NerfAcc: Efficient Sampling Accelerates NeRFs

Ruilong Li
UC Berkeley
ruilongli@berkeley.edu

Hang Gao
UC Berkeley
hangg@berkeley.edu

Matthew Tancik
UC Berkeley
tancik@berkeley.edu

Angjoo Kanazawa
UC Berkeley
kanazawa@berkeley.edu

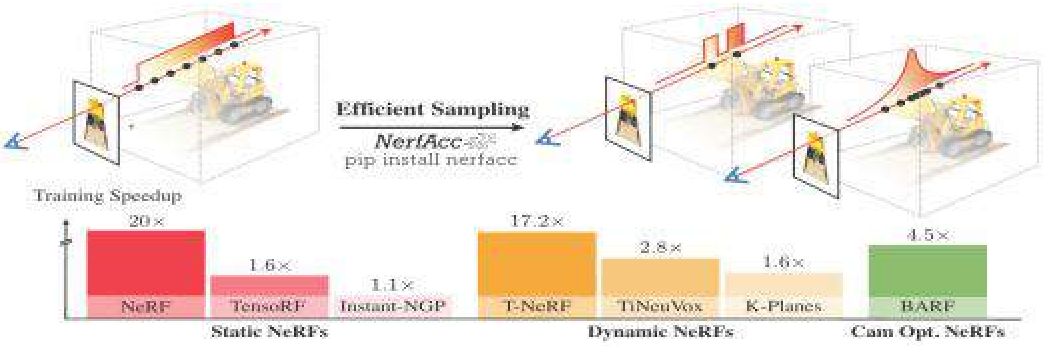


Figure 1: **NerfAcc Toolbox**. Our proposed toolbox, *NerfAcc*, integrates advanced efficient sampling techniques that lead to significant speedups in training various recent NeRF papers with minimal modifications to existing codebases.

Abstract

Optimizing and rendering Neural Radiance Fields is computationally expensive due to the vast number of samples required by volume rendering. Recent works have included alternative sampling approaches to help accelerate their methods, however, they are often not the focus of the work. In this paper, we investigate and compare multiple sampling approaches and demonstrate that improved sampling is generally applicable across NeRF variants under an unified concept of transmittance estimator. To facilitate future experiments, we develop NerfAcc, a Python toolbox that provides flexible APIs for incorporating advanced sampling methods into NeRF related methods. We demonstrate its flexibility by showing that it can reduce the training time of several recent NeRF methods by 1.5× to 20× with minimal modifications to the existing codebase. Additionally, highly customized NeRFs, such as Instant-NGP, can be implemented in native PyTorch using NerfAcc. Our code are open-sourced at https://www.nerfacc.com

1. Introduction

Neural volume rendering has revolutionized the inverse rendering problem, with the Neural Radiance Field (NeRF) [16] being a key innovation. The continuous radiance field representation allows for rendering novel views of a scene from any camera position. However, the optimization of a NeRF can be computationally expensive due to the neural representation of radiance field and the large number of samples required by volume rendering. These challenges have limited practical applications of NeRF-based optimization and rendering.

Several recent works have successfully reduced the computational cost of neural volume rendering by proposing more efficient radiance field representations [17, 35, 36, 4, 24, 7]. While there are differences in the specific radiance field representations and their applications, most of these methods share a similar volume rendering pipeline which involves creating samples along the ray and accumulating them through alpha-composition.

However, compared to the considerable efforts focused on developing efficient radiance field representations, there has

GETMusic: Generating Music Tracks with a Unified Representation and Diffusion Framework

Ang Lv[‡†], Xu Tan[†*], Peiling Lu[†], Wei Ye[§], Shikun Zhang[§], Jiang Bian[†], Rui Yan[‡*]
[†]Microsoft Research Asia
[‡]Gaoling School of Artifical Intelligence, Renmin University of China
[§]National Engineering Research Center for Software Engineering, Peking University
{anglv, ruiyan}@ruc.edu.cn, {xuta, peil, jiabia}@microsoft.com,
{wye,zhangsk}@pku.edu.cn

https://github.com/microsoft/muzic

Abstract

Symbolic music generation aims to create musical notes, which can help users compose music, such as generating target instrument tracks based on provided source tracks. In practical scenarios where there's a predefined ensemble of tracks and various composition needs, an efficient and effective generative model that can generate any target tracks based on the other tracks becomes crucial. However, previous efforts have fallen short in addressing this necessity due to limitations in their music representations and models. In this paper, we introduce a framework known as GETMusic, with "GET" standing for "GEnerate music Tracks." This framework encompasses a novel music representation "GETScore" and a diffusion model "GETDiff." GETScore represents musical notes as tokens and organizes tokens in a 2D structure, with tracks stacked vertically and progressing horizontally over time. At a training step, each track of a music piece is randomly selected as either the target or source. The training involves two processes: In the forward process, target tracks are corrupted by masking their tokens, while source tracks remain as the ground truth; in the denoising process, GETDiff is trained to predict the masked target tokens conditioning on the source tracks. Our proposed representation, coupled with the non-autoregressive generative model, empowers GETMusic to generate music with any arbitrary source-target track combinations. Our experiments demonstrate that the versatile GETMusic outperforms prior works proposed for certain specific composition tasks.

1 Introduction

Symbolic music generation aims to create musical notes, which can help users in music composition. Due to the practical need for flexible and diverse music composition, the need for an efficient and unified approach capable of generating arbitrary tracks based on the others is high[2]. However, current research falls short of meeting this demand due to inherent limitations imposed by their representations and models. Consequently, these approaches are confined to specific source-target combinations, such as generating piano accompaniments based on melodies.

[*]Corresponding authors: Xu Tan (xuta@microsoft.com) and Rui Yan (ruiyan@ruc.edu.cn).

[2]A music typically consists of multiple instrument tracks. In this paper, given a predefined track ensemble, we refer to the tracks to be generated as "target tracks" and those acting as conditions as "source tracks." We refer to such an orchestration of tracks as a "source-target combination."

Tree-Ring Watermarks: Fingerprints for Diffusion Images that are Invisible and Robust

Yuxin Wen, John Kirchenbauer, Jonas Geiping, Tom Goldstein
University of Maryland

Abstract

Watermarking the outputs of generative models is a crucial technique for tracing copyright and preventing potential harm from AI-generated content. In this paper, we introduce a novel technique called *Tree-Ring Watermarking* that robustly fingerprints diffusion model outputs. Unlike existing methods that perform post-hoc modifications to images after sampling, *Tree-Ring Watermarking* subtly influences the entire sampling process, resulting in a model fingerprint that is invisible to humans. The watermark embeds a pattern into the initial noise vector used for sampling. These patterns are structured in Fourier space so that they are invariant to convolutions, crops, dilations, flips, and rotations. After image generation, the watermark signal is detected by inverting the diffusion process to retrieve the noise vector, which is then checked for the embedded signal. We demonstrate that this technique can be easily applied to arbitrary diffusion models, including text-conditioned Stable Diffusion, as a plug-in with negligible loss in FID. Our watermark is semantically hidden in the image space and is far more robust than watermarking alternatives that are currently deployed. Code is available at `https://github.com/YuxinWenRick/tree-ring-watermark`.

1 Introduction

The development of diffusion models has led to a surge in image generation quality. Modern text-to-image diffusion models, like Stable Diffusion and Midjourney, are capable of generating a wide variety of novel images in an innumerable number of styles. These systems are general-purpose image generation tools, able to generate new art just as well as photo-realistic depictions of fake events for malicious purposes.

The potential abuse of text-to-image models motivates the development of *watermarks* for their outputs. A watermarked image is a generated image containing a signal that is invisible to humans and yet marks the image as machine-generated. Watermarks document the use of image generation systems, enabling social media, news organizations, and the diffusion platforms themselves to mitigate harms or cooperate with law enforcement by identifying the origin of an image [Bender et al., 2021, Grinbaum and Adomaitis, 2022].

Research and applications of watermarking for digital content have a long history, with many approaches being considered over the last decade [O'Ruanaidh and Pun, 1997, Langelaar et al., 2000]. However, so far research has always conceptualized the watermark as a minimal modification imprinted onto an existing image [Solachidis and Pitas, 2001, Chang et al., 2005, Liu et al., 2019, Fei et al., 2022]. For example, the watermark currently deployed in Stable Diffusion [Cox et al., 2007], works by modifying a specific Fourier frequency in the generated image.

The watermarking approach we propose in this work is conceptually different: This is the first watermark that is truly invisible, as no post-hoc modifications are made to the image. Instead, the *distribution of generated images is imperceptibly modified and an image is drawn from this*

Constrained sampling Diffusion models have been shown to be effective at solving inverse problems such as image in-painting, colorization and sparse-view computed tomography by using a controllable sampling process [4–6, 22, 24, 43, 44]. Concurrent work [53] explores diffusion modeling for controllable traffic generation, which we compare to in Sec. 3.4. In diffusion models, the generation process can be conditioned on information not available during training. The inverse problem can be posed as sampling from the posterior $p(\boldsymbol{x}; \boldsymbol{y})$ based on a learned unconditional distribution $p(\boldsymbol{x})$, where $\boldsymbol{y}$ is an observation of the event $\boldsymbol{x}$. We defer further technical details to Sec. 3.4.

Motion prediction There are two main categories of approaches for motion prediction: supervised learning and generative learning. Supervised learning trains a model with logged trajectories with supervised losses such as L2 loss. One of the challenges is to model inherent multi-modal behavior of the agents. For this, MultiPath [40] uses static anchors, and MultiPath++ [48], Wayformer [31], SceneTransformer [32] use learned anchors, and DenseTNT [13] uses goal-based predictions. Home [9] and GoHome [10] predict future occupancy heatmaps, and then decode trajectories from the samples. MP3 [2] and NMP [50] learn the cost function evaluator of trajectories, and then the output trajectories are heuristically enumerated. Many of these approaches use ensembles for further diversified predictions. The next section covers generative approaches.

