correlation between MRC and multi-view inconsistency with increasing intensity, introduced by inpainting with increasing mask sizes. Left: the four ground truth views. Right: the 4th view is inpainted with increasing area sizes, i.e. 0×0, 50×50 and 110×110 pixels. The top row is the image after inpainting and the bottom row is the image rendered from the NeRF reconstructed with the top inpainted 4th view and the other 3 original GT views. We mark the inpainting area with blue and red boxes. Since the lion's right paw in the inpainted 4th views look different from the other three original views, its shape is broken in the NeRF and the rendered views. This difference is captured in MRC's image dissimilarity metric.

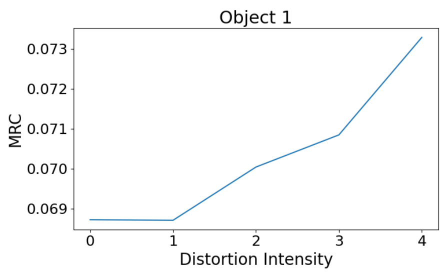

Figure 8. Quantitative correlation between MRC and multi-view inconsistency with increasing intensity, for the object shown in Figure 7. As inconsistency intensity rises, MRC also monotonically increases.

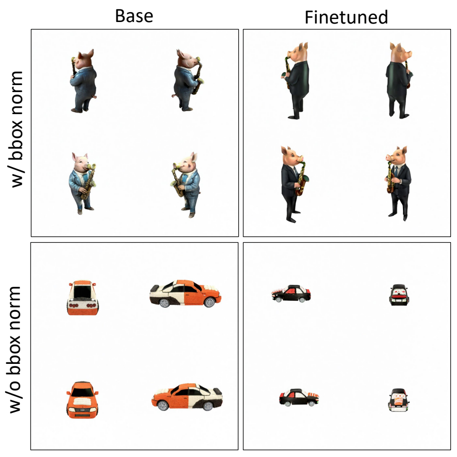

Figure 9. Alation study on the boundingbox normalization of LPIPS for MRC evaluation. Top: with boundingbox normalization, the size of the foreground object is similar that of the base model. Bottom: without boundingbox normalization, the size of the foreground object after RL finetuning is substantially smaller than that of the base model.

notably time-consuming, requiring multiple hours to generate a single 3D asset. In contrast, Large Reconstruction Model (LRM) [17], trained on the extensive 3D dataset Objaverse [12], can efficiently reconstruct NeRF models from a single image in a feed-forward manner. In this work, we focus exclusively on text-to-3D using feed-forward sparse-view NeRF reconstruction, specifically employing sparse-view LRM [24]. This choice is driven by its significantly faster performance compared to SDS-based optimization methods and its superior quality relative to feed-forward text-to-3D diffusion models [19, 34]. We choose Instant3D [24] as our base multi-view diffusion model, owing to its light-weight Supervised Finetuning (SFT) that preserves the strong semantic understanding and high-quality image generation capabilities of SDXL [39], similar to the instruction finetuning stage in InstructGPT [36].

F.2. RLFT of LLMs and Diffusion Models

Reinforcement Learning (RL) has been widely used to finetune large pre-trained models in Natural Language Processing [2, 3, 22, 36] and Computer Vision [5, 11, 15, 38, 41, 58], due to its advantage over Supervised Finetuning (SFT). SFT directly fits a model to the distribution of the SFT dataset containing inputs and ground-truth target data, which unavoidably causes some degree of distribution shift [44]. On the contrary, based on an objective function

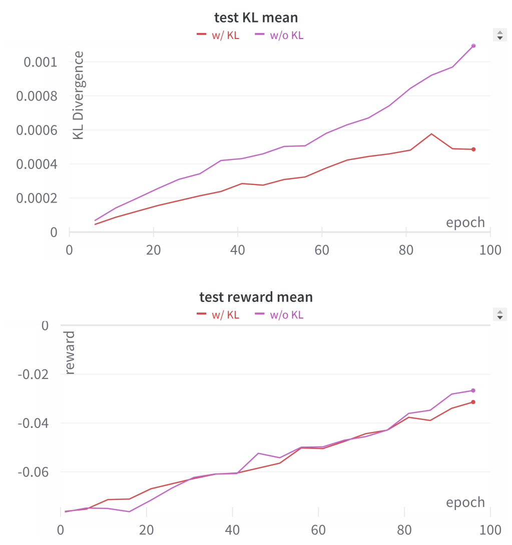

Figure 10. Ablation study on KL divergence regularization. Top: KL Divergence between the base model and the finetuned model on testing set. Bottom: mean MRC reward on testing set. Our KL divergence regularization does not sacrifice the model's sample efficiency of the reward curve. The finetuned model has a lower KL divergence to the base model when using KL divergence regularization than without, which prevents degraded object identity and reduced texture details.

and a dataset containing only inputs, Reinforcement Learning Finetuning (RLFT) optimizes a model beyond the limitation of a SFT dataset by using its own outputs and effectively mitigates distribution shift [7].

