

Deep Generative Priors for View Synthesis at Scale

*Hang Gao
Angjoo Kanazawa
Jitendra Malik
Alexei (Alyosha) Efros
Shubham Tulsiani*

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2025-170

<http://www2.eecs.berkeley.edu/Pubs/TechRpts/2025/EECS-2025-170.html>

August 19, 2025



Copyright © 2025, by the author(s).
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

Deep Generative Priors for View Synthesis at Scale

by

Hang Gao

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Assistant Professor Angjoo Kanazawa, Chair

Professor Jitendra Malik

Professor Alexei A. Efros

Assistant Professor Shubham Tulsiani

Summer 2025

Deep Generative Priors for View Synthesis at Scale

Copyright 2025
by
Hang Gao

Abstract

Deep Generative Priors for View Synthesis at Scale

by

Hang Gao

Doctor of Philosophy in Computer Science

University of California, Berkeley

Assistant Professor Angjoo Kanazawa, Chair

View synthesis—the task of generating photorealistic images of a scene from *novel* camera viewpoints—is a cornerstone of computer vision, underpinning graphics, immersive reality, and embodied AI. Yet despite its importance, view synthesis has not demonstrated scaling properties comparable to those in language or 2D generation, even when provided with more data and compute: reconstruction-based methods collapse under sparse views or scene motion, while generative models struggle with 3D consistency and precise camera control.

This thesis shows that deep generative priors—instantiated as diffusion models conditioned on camera poses—bridge this gap. We proceed in three steps. First, we start by revealing that state-of-the-art dynamic view-synthesis benchmarks quietly rely on multi-view cues; removing those cues triggers steep performance drops and exposes the brittleness of reconstruction-based models. Then, we present a working solution that injects learned monocular depth and long-range tracking priors into a dynamic 3D Gaussian scene representation, recovering globally consistent geometry and motion from a single video. Finally, we abandon explicit reconstruction altogether, coupling camera-conditioned diffusion with a two-pass sampling strategy to synthesize minute-long, camera-controlled videos from as little as one input image.

From diagnosing the limits of reconstruction, to augmenting it with data-driven regularizers, to replacing it with a fully generative pipeline, our results trace a clear progression that delivers state-of-the-art fidelity, temporal coherence, and camera control precision while requiring orders-of-magnitude less input signal. We conclude by outlining open challenges and future directions for scaling view synthesis to truly world-scale 3D environments.

To my loves, who I become.

Contents

Contents	ii
List of Figures	iv
List of Tables	vi
1 Introduction	1
1.1 View synthesis in the age of scaling	1
1.2 Thesis overview	2
2 Monocular dynamic view synthesis: A reality check	3
2.1 Introduction	3
2.2 Related work	4
2.3 Effective multi-view in a monocular video	5
2.4 Towards better experimentation practice	7
2.5 Reality check: Re-evaluating the state of the art	12
2.6 Discussion and recommendation for future works	17
3 Shape of Motion: 4D reconstruction from a single video	18
3.1 Introduction	18
3.2 Related work	19
3.3 Method	20
3.4 Experiments	25
3.5 Discussion and conclusion	32
4 Stable Virtual Camera: Generative view synthesis with diffusion models	33
4.1 Introduction	33
4.2 Background	35
4.3 Method	38
4.4 Experiments	42
4.5 Conclusion	51
5 Conclusion	54

5.1	Limitations and future work	54
5.2	The future of 3D	55
Bibliography		56
A	Monocular dynamic view synthesis: A reality check	71
A.1	Outline	71
A.2	Computation for effective multi-view factors (EMFs)	71
A.3	Computation for co-visibility mask and masked image metrics	73
A.4	Correspondence readout from existing works	76
A.5	Summary of the capture setup and data processing for our iPhone dataset	78
A.6	Summary of the implementation details and remaining differences	80
A.7	Additional results on the impact of effective multi-view	82
A.8	Additional results on per-sequence quantitative performance breakdown	82
A.9	Additional results on novel-view synthesis	82
A.10	Additional results on inferred correspondence	82
B	Shape of Motion: 4D reconstruction from a single video	95
B.1	Outline	95
B.2	Additional preprocessing details	95
B.3	Additional training details	96
B.4	Additional evaluation details	98
B.5	Visualization of Kubric experiment	98
B.6	NVIDIA dataset evaluation	98
C	Stable Virtual Camera: Generative view synthesis with diffusion models	99
C.1	Outline	99
C.2	Broader impact and limitations	99
C.3	Related work	100
C.4	Benchmark	101
C.5	Additional experiments	106