Generative models for motion prediction Various recent works have modeled the motion prediction task as a conditional probability inference problem of the form $p(\boldsymbol{s}; \boldsymbol{c})$ using generative models, where $\boldsymbol{s}$ denote the future trajectories of one or more agents, and $\boldsymbol{c}$ denote the context or observation. HP-GAN [1] learns a probability density function (PDF) of future human poses conditioned on previous poses using an improved Wasserstein Generative Adversarial Network (GAN). Conditional Variational Auto-Encoders (C-VAEs) [11, 20, 34], Normalizing Flows [8, 28, 29, 41] have also been shown to be effective at learning this conditional PDF of future trajectories for motion prediction. Very recent works have started looking into diffusion models as an alternative to modeling the conditional distributions of future sequences such as human motion pose sequences [38, 52] and planning [21]. In a more relevant work, [14] the authors utilize diffusion models to model the uncertainties of pedestrian motion. As far as we are aware, we are the first to utilize diffusion models to model the multi-agent joint motion distribution.

Multi-agent motion prediction While much of the motion prediction literature has worked on predicting motions of individual agents independently, there has been some work to model the motion of multiple agents jointly. SceneTransformer [32] outputs a fixed set of joint motion predictions for all the agents in the scene. M2I [45], WIMP [25], PIP [42], and CBP [47] propose a conditional model where the motions of the other agents are predicted by given motions of the controlled agents.

There is a set of literature using probabilistic graphical models. DSDNet [51] and MFP [46] use fully connected graphs. JFP [27] supports static graphs such as fully connected graphs and autonomous vehicle centered graphs, and dynamic graphs where the edges are constructed between the interacting agents. RAIN [26] learns the dynamic graph of the interaction through separate RL training.

3. Method

3.1. Diffusion Model Preliminaries

Preliminaries Diffusion models [23] provide a learned parameterization of the probability distribution $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ through learnable parameters $\boldsymbol{\theta}$. Denote this probability density function, convolved with a Gaussian kernel of standard deviation σ to be $p_{\boldsymbol{\theta}}(\boldsymbol{x}, \sigma)$. Instead of directly learning a normalized probability density function $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ where the normalization constant is generally intractable [19], diffusion models learn the score function of the distribution: $\nabla_{\boldsymbol{x}} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}; \sigma)$ at a range of noise levels σ.

Given the score function $\nabla_{\boldsymbol{x}} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}; \sigma)$, one can sample from the distribution by denoising a noise sample. Samples can be drawn from the underlying distribution $\boldsymbol{x}_0 \sim p_{\boldsymbol{\theta}}(\boldsymbol{x})$ via the following dynamics:

$$\boldsymbol{x}_0 = \boldsymbol{x}(T) + \int_T^0 -\dot{\sigma}(t)\sigma(t)\nabla_{\boldsymbol{x}} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}(t); \sigma(t))dt$$
$$\text{where} \quad \boldsymbol{x}(T) \sim \mathcal{N}(\boldsymbol{0}, \sigma_{\max}^2 \boldsymbol{I}) \quad (1)$$

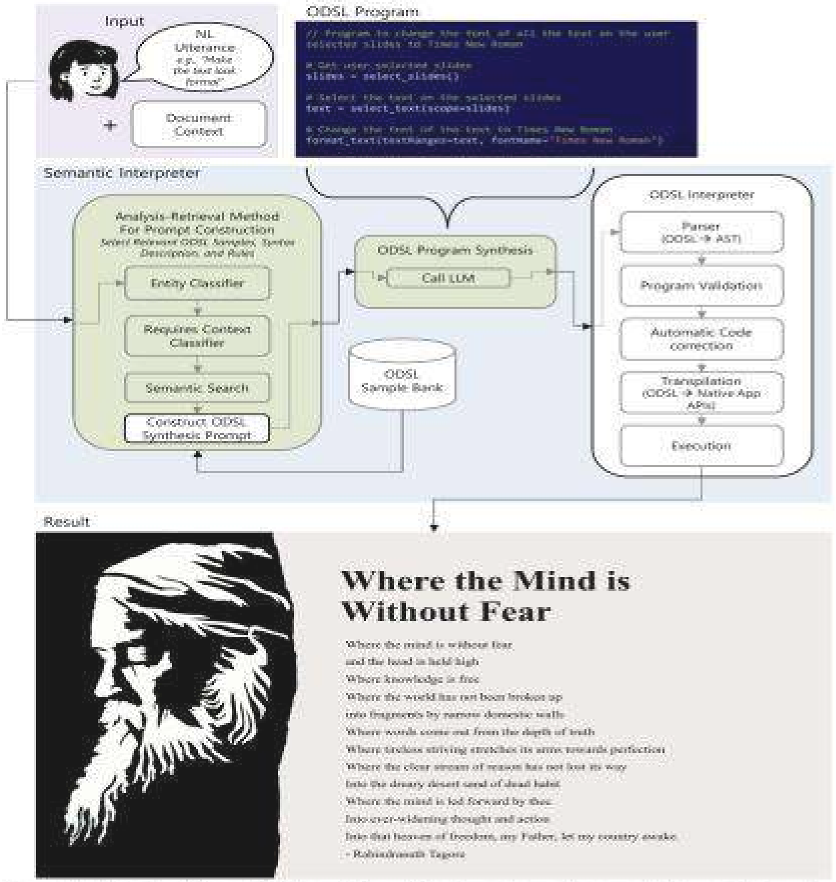
where variance $\sigma(t)$ is a monotonic, deterministic function of an auxiliary parameter of time t. Following [23], we use the linear noise schedule $\sigma(t) = t$. The initial noise sample is sampled i.i.d. from a unit Gaussian scaled to the highest standard deviation $\sigma(T) = \sigma_{\max}$.

The diffusion model can be trained to approximate a data distribution $p_{\chi}(\boldsymbol{x})$, where $\chi = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_{N_d}\}$ denote the set of training data. The empirical distribution of the data can be viewed as a sum of delta functions around each data point: $p_{\chi}(\boldsymbol{x}) = \frac{1}{n}\sum_{i=0}^{N_d} \delta(\boldsymbol{x} - \boldsymbol{x}_i)$. Denote the denoiser as $D(\boldsymbol{x}; \sigma)$ which is a function that recovers the unnoised sample corresponding to the noised sample $\boldsymbol{x}$. The denoiser is related to the score function via:

$$\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}; \sigma) = (D(\boldsymbol{x}; \sigma) - \boldsymbol{x})/\sigma^2 \quad (2)$$

The denoiser can be learned by minimizing the expected L_2 denoising error for a perturbed sample $\boldsymbol{x}$ at any noise level σ sampled from the noise distribution $q(\sigma)$:

$$\arg\min_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{x} \sim p_{\chi}} \mathbb{E}_{\sigma \sim q(\sigma)} \mathbb{E}_{\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \sigma^2 \boldsymbol{I})} ||D_{\boldsymbol{\theta}}(\boldsymbol{x} + \boldsymbol{\epsilon}; \sigma) - \boldsymbol{x}||_2^2 \quad (3)$$


Input
NL Utterance e.g., "Make the text look formal"
+
Document Context
ODSL Program
Semantic Interpreter
Analysis-Retrieval Method For Prompt Construction
Select Relevant ODSL Samples, Syntax Description, and Rules
Entity Classifier
Requires Context Classifier
Semantic Search
Construct ODSL Synthesis Prompt
ODSL Program Synthesis
Call LLM
ODSL Sample Bank
ODSL Interpreter
Parser (ODSL → AST)
Program Validation
Automatic Code correction
Transpilation (ODSL → Native App APIs)
Execution
Result

```
// Program to change the font of all the text on the user
selected slides to Times New Roman

# Get user selected slides
slides = select_slides()

# Select the text on the selected slides
text = select_text(scope=slides)

# Change the font of the text to Times New Roman
format_text(textRange=text, fontName="Times New Roman")
```

Where the Mind is Without Fear

Where the mind is without fear
and the head is held high
Where knowledge is free
Where the world has not been broken up
into fragments by narrow domestic walls
Where words come out from the depth of truth
Where tireless striving stretches its arms towards perfection
Where the clear stream of reason has not lost its way
Into the dreary desert sand of dead habit
Where the mind is led forward by thee
Into ever-widening thought and action
Into that heaven of freedom, my Father, let my country awake
- Rabindranath Tagore

Figure 1: Illustration of Semantic Interpreter's architecture and overall approach. Semantic Interpreter translates a user utterance to an ODSL program by leveraging an an analysis-retrieval few-shot prompting approach with an LLM for program synthesis. The ODSL program is then validated and transpiled to the app's native APIs for execution.

Using a general-purpose language also makes it challenging to ensure safe code. Unlike a scoped DSL, using a general-purpose language encourages the model to use all available language features and libraries, some of which are potentially unsafe.

ing task, in our case, prediction of dialog states (34.5% JGA). Thus, the combined system with the Speech2Text adapter and the Speech2Text retriever outperforms a strong cascade baseline system (31.8% JGA) where the DST was trained on error-prone ASR transcripts. Similarly, the ReSLM model (8.6% WER) with the adapter and the retriever outperforms a strong in-domain ASR baseline (10.4% WER).

While the experiments are performed on DST task, the model is more widely applicable and its performance can be further improved with better retriever.

6. Acknowledgements

We would like to acknowledge Jeffrey Zhao, Abhinav Rastogi and Aramys Miranda for their invaluable help.