RLFT of LLMs Large Language Models (LLMs) like GPT-3 [6] are pre-trained on the next-word prediction task on an internet-scale corpus. While the autoregressive pre-training is a powerful self-supervised objective that allows LLMs to extract substantial knowledge from the internet-scale unlabeled dataset, pre-trained LLMs can only perform the corresponding text completion task. The pre-training lacks an objective that allows LLMs to respond to text prompts. In InstructGPT [36], the paper behind ChatGPT 3.5, a two-stage finetuning solution is proposed to align GPT-3 to answer instructions according to human preferences. In the first stage, InstructGPT employs SFT with a small dataset of hand-crafted prompt-answer pairs. While SFT changes the model's output distribution from text completion to answering instructions, it also introduces hallucination [44]. This is because the output distribution drifts too much towards the instruction-following dataset, and the model tries to imitate the behavior in the data and always provide plausible answers even when the model is uncertain about the answer [44]. To address this issue, InstructGPT opts for a light-weight SFT stage and relies on RLFT in the second stage, using a human-preference reward model. This approach provides general alignment to human values and causes minimal hallucination [44], because RLFT does not rely on a potentially biased dataset containing fixed ground-truth answers, but instead learns the general concept of human-preference through the reward model. The success of InstructGPT [36] and its analogy to the distribution shift problem in multi-view SFT [24] motivate us to pursue RLFT for 2D diffusion models.

RLFT of Diffusion Models Witnessing the success of RLFT methods in LLMs [2, 3, 22, 36], recently, a few RLFT algorithms have been proposed for text-to-image diffusion models. RWR [23] is the first work to bring the human feedback reward finetuning idea to diffusion models. While RWR only finetunes stable diffusion [42] via a single log probability of the entire denoising process, multi-step RLFT can be facilitated by treating the denoising process as a multi-step MDP, as demonstrated in DDPO [5] and DPOK [15]. Our RLFT is based on DDPO [5], while our KL-divergence regularization is similar to DPOK [15] and InstructGPT [36]. Furthermore, RWR, DDPO, and DPOK all finetune SD-1.5 [42], while we finetune a much larger diffusion model based on SDXL. We also study training stability, a notorious challenge in both traditional RL and RLFT [7, 60], and scaling laws [20] for RLFT.

G. Broader Impacts and Future Work

Our Multi-view Reconstruction Consistency (MRC) metric can serve as a valuable tool for evaluating any multi-view generative methods and guiding future developments in the field. Although we only demonstrate our Reinforcement Learning Finetuning (RLFT) with MRC on one multi-view diffusion model [24], it can be directly adapted to other text-to-multi-view diffusion models; such adaptation only requires tuning a few hyperparameters related to the scaling laws for diffusion model RLFT (Sec. 4.2.3). Our surprising finding behind the choice of REINFORCE [54] over PPO [45] for better training stability could also be applied in broader RLFT scenarios.

As AI models grow more powerful, it becomes more important to evaluate and improve their safety and reduce their bias. RLFT has been widely used for Large Language Model (LLM) alignment as it allows models to be finetuned with hard-to-specify objectives and its results are generalizable without undermining the base model's knowledge. As the first work to use RLFT for text-to-3D and on diffusion models at the SDXL scale, we hope Carve3D can inspire more alignment research in the computer vision community.

Carve3D is limited by the reconstruction quality of Sparse View Large Reconstruction Model (LRM) [17, 24]. Because its reconstruction is not perfect, this leads to non-zero MRC metric on GT views as shown in Figs. 14 to 16.

Due to this limitation of Sparse View LRM, Carve3D RL finetuned model can produce less high-frequency details than the base model in order to lower the image distance to the NeRF rendered views. This might be solved by using a future sparse view reconstructor that can preserve more details or training a dedicated model for computing MRC.

Further increasing data size and batch size to further improve reconstruction consistency is possible. However, in this work, we are limited by the high computation cost of SDXL [39], Instant3D's 100 denoising steps, and the high number of samples needed in DDPO. A few concurrent works could address this challenge. It is possible to substantially reduce the computation cost by switching to Consistency Models for one/few-step inference (e.g., LCM-LoRA [30]). In addition, we can also switch from DDPO to direct backpropagation of reward (e.g. Align-Prop [41], and DRaFT [11]) to reduce the number of samples needed. We leave these extensions as future work.