List of Figures

2.1	Visualizing the training data of existing benchmarks.	5
2.2	The spectrum of effective multi-view in a monocular video.	6
2.3	Statistics of effective multi-view factors (EMFs) across different datasets.	8
2.4	Camera teleportation from a multi-camera rig.	9
2.5	Evaluation without teleportation with the co-visibility mask.	10
2.6	High-quality novel-view synthesis does not imply accurate correspondence modeling.	11
2.7	Capture setup and sampled sequences from our proposed iPhone dataset.	11
2.8	Impact of effective multi-view on the Nerfies-HyperNeRF dataset.	13
2.9	Qualitative results on the Nerfies-HyperNeRF dataset without camera teleportation.	14
2.10	Ablation study on improving the state of the art on the proposed iPhone dataset.	15
2.11	Qualitative results on the proposed iPhone dataset.	16
3.1	Shape of Motion.	21
3.2	Shape of Motion system overview.	22
3.3	3D tracking visualization on iPhone dataset.	27
3.4	Qualitative comparison of novel-view synthesis on iPhone dataset.	28
3.5	Visualization of motion coefficients after PCA and predicted depth maps on iPhone dataset.	29
3.6	First three PCA components of the optimized motion coefficients.	31
4.1	Generative view synthesis.	34
4.2	Diverse camera control.	36
4.3	Set NVS <i>versus</i> trajectory NVS.	38
4.4	Stable Virtual Camera method overview.	39
4.5	Anchor generation for long video generation.	42
4.6	SOTA comparison on set NVS and trajectory NVS across varying numbers of input views.	45
4.7	Temporal quality.	48
4.8	Long-range 3D consistency.	49
4.9	Generation quality on the number of input views.	50
4.10	Generation quality on different image resolutions.	52
4.11	Generation uncertainty on CFG.	53
4.12	Generation diversity in unseen regions.	53

A.1	Illustration of the computation process for co-visibility.	75
A.2	Visualizations of the multi-camera captures after time synchronization from the proposed iPhone dataset.	84
A.3	Visualizations of the depth filtering during data preprocessing of the proposed iPhone dataset.	85
A.4	Visualizations of the keypoint annotation during data preprocessing of the proposed iPhone dataset.	86
A.5	Additional qualitative results on the impact of effective multi-view on the Nerfies-HyperNeRF dataset.	87
A.6	Additional qualitative results on the Nerfies-HyperNeRF dataset without camera teleportation.	89
A.7	Additional qualitative results on the multi-camera captures from the proposed iPhone dataset.	90
A.8	Additional qualitative results on the single-camera captures from the proposed iPhone dataset.	91
A.9	Additional qualitative results on the full image rendering on both the Nerfies-HyperNeRF dataset and the proposed iPhone dataset.	92
A.10	Additional qualitative results of keypoint transferring on the Nerfies-HyperNeRF dataset without camera teleportation.	93
A.11	Additional qualitative results of keypoint transferring on the proposed iPhone dataset.	94
C.1	3DGS versus samples.	106
C.2	Padding.	107
C.3	Diverse camera motions and effects (object-centric).	108
C.4	Diverse camera motions and effects (text-prompted scene).	109
C.5	Diverse camera motions and effects (real-life object-centric).	110
C.6	Diverse camera motions and effects (real-life scene).	111

List of Tables

2.1	Summary of the existing and proposed iPhone datasets.	8
2.2	Benchmark results on the rectified Nerfies-HyperNeRF dataset.	15
2.3	Benchmark results on the proposed iPhone dataset.	15
3.1	Evaluation on iPhone dataset.	26
3.2	3D tracking evaluation on Kubric dataset.	30
3.3	Ablation studies on iPhone dataset.	31
4.1	Comparison of existing NVS models based on the source of training data and key attributes.	37
4.2	PSNR \uparrow on small-viewpoint set NVS.	43
4.3	PSNR \uparrow on large-viewpoint set NVS.	44
4.4	PSNR \uparrow on 3DGS renderings for set NVS.	44
4.5	PSNR \uparrow on trajectory NVS.	47
4.6	3D consistency (TSED \downarrow and PSNR \uparrow) and temporal quality (MS \uparrow) on trajectory NVS.	47
A.1	Per-sequence breakdowns of the statistics of different datasets.	74
A.2	Our re-implementation reproduces Nerfies’s and HyperNeRF’s results.	81
A.3	Reproduced results of NSFF on the dynamic sequence.	81
A.4	Per-scene breakdowns of the quantitative results on the Nerfies-HyperNeRF dataset.	83
A.5	Per-scene breakdowns of the quantitative results on the proposed iPhone dataset.	88
A.6	Additional quantitative results of the PCK-T evaluation on the single-camera captures from the proposed iPhone dataset.	88
C.1	Statistics for NVS benchmark.	102
C.2	LPIPS \downarrow (top) and SSIM \uparrow (bottom) on small-viewpoint set NVS.	103
C.3	LPIPS \downarrow (top) and SSIM \uparrow (bottom) on large-viewpoint set NVS.	104
C.4	LPIPS \downarrow (top) and SSIM \uparrow (bottom) on 3DGS renderings for set NVS.	104
C.5	LPIPS \downarrow (top) and SSIM \uparrow (bottom) on trajectory NVS.	105

Acknowledgments

There are two kinds of fun in hiking: the kind you do with friends, and the kind you do alone. Over the years, I've come to appreciate both—and the same holds true for graduate school. I've been lucky to walk alongside kind souls who support me, and just as lucky to have stretches of solitude for exploring, questioning, and molding myself. I leave this chapter feeling less vague about who I am, less unclear about who I ought to become, and less afraid of my strength to bring it into reality.