7. References

[1] A. Bapna, Y. Chung, N. Wu, A. Gulati, Y. Jia, J. H. Clark, M. Johnson, J. Riesa, A. Conneau, and Y. Zhang, "SLAM: A unified encoder for speech and language modeling via speech-text joint pre-training," *CoRR*, vol. abs/2110.10329, 2021. [Online]. Available: https://arxiv.org/abs/2110.10329

[2] S. Thomas, B. Kingsbury, G. Saon, and H.-K. J. Kuo, "Integrating text inputs for training and adapting rnn transducer asr models," in *Proc. ICASSP*, 2022.

[3] Z. Chen, Y. Zhang, A. Rosenberg, B. Ramabhadran, P. J. Moreno, A. Bapna, and H. Zen, "MAESTRO: Matched Speech Text Representations through Modality Matching," in *Proc. Interspeech*, 2022.

[4] J. Zhao, R. Gupta, Y. Cao, D. Yu, M. Wang, H. Lee, A. Rastogi, I. Shafran, and Y. Wu, "Description-driven task-oriented dialog modeling," *arXiv preprint arXiv:2201.08904*, 2022.

[5] H. Soltau, I. Shafran, M. Wang, A. Rastogi, J. Zhao, Y. Jia, W. Han, Y. Cao, and A. Miranda, "Speech aware dialog system technology challenge (dstc11)," *arXiv preprint arXiv:2212.08704*, 2022.

[6] M. Eric, R. Goel, S. Paul, A. Sethi, S. Agarwal, S. Gao, and D. Hakkani-Tür, "Multiwoz 2.1: Multi-domain dialogue state corrections and state tracking baselines," *CoRR*, vol. abs/1907.01669, 2019. [Online]. Available: http://arxiv.org/abs/1907.01669

[7] A. Rosenberg, Y. Zhang, B. Ramabhadran, Y. Jia, P. J. Moreno, Y. Wu, and Z. Wu, "Speech recognition with augmented synthesized speech," *CoRR*, vol. abs/1909.11699, 2019. [Online]. Available: http://arxiv.org/abs/1909.11699

[8] Z. Chen, Y. Zhang, A. Rosenberg, B. Ramabhadran, P. Moreno, and G. Wang, "Tts4pretrain 2.0: Advancing the use of text and speech in asr pretraining with consistency and contrastive losses," in *Proc. ICASSP*, 2022.

[9] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *Proc. ICML*. Association for Computing Machinery, 2006.

[10] Y. Wang, Z. Chen, C. Zheng, Y. Zhang, W. Han, and P. Haghani, "Accelerating rnn-t training and inference using ctc guidance," 2022. [Online]. Available: https://arxiv.org/abs/2210.16481

[11] A. Graves, "Sequence transduction with recurrent neural networks," *CoRR*, 2012.

[12] A. Jaegle, S. Borgeaud, J. Alayrac, C. Doersch, C. Ionescu, D. Ding, S. Koppula, D. Zoran, A. Brock, E. Shelhamer, O. J. Hénaff, M. M. Botvinick, A. Zisserman, O. Vinyals, and J. Carreira, "Perceiver io: A general architecture for structured inputs & outputs," arXiv, 2021.

[13] J. Alayrac, J. Donahue, P. Luc, A. Miech, I. Barr, Y. Hasson, K. Lenc, A. Mensch, K. Millican, M. Reynolds, R. Ring, E. Rutherford, S. Cabi, T. Han, Z. Gong, S. Samangooei, M. Monteiro, J. Menick, S. Borgeaud, A. Brock, A. Nematzadeh, S. Sharifzadeh, M. Binkowski, R. Barreira, O. Vinyals, A. Zisserman, and K. Simonyan, "Flamingo: a visual language model for few-shot learning," *CoRR*, vol. abs/2204.14198, 2022.

[14] Z. Borsos, R. Marinier, D. Vincent, E. Kharitonov, O. Pietquin, M. Sharifi, O. Teboul, D. Grangier, M. Tagliasacchi, and N. Zeghidour, "Audiolm: a language modeling approach to audio generation," *arXiv preprint arXiv:2209.03143*, 2022.

[15] F. Wu, K. Kim, S. Watanabe, K. Han, R. McDonald, K. Q. Weinberger, and Y. Artzi, "Wav2seq: Pre-training speech-to-text encoder-decoder models using pseudo languages," 2022. [Online]. Available: https://arxiv.org/abs/2205.01086

[16] U. Khandelwal, A. Fan, D. Jurafsky, and L. Z. and M. Lewis, "Nearest neighbor machine translation," in *Proc. ICLR*, 2021.

[17] S. Borgeaud, A. Mensch, J. Hoffmann, T. Cai, E. Rutherford, K. Millican, G. v. d. Driessche, J.-B. Lespiau, B. Damoc, A. Clark, D. d. L. Casas, A. Guy, J. Menick, R. Ring, T. Hennigan, S. Huang, L. Maggiore, C. Jones, A. Cassirer, A. Brock, M. Paganini, G. Irving, O. Vinyals, S. Osindero, K. Simonyan, J. W. Rae, E. Elsen, and L. Sifre, "Improving language models by retrieving from trillions of tokens," *CoRR*, vol. abs/2112.04426, 2021.

[18] G. Izacard, P. Lewis, M. Lomeli, L. Hosseini, F. Petroni, T. Schick, J. Dwivedi-Yu, A. Joulin, S. Riedel, and E. Grave, "Atlas: Few-shot learning with retrieval augmented language models," 2022. [Online]. Available: https://arxiv.org/abs/2208.03299

[19] K. Guu, K. Lee, Z. Tung, P. Pasupat, and M.-W. Chang, "Retrieval augmented language model pre-training," in *Proc. ICML*, vol. 119. PMLR, 2020, pp. 3929–3938.

[20] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-augmented generation for knowledge-intensive nlp tasks," in *Proc. NIPS*, vol. 33, 2020, pp. 9459–9474.

[21] G. Izacard and E. Grave, "Distilling knowledge from reader to retriever for question answering," in *Proc. ICLR*, 2021. [Online]. Available: https://openreview.net/forum?id=NTEz-6wysdb

[22] C.-S. Wu, A. Madotto, E. Hosseini-Asl, C. Xiong, R. Socher, and P. Fung, "Transferable multi-domain state generator for task-oriented dialogue systems," in *Proc. ACL*, Jul. 2019, pp. 808–819.

[23] L. Zhou and K. Small, "Multi-domain dialogue state tracking as dynamic knowledge graph enhanced question answering," *CoRR*, vol. abs/1911.06192, 2019. [Online]. Available: http://arxiv.org/abs/1911.06192

[24] A. Rastogi, X. Zang, S. Sunkara, R. Gupta, and P. Khaitan, "Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset," *Proc. AAAI Conference on Artificial Intelligence*, 2020.

[25] P. Pasupat, Y. Zhang, and K. Guu, "Controllable semantic parsing via retrieval augmentation," in *Proc. EMNLP*, Nov. 2021, pp. 7683–7698.

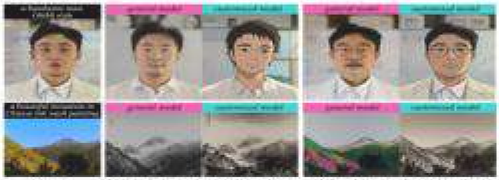
[26] R. Gupta, H. Lee, J. Zhao, Y. Cao, A. Rastogi, and Y. Wu, "Show, don't tell: Demonstrations outperform descriptions for schema-guided task-oriented dialogue," in *Proc. NACCL*. ACL, Jul. 2022.

[27] D. Yu, M. Wang, Y. Cao, L. El Shafey, I. Shafran, and H. Soltau, "Knowledge-grounded dialog state tracking," in *Proc. EMNLP*, 2022.

[28] J. Devlin, M. Chang, K.Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[29] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *The Journal of Machine Learning Research*, 2020.

[30] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," 2018.



(a) input (b) Translation results w/o ControlNet (c) Translation results w/ ControlNet

Figure 2. Customized model and ControlNet generate high-quality results with better consistency with both prompt and content. Our method is designed to be compatible with these existing image diffusion techniques, and thus can take advantage of them to strike a good balance between the style (prompt) and the content.

models. The third solution involves zero-shot methods [20] that require no training. During the diffusion sampling process, cross-frame constraints are imposed on the latent features for temporal consistency. The zero-shot strategy requires fewer computing resources and is mostly compatible with existing image models, showing promising potential. However, current cross-frame constraints are limited to global styles and are unable to preserve low-level consistency, *e.g.*, the overall style may be consistent, but the local structures and textures may still flicker.

Achieving successful application of image diffusion models to the video domain is a challenging task. It requires **1)** Temporal consistency: cross-frame constraints for low-level consistency; **2)** Zero-shot: no training or fine-tuning required; **3)** Flexibility: compatible with off-the-shelf image models for customized generation. As mentioned above, image models can be customized by fine-tuning on specific objects to capture the target style more precisely than general models. Figure 2 shows two examples. To take advantage of it, in this paper, we employ zero-shot strategy for model compatibility and aim to further solve the key issue of this strategy in maintaining low-level temporal consistency.

To achieve this goal, we propose novel hierarchical cross-frame constraints for pre-trained image models to produce coherent video frames. Our key idea is to use optical flow to apply dense cross-frame constraints, with the previous rendered frame serving as a low-level reference for the current frame and the first rendered frame acting as an anchor to regulate the rendering process to prevent deviations from the initial appearance. Hierarchical cross-frame constraints are realized at different stages of diffusion sampling. In addition to global style consistency, our method enforces consistency in shapes, textures and colors at early, middle and late stages, respectively. This innovative and lightweight modification achieves both global and local temporal consistency. Figure 1 presents our coherent video translation results over off-the-shelf image models customized for six unique styles.