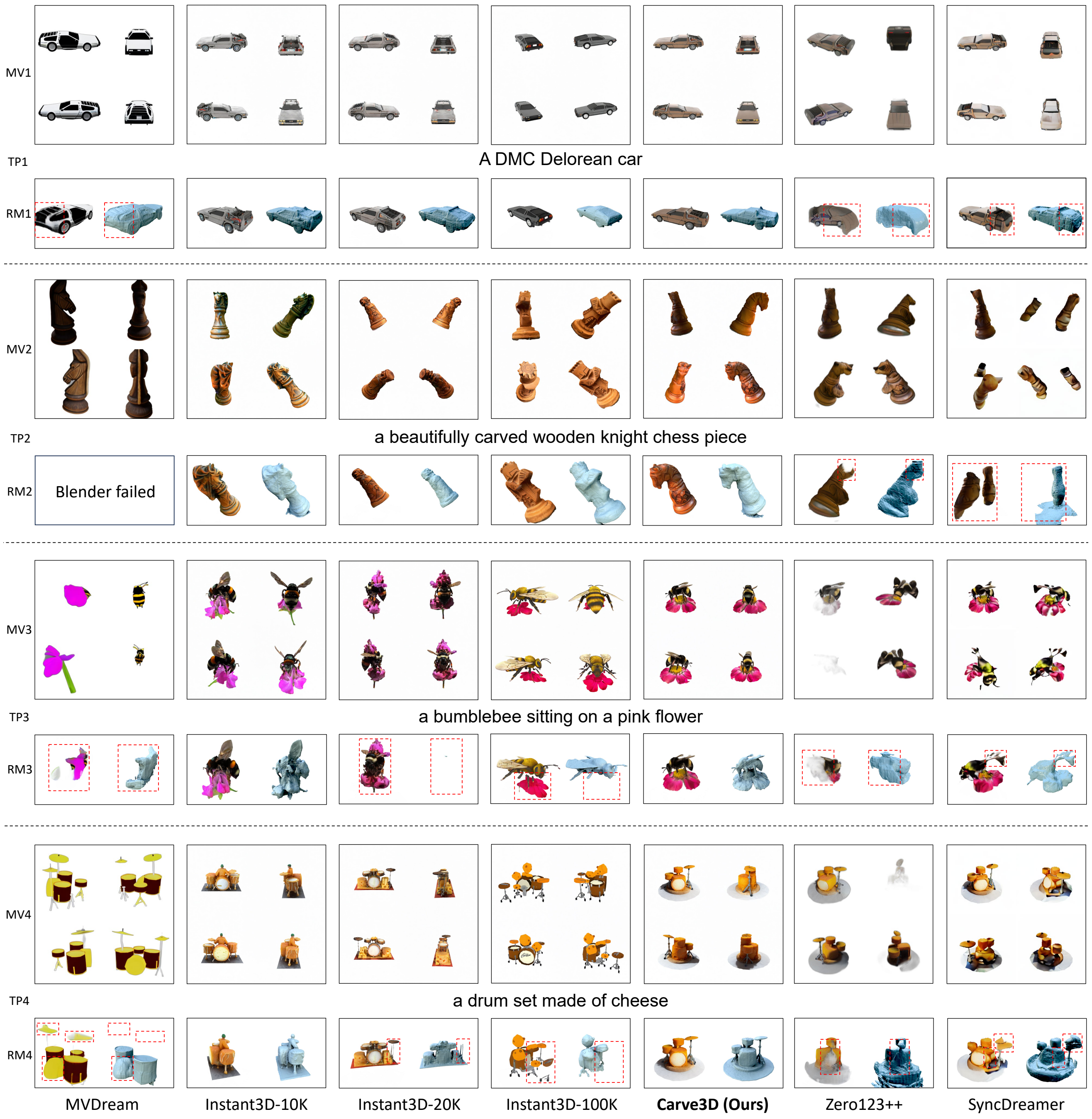

Figure 11. Qualitative comparison of MVDream [46], Instant3D [24] with 10K (the base model), 20K, and 100K SFT steps, Carve3D (ours, finetuned from Instant3D-10K), Zero123++ [47], and SyncDreamer [28] (7 models in 7 columns) on 4 prompts (in 4 rows, numbered as 1-4, separated by dotted line). In each row, we show their generated multi-view images in the 2-by-2 grid (denoted as MV), reconstructed NeRF and extracted mesh (denoted as RM) when given the text prompt (denoted as TP). MVDream, Zero123++, and SyncDreamer generates inconsistent multi-view images and reconstruction artifacts (highlighted in red). For each result, we use the same randomly sampled initial noise for all models to ensure the comparison is fair. We let Zero123++ and SyncDreamer to use one of Carve3D's output multi-view images as their input image conditioning. Instant3D-10K, -20K, and -100K and Carve3D demonstrates progressively better multi-view consistency and reconstruction quality. Instant3D-10K, Carve3D, Zero123++, and SyncDreamer exhibits the best texture details and realism, whereas Instant3D-20K and -100K with prolonged SFT steps compromise those qualities.