Based on the insight, this paper introduces a novel zero-shot framework for text-guided video-to-video translation, consisting of two parts: key frame translation and full video translation. In the first part, we adapt pre-trained image diffusion models with hierarchical cross-frame constraints for generating key frames. In the second part, we propagate the rendered key frames to other frames using temporal-aware patch matching and frame blending. The diffusion-based generation is excellent at content creation, but its multi-step sampling process is inefficient. The patch-based propagation, on the other hand, can efficiently infer pixel-level coherent frames but is not capable of creating new content. By combining these two parts, our framework strikes a balance between quality and efficiency. To summarize, our main contributions are as follows:

- A novel zero-shot framework for text-guided video-to-video translation, which achieves both global and local temporal consistency, requires no training, and is compatible with pre-trained image diffusion models.
- Hierarchical cross-frame consistency constraints to enforce temporal consistency in shapes, textures and colors, which adapt image diffusion models to videos.
- Hybrid diffusion-based generation and patch-based propagation to strike a balance between quality and efficiency.

2. Related Work

2.1. Text Driven Image Generation

Generating images with descriptive sentences is intuitive and flexible. Early attempts explore GAN [42–44, 46] to synthesize realistic images. With the powerful expressivity of Transformer [38], autoregressive models [6, 9, 31] are proposed to model image pixels as a sequence with autoregressive dependency between each pixel. DALL-E [31] and CogView [6] train an autoregressive transformer on image and text tokens. Make-A-Scene [9] further considers segmentation masks as condition.

Recent studies focus on diffusion models [15] for text-to-image generation, where images are synthesized via a gradual denoising process. DALLE-2 [30] and Imagen [34] introduce pretrained large language models [28, 29] as text encoder to better align the image with text, and cascade diffusion models for high resolution image generation. GLIDE [26] introduces classifier-free guidance to improve text conditioning. Instead of applying denoising in the image space, Latent Diffusion Models [32] uses the low-resolution latent space of VQ-GAN [7] to improve the efficiency. We refer to [4] for a thorough survey.

In addition to diffusion models for general images, customized models are studied. Textual Inversion [10] and DreamBooth [33] learn special tokens to capture novel concepts and generate related images given a small number of example images. LoRA [17] accelerates the fine-tuning not part of MiDaS v3.1, because SwinV2 generally yields better results than Swin. Finally, we have also studied the MobileViTv2 family of transformers, which contains MobileViTv2-0.5 as our smallest model with 13 million parameters. However, both variants MobileViTv2-0.5 and MobileViTv2-2.0 have values of I around -300%, which reflects a too low quality to be relevant.

As the models below the horizontal separator of Tab. 2 are explained in Sec. 4.3, we proceed with the models between the first and last horizontal separator of Tab. 3. The models shown there split into models with transformer and convolutional encoder backbones, which are separated by the dashed separator. We start with the transformer models, where we first have DeiT3-L-22K-1K and DeiT3-L. These two models have a high depth estimation quality, *e.g.*, 0.070 for the relative error (REL) of the BlendedMVS dataset, which is equal to the value of BEiT$_{384}$-L also visible in Tab. 2 for a comparison. However, as the DeiT3 transformers do not surpass the quality of BEiT$_{384}$-L, we did not train them beyond the first stage. The same criterion holds for ViT-L Hybrid, which was explored, because ViT-B Hybrid is part of MiDaS v3.0 (cf. Tab. 1). For Next-ViT-L-1K and Next-ViT-L-1K-6M, we have decided to include the better of the two variants in MiDaS v3.1, which is Next-ViT-L-1K-6M according to Tab. 3.

Finally, we have also explored the three convolutional models ConvNeXt-XL, ConvNeXt-L and EfficientNet-L2. As we explored them with the intention to get a model of highest quality and it did not beat BEiT$_{384}$-L, we have discarded these models. In particular, EfficientNet-L2 shows a low depth estimation quality with errors of 0.165, 0.227 and 0.219 according to Tab. 3.

4.3. Ablation Studies

In the following, we discuss experimental modifications of some of the investigated backbones, which helps to get a better understanding of the associated configurations. The modifications can be found at the bottom of Tabs. 2 and 3. In addition to that, we also walk through the models at the top of Tab. 2, which are included for a comparison with the other models in that table.

We begin with the four reference models at the top of Tab. 3. Variants of these models are also available in Tab. 1. For BEiT$_{384}$-L and Next-ViT-L-1K-6M, these are models with different training datasets, *i.e.* 3+10 in Tab. 3 and 5+12 in Tab. 1. For Swin-L, no such difference is given between the two tables. However, in Tab. 3, we have included two separate training runs to provide an approximation of the variance in the training process. ViT-L is basically the same model in both tables, but the training runs are independent, because a retraining was required to get the data required for Tab. 3.

We continue with the two experimental modifications at the bottom of Tab. 3, which have undergone only one training stage. The first modification, denoted as ViT-L Reversed, is the vanilla vision transformer backbone ViT-L already released in MiDaS v3.0, but with the order of the hooks reversed. Instead of providing the depth decoder hooks with the absolute positions 5, 11, 17, 23, we set them to 23, 17, 11, 5. This is possible, because the ViT encoder family is based on a series of similar transformer blocks, which do not differ like the transformer blocks in for instance the hierarchical structure of the Swin transformers. Astonishingly, as shown in Tab. 3, the reversal of the hooks has practically no impact on the depth estimation quality. So, there is no major difference if the four hierarchy levels of the decoder are connected in forward or reverse order to the transformer blocks of the encoder.

The second experiment is Swin-L Equidistant where the hooks are chosen as equidistantly as possible, similar to ViT-L. As we consider a Swin transformer here, the hook positions are relative and constrained to 0-1, 0-1, 0-17, 0-1 (cf. Sec. 3.2). To homogenize the distance between the hooks, we replace the positions 1, 1, 17, 1 of Swin-L by 1, 1, 9, 1. Note that the distances could be made even more similar by setting the first hook to zero. However, here we follow ViT-L, where a gap is chosen before the first hook. As we see from Tab. 3, the modification leads to a small decrease of the depth estimation quality when compared to the unmodified model Swin-L such that we have not released the corresponding model. To also get at least a very rough estimate of the significance of this change, we have actually included two independent training runs for Swin-L, denoted by training 1 and 2 in Tab. 3. As we see, the training variance seems to be rather small for Swin-L.

Tab. 2 shows four additional modifications, where we have also trained the second stage. We first consider the model BEiT$_{384}$-L Wide, where the hooks are widened by removing the hook gap at the beginning of the encoder. Instead of the absolute hook positions 5, 11, 17, 23 of BEiT$_{384}$-L in Tab. 1 (see Sec. 3.2), the modification uses 0, 7, 15, 23. As we see from Tab. 2, there is nearly no impact on the depth estimation quality. For unconstrained resolutions, the relative improvement I is 17.4% for the widened variant and thus a bit better than the value 16.8% for the original variant in Tab. 1. For square resolutions, the situation is the opposite, where we have the values 32.7% and 33.0%. With the effect being so small, we have decided to keep the hook gap.

The remaining three modifications in Tab. 2, denoted as BEiT$_{384}$-L 5+12+12K, BEiT$_{384}$-L 5K+12K and BEiT$_{384}$-L 5A+12A, address the large value $\delta_1 = 9.847$ of KITTI for the unconstrained resolution of BEiT$_{384}$-L when compared to $\delta_1 = 2.212$ of NYU Depth v2 in Tab. 1. The reason for the large δ_1 value is that the training images of KITTI have a high aspect ratio caused by the resolution 1280x384, where

Table 2: Performance on GSM8K dataset.

Method	Sampled paths	Accuracy(%)
GPT-3.5 (5-shot)	-	57.1
GPT-4 (5-shot CoT)	-	92.0
GPT-4 (PHP)	40	96.5
GPT-4 (Model selection)	15	96.8
GPT4-Code	-	92.9
GPT4-Code + **CSV** + **Voting**	**5**	**97.0**

Table 3: Performances on MMLU dataset.

Method	Dataset	Accuracy(%)	Few-shot
Chinchilla (Hoffmann et al., 2022)	Math	35.7	5-shot
Galactica (Taylor et al., 2022)	Math	41.3	5-shot
GPT4-Code	Math	87.5	zero-shot
GPT4-Code + **CSV** + **Voting**	Math	**89.2**	**zero-shot**
LLaMA 2	STEM	58.0	5-shot
OpenLLM	STEM	70.6	5-shot
GPT-4	STEM	82.7	zero-shot
GPT4-Code	STEM	86.8	zero-shot
GPT4-Code + **CSV** + **Voting**	STEM	**87.0**	**zero-shot**

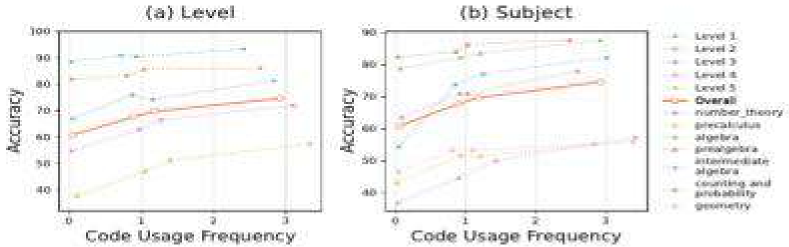


Figure 5: The four points on each curve correspond to results using **Prompt 1**, **Prompt 2**, **Basic Prompt** and **Code-based Self-verification Prompt**, respectively. **(a)** The accuracy of different levels at various code usage frequencies. **(b)** The accuracy of different subjects at various code usage frequencies.