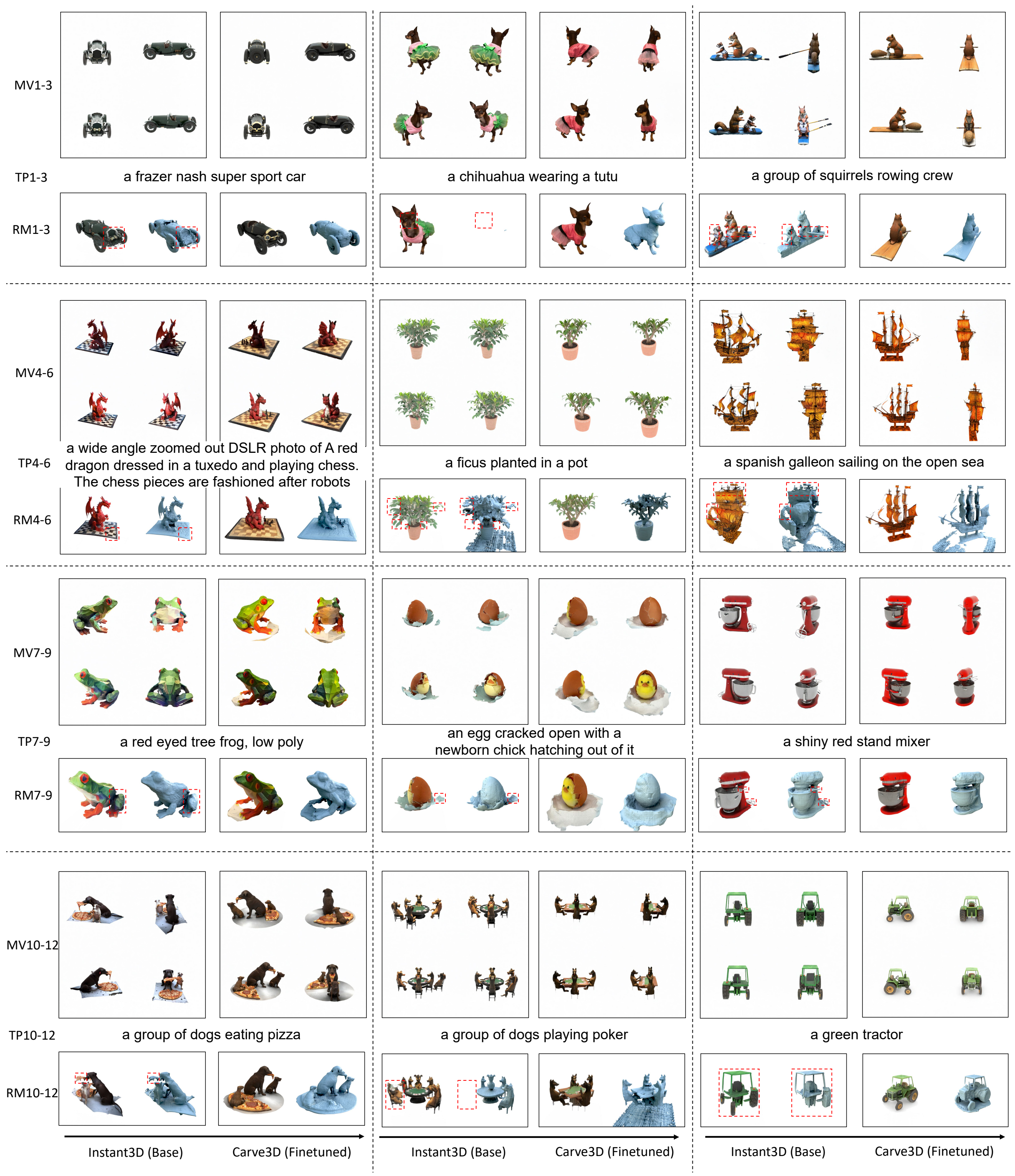
MV1-3
TP1-3
a frazer nash super sport car
a chihuahua wearing a tutu
a group of squirrels rowing crew
RM1-3
MV4-6
TP4-6
a wide angle zoomed out DSLR photo of A red dragon dressed in a tuxedo and playing chess. The chess pieces are fashioned after robots
a ficus planted in a pot
a spanish galleon sailing on the open sea
RM4-6
MV7-9
TP7-9
a red eyed tree frog, low poly
an egg cracked open with a newborn chick hatching out of it
a shiny red stand mixer
RM7-9
MV10-12
TP10-12
a group of dogs eating pizza
a group of dogs playing poker
a green tractor
RM10-12
Instant3D (Base)
Carve3D (Finetuned)
Instant3D (Base)
Carve3D (Finetuned)
Instant3D (Base)
Carve3D (Finetuned)