4-code, our method outperforms other methods in the competition, achieving state-of-the-art results across all datasets. Other subjects in MMLU benchmarks are provided in Fig. 8. A comparative analysis of our results with those of previous state-of-the-art techniques and open-source models is also provided.

Tab. 2 illustrates that verification-guided majority voting is an effective framework to reduce the number of sampled paths, compared to GPT-4 with model selection (Zhao et al., 2023) and PHP (Zheng et al., 2023).

Tab. 3 presents a comparison of our model's performance with existing models (Hoffmann et al., 2022; Taylor et al., 2022) on the MMLU-Math dataset and with state-of-the-art open-sourced models[4] on MMLU-STEM. The open-source models remain significantly outpaced by their closed-source counterparts. To address this gap, we have made the dataset and will make it publicly available in the near future. Our intention is to facilitate the fine-tuning of open-source LLMs. For example, the open-source model LLaMA 2 (Touvron et al., 2023) can potentially utilize this data to further bolster its math reasoning capabilities.

4.3 Code usage frequency of proposed prompts

Analogous to the approach taken in Sec. 3.1, we gather data to elucidate the correlation between accuracy and Code Usage Frequency across various dimensions - prompts (proposed CSV prompt as well as prompts used in pilot experiments), subjects, and difficulty levels. As shown in Fig. 5, the model's behavior is in accordance with our expectations when adding the code-based prompts. Each line in Fig. 5 has an obvious trend of going upwards, proving that the increase of Code Usage Frequency induces a general improvement in accuracy. The performance gain when using more code is more obvious in the higher difficulty levels, while in lower levels, the performance gain is not very prominent, as shown in Fig. 5 (a). Also, the Code Usage Frequency increases steadily with

[4]https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard

(Mildenhall et al., 2020). To better model geometry, we adopt the MLP architecture and signed distance field formulation from VolSDF (Yariv et al., 2021) that defines density function as Laplace's cumulative distribution function applied to SDF. We refer readers to the supplementary for the results with the NeRF backbone and further implementation details.

Following (Wang et al., 2021b), all models are supervised by minimizing the pixel-wise difference between the rendered and ground truth colors (l_1 error), the rendered opacity and the gt mask (binary cross-entropy), and further adopting the Eikonal (Gropp et al., 2020) mean squared error loss for well-behaved surface reconstruction under the sparse capture setup:

$$\mathcal{L} = \mathcal{L}_{color} + \lambda_1 \mathcal{L}_{igr} + \lambda_2 \mathcal{L}_{mask}. \tag{5}$$

We use four sequences from the Owlii (Xu et al., 2017) dataset to evaluate the methods. Compared to fully synthetic sequences previously utilized for the task (Pumarola et al., 2021), the dynamic Owlii sequences exhibit more rapid and complex high-frequency motions, making it a harder task for MLP-based methods. At the same time, the presence of ground truth 3D scans allows us to evaluate both geometry and appearance reconstruction quality, as compared to the sequences with only RGB data available (Li et al., 2022; Shao et al., 2023). We render 400 RGB training images from four static camera views from 100 frames/time intervals and 100 test images from a rotating camera from 100 frames. We report L1 Chamfer distance (CD↓) (scaled by 10^3) and the standard image-based metrics (PSNR↑, SSIM↑).

We benchmark recent state-of-the-art methods and their variations implemented with ResField layers of rank ten ($R_i = 10$) – TNeRF (Pumarola et al., 2021; Li et al., 2022), DyNeRF (Li et al., 2022), DNeRF (Pumarola et al., 2021), Nerfies (Park et al., 2021a), HyperNeRF (Park et al., 2021b), NDR (Cai et al., 2022), and HexPlane (Cao & Johnson, 2023; Fridovich-Keil et al., 2023) – as well as a recent timespace-partitioning methods Tensor4D (Shao et al., 2023) and NeuS2 (Wang et al., 2023d) (with default training configurations). Please see the Sup. Mat. for further details.

Insights. We report all quantitative and qualitative results in Tab. 3 and Fig. 5. Results demonstrate that our method consistently improves all baseline methods, achieving new state-of-the-art results for sparse multi-view reconstruction of dynamic scenes. We further observe that more ResField layers gradually improve results until the point of saturation ($i = 1, 2, 3$). This experiment confirms that increasing the modeling capacity to a more-than-needed level does not cause overfitting. Importantly, the simplest/cheapest baseline method TNeRF implemented with ResFields performs better than every other more expensive baseline method in the original form. We believe that such speedup and lower memory requirements are of great benefit to the research community, as they enable the use of lower-end hardware for high-fidelity reconstructions. Given this observation, we set up a simple camera rig and captured longer and more complex sequences to better understand the limitations.

Lightweight capture from three RGBD views. We capture four sequences (150 frames) via synchronized Azure Kinects (three for reconstruction and one for validation) and compare TNeRF (w. depth supervision), a baseline with a good balance between computational complexity and accuracy, and its enhancement with ResFields applied to all middle layers. Quantitative evaluation in terms of mean SSIM↑ and LPIPS↓ (Zhang et al., 2018) reported in Tab. 4 demonstrates that ResFields consistently benefits the reconstruction (see visuals in Fig. 1 and the Sup. video). However, we observe that both methods struggle to capture thin and tiny surfaces such as the cord of sunglasses.

Table 4: **Lightweight capture from three RGBD views.**

	Mean		Book		Glasses		Hand		Writing	
	LPIPS↓	SSIM↑	LPIPS	SSIM	LPIPS	SSIM	LPIPS	SSIM↑	LPIPS	SSIM
TNeRF	0.234	79.16	0.323	68.85	0.206	80.44	0.239	81.30	0.168	86.08
+ResFields	**0.203**	**80.00**	**0.284**	**70.84**	**0.164**	**80.65**	**0.210**	**82.09**	**0.155**	**86.43**

4.4 Scene Flow

Scene flow models a 3D motion field for every point in space x and time t. We take the same four Deforming Things sequences from Sec 4.2 and learn bi-directional scene flow. We use 80% of tracked mesh vertices to learn the flow and the remaining 20% for evaluation. As a supervision, we use l_1 error between the predicted and the ground truth flow vectors. We consider three motion models that predict 1) offset vectors (Prokudin et al., 2023; Li et al., 2021b), 2)

Table 5: **Scene flow.**

		it/s ↑	type	fwd/bwd l_1 ↓
[illegible]	ReLU MLP	16.5	offset	6.88 / 7.31
	+ResFields			**3.85 / 3.85**
	ReLU MLP		SE(3)	3.57 / 3.58
	+ResFields			**2.64 / 2.56**
	ReLU MLP		DCT	3.19 / 3.19
	+ResFields			**2.18 / 2.19**
[illegible]	ReLU MLP	5.6	offset	6.50 / 6.44
	+ResFields			**4.43 / 4.59**
	ReLU MLP		SE(3)	3.00 / 3.10
	+ResFields			**2.88 / 2.84**
	ReLU MLP		DCT	3.47 / 3.48

separate reward models during training, we compute critique offline and directly insert them into the training corpus, where the generator LM is trained with a standard LM objective. This significantly reduces training costs compared to PPO. Our work also relates to prior work that incorporates special tokens to control generation (Keskar et al., 2019; Lu et al., 2022; Korbak et al., 2023). Our SELF-RAG learns to generate special tokens *to evaluate its own prediction* after each generated segment, enabling the use of a soft re-ranking mechanism or hard constraints at inference (discussed next).

3.3 SELF-RAG INFERENCE

Generating reflection tokens to self-evaluate its own output makes SELF-RAG controllable during the inference phase, enabling it to tailor its behavior to diverse task requirements. For tasks demanding factual accuracy (Min et al., 2023), we aim for the model to retrieve passages more frequently to ensure that the output aligns closely with the available evidence. Conversely, in more open-ended tasks, like composing a personal experience essay, the emphasis shifts towards retrieving less and prioritizing the overall creativity or utility score. In this section, we describe approaches to enforce control to meet these distinct objectives during the inference process.

Adaptive retrieval with threshold. SELF-RAG dynamically decides when to retrieve text passages by predicting Retrieve. Alternatively, our framework allows a threshold to be set. Specifically, if the probability of generating the Retrieve=Yes token normalized over all output tokens in Retrieve surpasses a designated threshold, we trigger retrieval (details in Appendix Section A.3).

Tree-decoding with critique tokens. At each segment step t, when retrieval is required, based either on hard or soft conditions, $\mathcal{R}$ retrieves K passages, and the generator $\mathcal{M}$ processes each passage in parallel and outputs K different continuation candidates. We conduct a segment-level beam search (with the beam size=B) to obtain the top-B segment continuations at each timestamp t, and return the best sequence at the end of generation. The score of each segment y_t with respect to passage d is updated with a critic score $\mathcal{S}$ that is the linear weighted sum of the normalized probability of each Critique token type. For each critique token group G (e.g., IsRel), we denote its score at timestamp t as s_t^G, and we compute a segment score as follows:

$$f(y_t, d, \text{Critique}) = p(y_t | x, d, y_{<t})) + \mathcal{S}(\text{Critique}), \text{ where} \tag{3}$$

$$\mathcal{S}(\text{Critique}) = \sum_{G \in \mathcal{G}} w^G s_t^G \text{ for } \mathcal{G} = \{\text{IsRel}, \text{IsSup}, \text{IsUse}\}, \tag{4}$$

where $s_t^G = \frac{p_t(\hat{r})}{\sum_{i=1}^{N^G} p_t(r_i)}$ stands for the generation probability of the most desirable reflection token $\hat{r}$ (e.g., IsRel =Relevant) for the critique token type G with N^G distinct tokens (that represent different possible values for G). The weights w^G in Eq. 4 are hyperparameters that can be adjusted at inference time to enable customized behaviors at test time. For instance, to ensure that result y is mostly supported by evidence, we can set a weight term for the IsSup score higher, while relatively lowering weights for other aspects. Alternatively, we could further enforce hard constraints during decoding using Critique. Instead of using a soft reward function in Eq. 4, we could explicitly filter out a segment continuation when the model generates an undesirable Critique token (e.g., IsSup =No support) . Balancing the trade-off between multiple preferences has been studied in RLHF (Touvron et al., 2023; Wu et al., 2023), which often requires training to change models' behaviors. SELF-RAG tailors an LM with no additional training.