Figure 12. Qualitative comparison of Instant3D (the base model) and Carve3D (ours, finetuned from Instant3D) on 12 prompts (in 12 blocks, numbered as 1-12, separated by dotted line). In each block, we show the their generated multi-view images in the 2-by-2 grid (denoted as MV), the reconstructed NeRF and the extracted mesh (denoted as RM) when given the text prompt (denoted as TP). For each result, we use the same randomly sampled initial noise for all models to ensure the comparison is fair. We draw red boxes on the NeRF and the extracted mesh to highlight the artifacts in the NeRF and the mesh, resulting from the inconsistencies in the multi-view images. Carve3D maintains the detailed texture and provides improved multi-view consistency and higher quality NeRF than the base Instant3D.

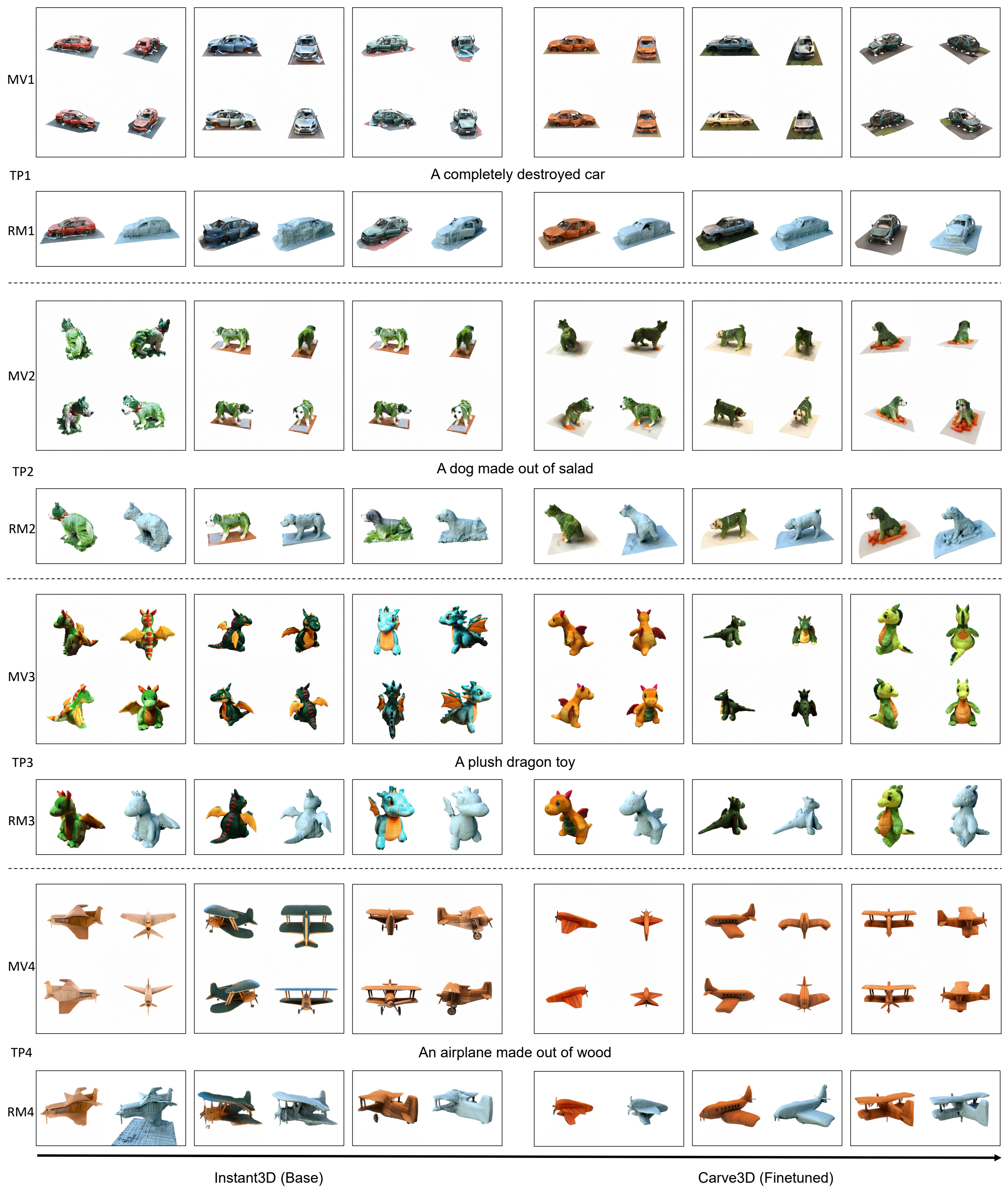

Figure 13. Qualitative comparison of Instant3D (the base model) and Carve3D (ours, finetuned from Instant3D) on 4 prompts (in 4 rows, numbered as 1-4, separated by