4 EXPERIMENTS

4.1 TASKS AND DATASETS

We conduct evaluations of our SELF-RAG and diverse baselines on a range of downstream tasks, holistically evaluating outputs with metrics designed to assess overall correctness, factuality, and fluency. Throughout these experiments, we conduct zero-shot evaluations, where we provide instructions describing tasks without few-shot demonstrations (Wei et al., 2022; Sanh et al., 2022). Details of our experiments' settings, including test-time instructions, are available in the Appendix Section B.1.

Closed-set tasks include two datasets, i.e., a fact *verification dataset* about public health (**PubHealth**; Zhang et al. 2023) and a *multiple-choice reasoning dataset* created from scientific exams (**ARC-**

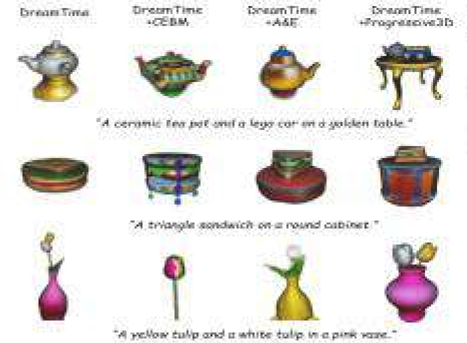

DreamTime
DreamTime +CEBM
DreamTime +A&E
DreamTime +Progressive3D
"A ceramic tea pot and a lego car on a golden table."
"A triangle sandwich on a round cabinet."
"A yellow tulip and a white tulip in a pink vase."

Figure 6: Visual comparison with DreamTime-based compositional generation baselines.

Table 1: Quantitative comparison on metrics and user studies over CSP-100.

Method	Metrics		Human
	B-VQA ↑	mGPT-CoT ↑	Preference ↑
DreamTime	0.227	0.522	16.8%
+ CEBM	0.186	0.491	-
+ A&E	0.243	0.528	-
+ Progressive3D	**0.474**	**0.609**	**83.2%**

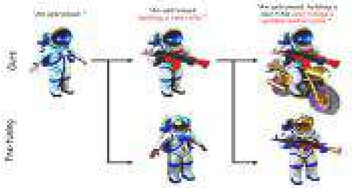


Figure 7: Qualitative ablations between fine-tuning with target prompts and editing with Progressive3D on MVDream.

including BLIP-VQA and mGPT-CoT (Huang et al., 2023a), evaluate the generation capacity of current methods and our Progressive3D when handling prompts with complex semantics.

Baselines. We incorporate our Progressive3D with 4 text-to-3D methods driven by different 3D representations: **(1)** DreamTime (Huang et al., 2023b) is a NeRF-based method which enhances DreamFusion (Poole et al., 2022) in time sampling strategy and produce better results. We adopt DreamTime as the main baseline for quantitative comparisons and ablations due to its stability and training efficiency. **(2)** TextMesh (Tsalicoglou et al., 2023) leverages SDF as the 3D representation to improve the 3D mesh extraction capacity. **(3)** Fantasia3D (Tsalicoglou et al., 2023) is driven by DMTet which produces impressive 3D content with a disentangled modeling process. **(4)** MVDream (Shi et al., 2023) is a NeRF-based method which leverages a pre-trained multi-view consistent text-to-image model for text-to-3D generation and achieves high-quality 3D content generation performance. To further demonstrate the effectiveness of Prgressive3D, we re-implement two composing text-to-image methods including Composing Energy-Based Model (CEBM) (Liu et al., 2022) and Attend-and-Excite (A&E) (Chefer et al., 2023) on DreamTime for quantitative comparison.

4.2 Progressive3D for Text-to-3D Creation and Editing

Comparison with current methods. We demonstrate the superior performance of our Progressive3D compared to current text-to-3D methods in both qualitative and quantitative aspects in this section. We first present visualization results in Fig. 4 to verify that DreamTime faces significant challenges including (a) object missing, (b) attribute mismatching, and (c) quality reduction when given prompts describe multiple interacted objects binding with different attributes. Thanks to our careful designs, Progressive3D effectively promotes the creation performance of DreamTime when dealing with complex prompts. In addition, more progressive editing processes based on various text-to-3D methods driven by different neural 3D representations are shown in Fig. 5, which further demonstrate that Progressive3D stably increases the generation capacity of based methods when given prompts are complex, and our framework is general for various current text-to-3D methods.

We also provide quantitative comparisons on fine-grained semantic consistency metrics including BLIP-VQA and mGPT-CoT, and the results are shown in Tab. 1, which verify that our Progressive3D achieves remarkable improvements for 3D content creation with complex semantics compared to DreamTime-based baselines. As shown in Fig. 6, baselines that combine 2D composing T2I methods including CEBM (Liu et al., 2022) and A&E (Chefer et al., 2023) with DreamTime

Figure 2 shows the workflow of our proposed system. Loop Copilot comprises 5 key components:

(1) The **large language model ($\mathcal{M}$)** for understanding and reasoning;
(2) The **system principles ($\mathcal{P}$)** that provide basic rules to guide the large language model;
(3) A list of **backend models ($\mathcal{F}$)** responsible for executing specific tasks;
(4) A **global attribute table ($\mathcal{T}$)** that maintains crucial information to ensure continuity throughout the creative process;
(5) A **framework handler ($\mathcal{D}$)** that orchestrates the interactions between these components.

Therefore, we can represent Loop Copilot as:

$$\text{LoopCopilot} = (\mathcal{M}, \mathcal{P}, \mathcal{F}, \mathcal{T}, \mathcal{D}).$$

The workflow of Loop Copilot involves several steps.

(1) **Input preprocessing.** the system processes the input by unifying the modality of the input. The framework handler $\mathcal{D}$ utilizes a music captioning model to describe the input music, while textual inputs are kept as they are.
(2) **Task analysis.** the framework handler performs task analysis if the text input contains an explicit demand. It calls the large language model $\mathcal{M}$ to analyze the task, resulting in a sequence of steps, which may involve a call to a single model or multiple chained calls to models, as the large language model may need to handle the task step by step. Section 3.2 demonstrates the details.
(3) **Task execution.** After task analysis, the framework handler records all the steps and proceeds to execute the tasks. It calls the backend models in the specified order, providing them with the necessary parameters obtained from the large language model. If it requires a chained call of multiple models, the intermediate results generated by the previous model will be used in the next model.
(4) **Response generation.** Once the task execution is complete, the handler $\mathcal{D}$ collects the final result and sends it to the large language model for the final output.

Throughout this process, all operations are tracked and recorded in the global attribute table $\mathcal{T}$, ensuring consistency and continuity in the generation process. We will demonstrate it in detail in Section 3.3. Algorithm 1 illustrates the process during a T-round dialogue.

3.2 Supported Tasks

The interaction process within Loop Copilot is essentially a two-stage workflow, as illustrated in Fig. 1 and Fig. 2. The first stage involves the user drafting a music loop, while the second stage is dedicated to iterative refinement through dialogue. Each stage necessitates different tasks. In the initial stage, the focus is on creating music from an ambiguous demand, essentially a requirement for global features. The second stage shifts the focus to music editing, where fine-grained localized revisions are made. These revisions can include regenerating specific areas, adding or removing particular instruments, and incorporating sound effects. A comprehensive list of all supported tasks is presented in Table 1.

Each task in Table 1 corresponds to one or more specific backend models, which are sequentially called. For instance, consider the task "impression to music". Here, a user can reference the title of a real-world music track. Loop Copilot first invokes ChatGPT to translate this music title into a music description, which is then forwarded to MusicGen to generate the music audio. This ability to chain multiple models opens up a wealth of opportunities to accomplish new tasks that have scarcely been explored before, although the results may not be as good as models trained for specific tasks.

Specifically, we explore new methods for the below tasks:

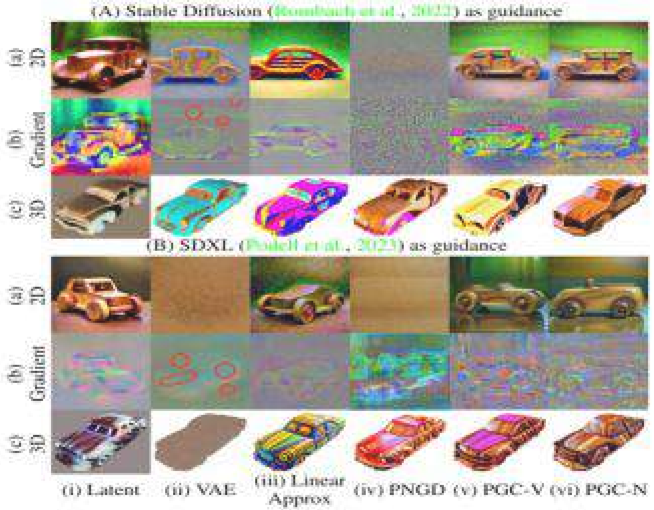


Figure 2: **Visualization of 2D/3D results and typical gradients guided by different LDMs. (A)** Stable Diffusion 2.1-base (Rombach et al., 2022) as guidance. **(B)** SDXL (Podell et al., 2023) as guidance. The text prompt is *a wooden car*. For each case, we visualize (a) directly optimizing a 2D image using SDS loss, alongside (b) the corresponding gradients; (c) optimizing a texture field (Chen et al., 2023b) based on a fixed mesh of car. We compare six gradient propagation methods: (i) Backpropagation of latent gradients, (ii) VAE gradients, (iii) linear approximated VAE gradients, (iv) normalized VAE gradients, (v) our proposed PGC VAE gradients by value and (vi) by norm. ○ highlights gradient noise.

latent variables. We assume a linear relationship between RGB pixels and latent variables, which allows for explicit gradient control. This control is achieved by applying L2-norm constraints to the projection matrix's norm during the training process.