the dotted line) demonstrating diversity. In each row, we show 3 results from each model, including the generated multi-view images in the 2-by-2 grid (denoted as MV), the reconstructed NeRF and the extracted mesh (denoted as bottom) when given the prompt (denoted as middle). For each result, we use the same randomly sampled initial noise for all models to ensure the comparison is fair. Our RLFT maintains the diversity of the base Instant3D model, while improving the consistency.

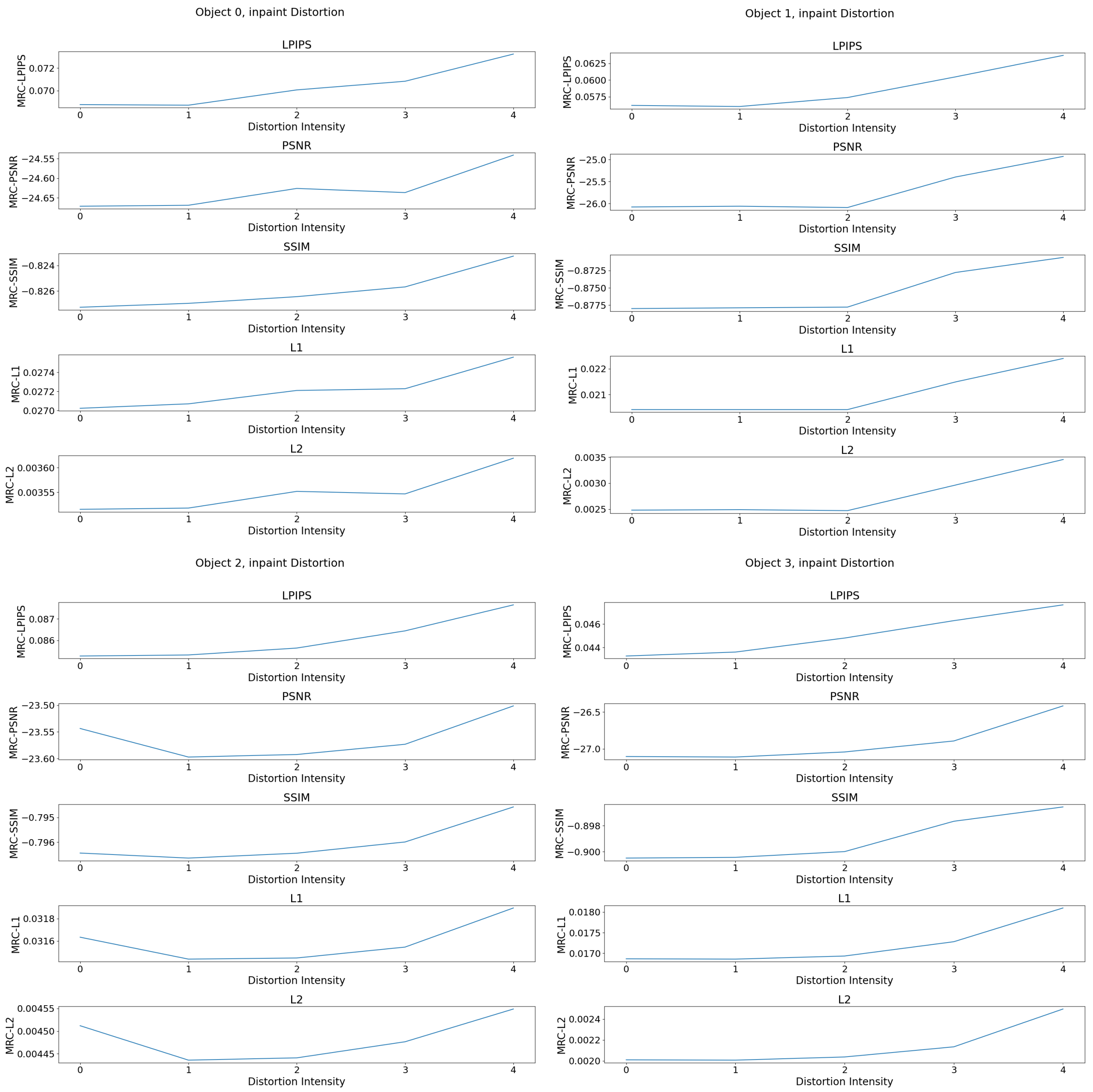

Figure 14. Quantitative correlation between five variants of MRC (our default LPIPS, as well as PSNR, SSIM, L1, and L2) and inconsistency introduced by inpaint distortion with increasing intensity on four objects. We take negative of the similarity metrics (PSNR and SSIM) for easy comparisons to the distance metrics (LPIPS, L1, and L2). LPIPS constantly exhibits monotonically increasing pattern with respect to the increased inconsistency, while other image metrics do not.

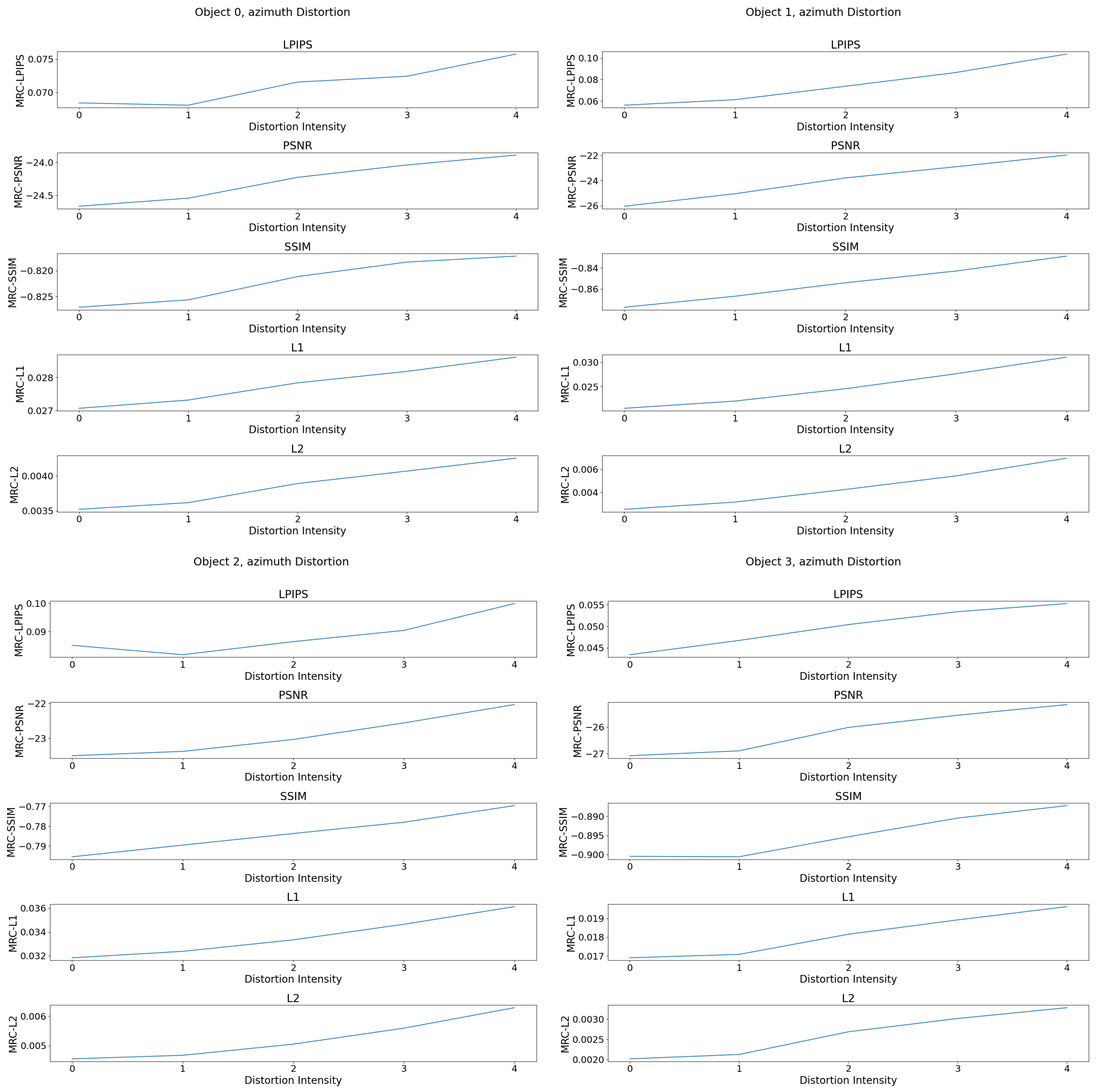

Figure 15. Quantitative correlation between five variants of MRC (our default LPIPS, as well as PSNR, SSIM, L1, and L2) and inconsistency introduced by azimuth rotation distortion with increasing intensity on four objects. We take negative of the similarity metrics (PSNR and SSIM) for easy comparisons to the distance metrics (LPIPS, L1, and L2). All metrics constantly exhibits monotonically, steadily increasing pattern with respect to the increased inconsistency.