To elaborate, when dealing with an RGB pixel vector $\boldsymbol{x} \in \mathbb{R}^3$ and a latent variable vector $\boldsymbol{y} \in \mathbb{R}^4$, we establish the relationship as follows:

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}, \tag{5}$$

where $\boldsymbol{A} \in \mathbb{R}^{4\times3}$ and $\boldsymbol{b} \in \mathbb{R}^4$ serve as analogs to the VAE parameters.

For evaluation, we use ridge regression methods with the COCO dataset (Lin et al., 2014) to determine the optimal configuration. For optimizing SDS, we approximate the term $\partial z/\partial x$ using the transposed linear matrix $\boldsymbol{A}^\top$. This matrix is regulated through ridge regression, enabling controlled gradient behavior. Nevertheless, as illustrated in the appendix, this attempt to approximate the VAE with a linear projection falls short. This linear approximation cannot adequately capture fine texture details, thus compromising the preservation of crucial texture-related gradients.

4.2 Score Distillation Sampling process regulation

As previously discussed in Section 3.2, parameter-wise gradient regularization techniques are commonly employed in neural network training. Additionally, we observe that regulating gradients at the pixel level plays a crucial role in managing the overall gradient during the SDS process.

Ben Wang and Aran Komatsuzaki. 2021. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. https://github.com/kingoflolz/mesh-transformer-jax.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. An explanation of in-context learning as implicit bayesian inference. In *International Conference on Learning Representations*.

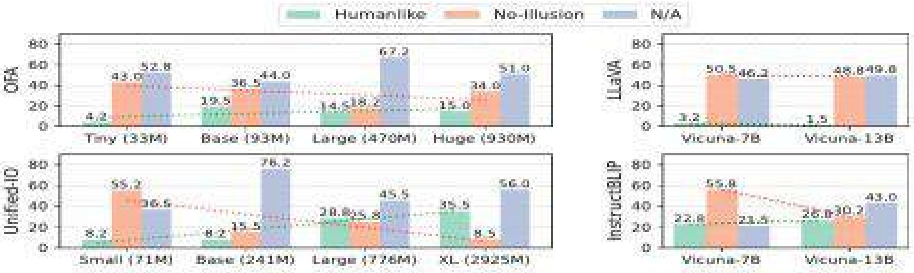


Figure 5: Results of *SameDiffQA*. The number shows the percentage of the answers. Each cluster represents the distribution over humanlike, no-illusion and N/A answers from a model. The green and red line correspond to the linear regression of humanlike rate and no-illusion rate across all the model sizes. Except for OFA-Large, Unified-IO-Large, InstructBLIP-13B, the differences between the humanlike rate and the no-illusion rate are statistically significant $P < 0.005$. Details are in Table 4 Appendix A.

mentioned models, there exists a range of variants in different sizes: OFA-{Tiny, Base, Large, Huge}, Unified-IO-{Small, Base, Large, XL}, LLaVA-{Vicuna-7B, Vicuna-13B}, InstructBLIP-{Vicuna-7B, Vicuna-13B}. This allows us to study the impact of size variations on model's understanding of visual illusions.

Metrics. Through the experiments, we keep track of the **Humanlike** rate to measure the alignment between humans and VLMs, which is the percentage of examples where the machine gives exactly the same answers as humans. For the *SameDiffQA* task, we also compute the **No-Illusion** rate, which corresponds to the percentage of examples where the machine consistently considers the objects as the same under both illusion and illusion-free settings. For examples where the model fails to identify the objects as the same in the illusion-free image or produces nonsense answers to the questions, we mark them as **Not Applicable (N/A)** and exclude them from the illusion recognition assessment.

5 Results Analysis

From our experiments, we are interested in investigating the following research questions:

- RQ1: to what extent do VLMs recognize the presence of illusions similar to humans?
- RQ2: how much do VLMs align with humans when communication happens under the influence of illusions?
- RQ3: does the degree of alignment between VLMs and human responses vary across different categories of illusions?

We highlight several of our findings across this three questions in below.

5.1 Illusion Recognition

The results of *SameDiffQA* are shown in Figure 5. Relative proportions of "humanlike," "no-illusion," and "not applicable (N/A)" responses are represented as green, orange, and grey bars respectively for each model, which all together account for 100%. First of all, we notice a large percentage of responses, across all models, fall under the N/A category. This suggests that these models often cannot even tell that the objects are identical in the illusion-free image, underscoring the need for improvement in standard vision-language reasoning capabilities beyond the scope of illusion contexts.

Given the high proportion of N/A responses, one might question the benchmark's adequacy in reliably reflecting a model's tendency towards either "humanlike" or "no-illusion". Excluding the N/A responses, we employed a χ^2-test and found that 9 out of 12 models would reject the null hypothesis which posits that the "humanlike" or "no-illusion" responses are uniformly distributed. In other words, these models do not behave randomly. Refer to Appendix A for more details. Such findings indicate that, despite certain limitations in their capabilities, our dataset and experimental design effectively

	Downstream Task Metric Dataset Size	AMP	MNLI Acc. 392.7k	QQP Acc. 363.8k	QNLI Acc. 104.7k	SST-2 Acc. 67.3k	CoLA Matthew 8.5k	STSB Pearson 5.7k	 Spear 	MRPC F1 3.7k	 Acc 	RTE Acc. 2.5k	Average Acc. -
Teacher	BERT (109M)		84.17	90.89	90.68	91.86	57.54	88.84	88.56	89.31	85.04	65.34	83.23
Student (67M)	One-way KD (Sanh et al., 2019b)	FP32	**81.93**	90.05	87.72	90.94	52.03	86.28	86.07	87.94	82.59	57.76	80.33
	Ours: Student 1	FP16	81.88	**90.17**	88.24	**91.81**	54.82	**86.70**	**86.49**	89.76	**85.29**	**59.21**	**81.40**
	Ours: Student 2	FP16	81.34	89.75	**88.37**	90.71	**56.08**	86.42	86.44	**89.80**	**85.29**	59.20	81.34

Table 2: **Efficacy of Community KD** The pre-trained BERT and 6-layer BERT is the teacher model and student architecture, respectively, for both ours and the conventional one-way KD method. We fine-tune the distilled students on dev sets of GLUE benchmark. We observe that learning from the soft knowledge of different student model improves performance over the conventional one-way KD method on most downstream tasks.

2019a). To ensure a fair comparison, we use the pre-trained BERT-base as the teacher model for both methods and the 6-layer BERT as the student, which is the same architecture used in the conventional one-way KD method. As described in Section 4, we train two student models concurrently and they learn from the pre-trained BERT, the ground truth labels, and each other's knowledge. Note that since we fine-tune **one** of the two students distilled by Community KD for downstream tasks, the inference cost is the same as the conventional one-way KD method. In Table 2, we report the results of BERT and the conventional one-way KD method using checkpoints provided by Hugging Face (HuggingFace) and both students (Ours: Student 1 and Ours: Student 2) on the dev sets of the GLUE benchmark. We apply Automatic Mixed Precision (AMP) (Paszke et al., 2019) to Community KD, which typically speeds up training but may hurt performance.

Results The results presented in Table 2 shows that Community KD, leads to improved performance on downstream tasks such as QQP, QNLI, SST-2, CoLA, STSB, MRPC, and RTE, even when applying quantization techniques. Specifically, the average performance gain of the student model distilled using our Community KD method is 1.04 (1.2%) higher than that of the student model distilled by the conventional one-way KD method. This suggests that incorporating knowledge distillation from both a student model and a pre-trained teacher model is more effective than only using knowledge distillation from the pre-trained teacher model.

6 Limitations & Future Work

Limitations The proposed method co-train models from scratch and may require a longer pre-training time than the conventional KD method. However, as we described in Section 5.3, when the student model is trained long enough with its teacher, it can outperform the models trained with the conventional KD on the downstream task. The proposed co-training method may increase the overall training cost compared with one-way distillation, and it may become a performance bottleneck depending on training resource constraints. However, note that CTCD can improve model quality while having the same inference cost as the one-way distillation on downstream tasks.

Future Work **1) Architecture Sharing.** Models can share some of their architectures by reusing the output of such architectures and updating them together during back-propagation. This may help reduce the additional computing and memory overhead incurred by model co-training, while improving the model quality, especially for the student model. **2) Integration of Student Architecture Design and Data Augmentation.** Future research can focus on effectively combining the CTCD framework with student architecture design and data augmentation techniques. This integration provides a promising alternative to traditional one-way knowledge distillation methods, leading to significant improvements in language modeling tasks.

7 Conclusion

The size and complexity of pre-trained language models (PLMs) can hinder their practicality for online downstream tasks. To address this, we introduced a novel framework called co-training and co-distillation (CTCD). By training models of different sizes together and extracting inter-model knowledge in both directions, the proposed CTCD framework improves both model efficiency and performance. The proposed framework overcomes the trade-off between efficiency and performance in traditional one-way knowledge distillation methods. Notably, our compressed models achieved an impressive gain of 1.66 on the GLUE benchmark, outperforming large models trained using standalone methods.

the models without fine-tuning. Upon evaluation, the fine-tuned CLAP model exhibits the most distinctive distribution spread compared to other models. Manual listening evaluations of matched pairs from all models further confirm that the fine-tuned CLAP and PANNs consistently produce pairs that match with human auditory perception. In conclusion, we use the fine-tuned CLAP in our copy detection analysis in the main text.