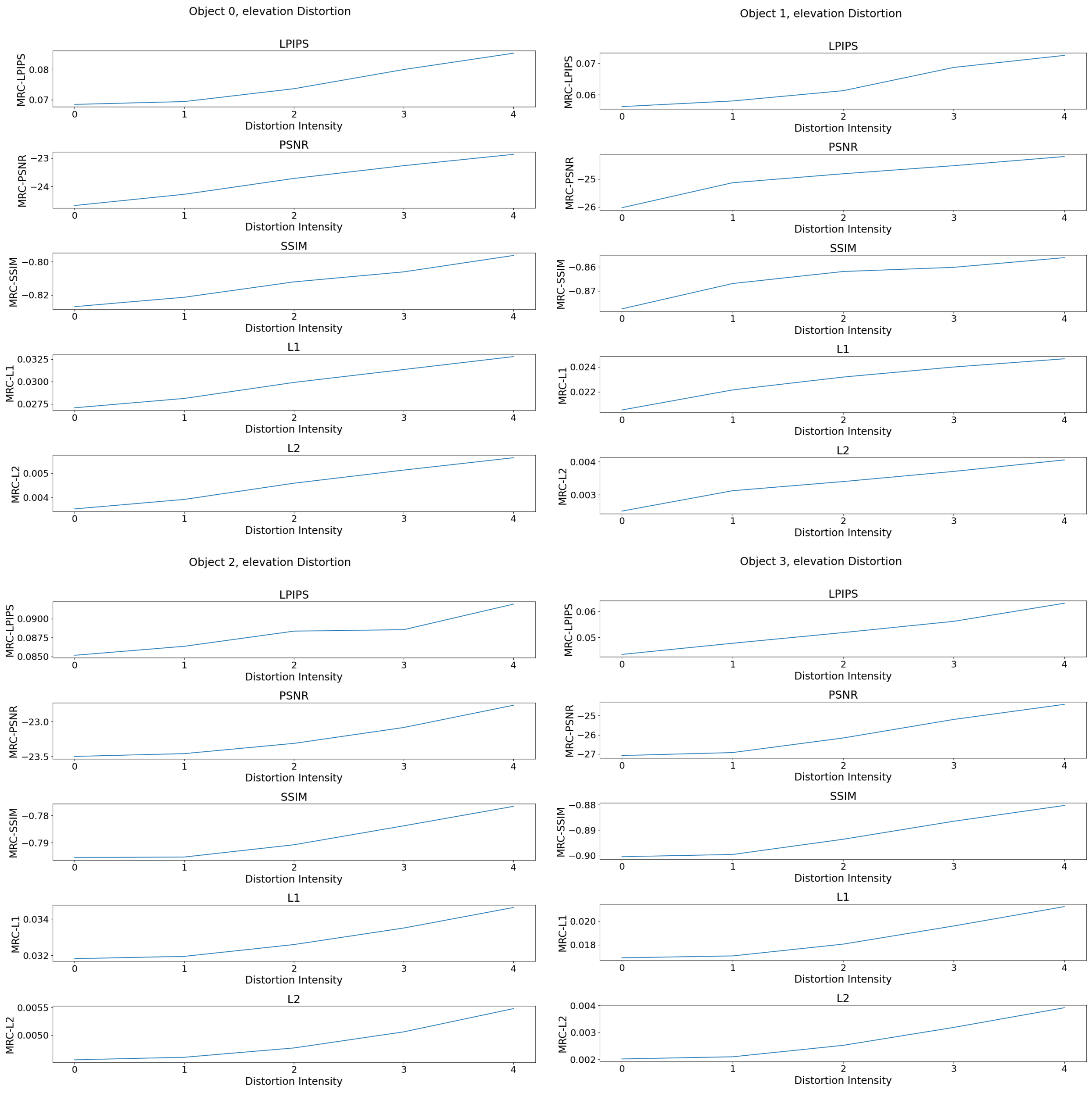

Figure 16. Quantitative correlation between five variants of MRC (our default LPIPS, as well as PSNR, SSIM, L1, and L2) and inconsistency introduced by elevation rotation distortion with increasing intensity on four objects. We take negative of the similarity metrics (PSNR and SSIM) for easy comparisons to the distance metrics (LPIPS, L1, and L2). All metrics constantly exhibits monotonically, steadily increasing pattern with respect to the increased inconsistency.