LLM begins to perform next-token-generation.

5.2 Model Training

We train our G-LLaVA in two phases, namely 1) geometric visual-language alignment, and 2) geometric instruction tuning. In both phases, we leverage the conventional language modeling loss, which can be formulated as follows:

$$\mathcal{L}(S_{tar}, S_{in}, I) = -\sum_{t=1}^{L} \log p \left[S_{tar}^{t} | \mathcal{F}(s_{tar}^{(<t)}, S_{in}, I) \right] \quad (1)$$

where $\mathcal{F}$ represents the model. I represents the geometric figure; S_{tar} and S_{in} represent the target and input sentences, respectively; S_{tar}^{t} denotes the t^{th} token of target output, and L stands for length.

6 Experiments

6.1 Setup

Dataset. We generate the alignment data and instruction data utilizing training set of GeoQA+ (Cao and Xiao, 2022) and Geometry3K (Lu et al., 2021). More specifically, the contrastive question-answer (QA) pairs in the alignment data are generated using Geometry3K, which features human-labeled logical forms. Note that GeoQA+ covers the training set of GeoQA (Chen et al., 2021), and share the same val/test set as GeoQA (Chen et al., 2021). More details of data split on GeoQA and GeoQA+ is listed in Table 9. Our approach results in 60K alignment data samples, and more than 110K instruction data samples.

We compare our model with other MLLMs on the geometry problems on the minitest split MathVista (Lu et al., 2023), and compare our model with traditional in-domain model on the test split of GeoQA following (Chen et al., 2022; Liang et al., 2023). The geometry problems in MathVista minitest set is collected from four source datasets Geometry3K (Lu et al., 2021), GeoQA+ (Cao and Xiao, 2022), GEOS (Seo et al., 2015) and UniGeo (Chen et al., 2022).

Implementation Details. We employ ChatGPT (gpt-3.5-turbo-0613) for data generation. A detailed description of our prompts will be provided in the appendix. We use LLaVA (Liu et al., 2023) as our backbone. More specifically, we utilize LLAMA-2 (Touvron et al., 2023) as the language model and employ the visual encoder of a pre-trained vision transformer (Radford et al., 2021) (ViT). The resolution of the input image is 336 by 336. We conduct experiments with both 7B and 13B LLMs. In the cross-modal alignment process, only the projection linear layer is trainable. During the instruction tuning phase, both the projection linear layer and the language model are trainable.

For training data, as we found the minitest split of MathVista contains some examples of Mix-train.pk of GeoQA+, we remove those samples

Model	Input	Accuracy (%)
Heuristics Baseline		
Random Chance	-	21.6
Frequent Guess	-	34.1
Human	Q, I	48.4
Close Source Model		
Text-Only LLMs		
2-shot CoT Claude-2	Q	29.8
2-shot CoT ChatGPT	Q	36.5
2-shot CoT GPT-4	Q	44.7
2-shot PoT ChatGPT	Q	30.8
2-shot PoT GPT-4	Q	33.2
Visual-Augmented LLMs		
2-shot CoT Claude-2	Q, I_c, I_t	31.7
2-shot CoT ChatGPT	Q, I_c, I_t	29.3
2-shot CoT GPT-4	Q, I_c, I_t	31.7
2-shot PoT ChatGPT	Q, I_c, I_t	26.4
2-shot PoT GPT-4	Q, I_c, I_t	39.4
Multimodal LLMs		
Multimodal Bard	Q, I	47.1
Gemini Nano 1	Q, I	21.6
Gemini Nano 2	Q, I	23.6
Gemini Pro	Q, I	40.4
Gemini Ultra	Q, I	56.3
GPT4-V	Q, I	50.5
Open Source Model		
IDEFICS (9B-Instruct)	Q, I	21.1
mPLUG-Owl (LLaMA-7B)	Q, I	23.6
miniGPT4 (LLaMA-2-7B)	Q, I	26.0
LLaMA-Adapter-V2 (7B)	Q, I	25.5
LLaVAR	Q, I	25.0
InstructBLIP (Vicuna-7B)	Q, I	20.7
LLaVA (LLaMA-2-13B)	Q, I	29.3
G-LLaVA-7B	Q, I	53.4
G-LLaVA-13B	Q, I	**56.7**

Table 8: Comparison of model performance on the testmini set of MathVista benchmarks (Lu et al., 2023) on geometry problem solving (GPS). For input, Q represents for question, I represents for image, I_c represents for image caption generated by Bard, and I_t represents fo OCR text detected in the image. Baseline results are obtained from Lu et al. (2023). Human performance and the results surpassing human performance are highlighted in grey. Our results are highlighted in blue.

robotics, and creating 3D experiences for augmented/virtual reality [31, 44]. Typical aspects of real-world scenes such as uniformly colored areas or non-Lambertian surfaces remain challenging.

As a traditional line of research, multi-view stereo methods [13, 33, 38, 42] usually estimate depth maps with photometric consistency and then reconstruct the surface as a post-processing step, *e.g.* point cloud fusion with screened Poisson surface reconstruction [15] or TSDF fusion [9]. However, they are unable to reconstruct reflective surfaces since their appearances are not multi-view consistent.

Recently, Neural Radiance Fields (NeRF) [22] render compelling photo-realistic images by parameterizing a scene as a continuous function of radiance and volume density using a multi-layer perceptron (MLP). More recent works [4, 7, 24, 35] replace or augment MLPs with grid based data structures to accelerate training. For example, Instant-NGP (iNGP) [24] uses a pyramid of grids and hashes to encode features and a tiny MLP to process them. Motivated by NeRF, neural implicit reconstruction methods [40, 43] combine signed distance functions (SDF) with volume rendering, and produce smooth and complete surfaces. For acceleration, recent works [17, 32] rely on hash grid representations and reconstruct surfaces with finer details. However, these NeRF-based methods cannot accurately reconstruct reflective surfaces.

To better capture the appearance of reflective surfaces, Ref-NeRF [37] computes the color with separate diffuse and specular components and parameterizes the appearance using reflected view that exploits the surface normals. BakedSDF [44] adopts the same Ref-NeRF components with a VolSDF [43] backbone to reconstruct large-scale scenes with shiny surfaces. However, BakedSDF is slow to train and struggles with reconstructing fine details.

We observe that while reflected view radiance fields can effectively reconstruct highly specular reflections, they struggle to represent more diffuse or ambiguous reflection types that can be found in real scenes. In contrast, we find that direct camera view radiance fields are more robust to difficult surfaces in real settings, although the reconstructions still present artifacts for reflective scenes. In this paper, we seamlessly bring together geometry based reflected view radiance fields and camera view-based radiance fields into a novel unified radiance field for representing 3D real scenes accurately in the presence of reflections. Our method is robust for reconstructing both real challenging scenes and highly reflective surfaces.

The proposed method, named UniSDF, achieves state-of-the-art performance on DTU [1], Shiny Blender [37], Mip-NeRF 360 dataset [3] and Ref-NeRF real dataset [37]. It demonstrates the capability to accurately reconstruct complex scenes with large scale, fine details and reflective surfaces as we see in Fig. 1. Our contributions are summarized as follows:

- We propose a novel algorithm that learns to seamlessly combine two radiance fields in 3D while exploiting the advantages or each representation. Our method produces high quality object surfaces in both reflective and non-reflective regions.
- Ours method relies on a hash grid backbone that enables fast training while maintaining high reconstruction quality. Moreover, our pipeline is robust and does not require large amounts of parameter tuning.

2. Related Works

Multi-view stereo (MVS). Many traditional [33, 41] and learning-based [13, 38, 39, 42] MVS methods first estimate multi-view depth maps and then reconstruct the surface by fusing depth maps in a post-processing step. As the core step, depth estimation is mainly based on the photometric consistency assumption across multiple views. However, this assumption fails for glossy surfaces with reflections, and thus MVS methods cannot reconstruct them accurately.

Neural radiance fields (NeRF). As a seminal method in view synthesis, NeRF [22] represents a scene as a continuous volumetric field with an MLP, with position and camera view direction as inputs, and renders an image using volumetric ray-tracing. Since NeRF is slow to train, some methods [7, 24, 35] use voxel-grid-like data structures to accelerate training. Many follow-up works apply NeRFs to different tasks, *e.g.* sparse-view synthesis [26, 36, 46], real-time rendering [8, 14, 29, 45], 3D generation [6, 20, 28] and pose estimation [19, 34, 49]. For the 3D reconstruction task, there are many methods [11, 17, 21, 27, 30, 32, 40, 43, 47] integrating NeRF with signed distance functions, a common implicit function for geometry. Specifically, they transform SDFs back to volume density for volume rendering. However, we observe that they are unable to reconstruct shiny / reflective surfaces since NeRF's camera view direction parameterization for the color prediction does not accurately model reflective parts of the scene.

NeRFs for reflections. Recently, Ref-NeRF [37] reparameterizes the appearance prediction with separate diffuse and reflective components by using the reflected view direction, which improves the rendering of specular surfaces. As a result, recent works such as BakedSDF [44] or ENVIDR [18] adopt this representation to reconstruct glossy surfaces of unbounded scenes and with material decomposition, respectively. While leading to strong view-synthesis results for reflective areas, we find that reflective radiance field approaches often lead to overly smooth reconstructions with missing details and that their optimization is not stable on real-world scenes. In contrast to existing methods with a