To my advisor, Angjoo Kanazawa, for your good heart. In a time of uncertainty and change, your enthusiasm and belief in things have been a haven along my trek. I've learned, sometimes the hard way, the value of strong opinions, loosely held, in the years we've worked together. I will miss you, and I don't expect to meet another person whose eyes light up the way yours do when talking about ideas. Now you've become a mother, and watching you nurture a family with the same care and spirit you've brought to our lab brings me great joy. I can't wait to see little Ayuna grow up in the warmth you've created.

To the members of my committee—Jitendra Malik, Alyosha Efros, and Shubham Tulsiani—thank you for kindly agreeing to let me put your names on my thesis. To Jitendra, for your inexhaustible knowledge and sweeping context. To Alyosha, for your blend of absolute playfulness and seriousness. To Shubham, for your level-headedness and sharp wit. To my intern hosts over the summers—Varun Japami, Bryan Russell, and Alex Yu—for showing me the outside world. To my outside collaborators—Jensen Zhou, Zhuoyang Pan, and Zhengqi Li—for the grind we shared.

To my best friends in graduate school: in our relationships, I learned that people with completely different skins can still connect on a deep level. I found fragments of my soul in you, and I'm lucky to have met you. To Ruilong Li, for being consistently reliable and my go-to for almost anything, and for showing me a different way of living. To Lucas Xie, for your spontaneity and edge, for walks through beaches, mountains, prairies, and geysers, for being true to yourself—and sometimes to me as well. In some way, you two define the spectrum of my personality, and because of you, I've been knowing who I am.

To my Berkeley cohort—Qianqian Wang, for being like a hound in water; Vickie Ye, for your reckoning act; Junchen Liu, for meme-posting; Brent Yi, for being a wildcard; Chung Min Kim, for your Jellycat; Songwei Ge, for being a great cat parent; Ethan Weber, for being the sweetest person I've known; David McAllister, for showing me your FUJIFILM X100VI; Aleksander Holynski, for being elusively kind; Georgios Palakos, for the Ouzo; Alex Yu, for your focus and wandering (mostly focus); Matthew Tancik, for your intelligence and characters; Boyi Li, for the sweets; Toru Lin, for trying; Tyler Bonnen, for your travel tips; Vongani Maluleke, for being my fellow GSI; Yifei Zhang, for reciting poems; Shubham Goel, for tricking me into the water; Karttikeya Mangalam, for the COVID walks; Ilija Radosavovic, for crossing our path; Wenlong Huang, for letting me use your room; Zhe Cao, for the lessons. To Justin Kerr, Haven Feng, Junyi Zhang, Riley Peterlinz, Lea Muller, Frederik Warburg, Lisa Dunlap, Tim Brooks, Bill Peebles, Jathushan Rajasegaran, Jeffrey Zhang. I wish you happiness and much of the greater things to come.

To those who broadened my world and pushed me to dabble: Yunqi Wei, for fiat lux; Nicole Xu, for your old soul; Xuyang Liu, for being my home on this continent; Chengfeng Luo, for challenging my values; Chang He, for climbing trees with me; Qian Huang, for being cool; Xiang

Li, for the good talks in the desert; another Xiang Li, for that naked run in the rain so we could get to the sauna; Yifan Gu, for breaking free. To Chengjing Li and Shiqi Ke, for our friendship for decades.

To more friends, Nanqing, Jiarui, Yini, Hao, Nanyi, Yongfei, Peihang, Yutong, Yi, Parvin, Songyou, Shangzhe, Gengshan, and Fangchen, for the good times.

To my fellow travelers—Lucia Berlin, Italo Calvino, Wislawa Szymborska, for your lightness and grace; Hermann Hesse, El Greco, William Blake, Fyodor Dostoevsky, for your faith; Ryuichi Sakamoto, Irvin Yalom, for reflection on my own demise; Francisco Goya, Henri Rousseau, Rene Magritte, Mario Vargas Llosa, Milan Kundera, Elena Ferrante, Friedrich Nietzsche, for lending me your eyes.

Lastly, to my family—for both struggle and growth. To my mother, for showing me how to endure; my father, for a study in intricate passion; my aunt, for protecting my memory and showing me how to love. To Sriracha, for coming into and out of my life, for your love, and for that night I will never forget. It is with profound fear, honor, and appreciation that I came into the realization of bearing your souls in mine.

Chapter 1

Introduction

We often turn to photographs of the places we love—not just to remember what they looked like, but to imagine what’s beyond the frame: a childhood street glowing under autumn light, a lone bench blanketed by phoenix trees in Central Park, or a moment during a live event when the rhythm, the crowd, and the mood fall into sync. Imagine being able to return to those moments and look around freely, as if they were sealed inside a time capsule.

This is the promise of view synthesis: the ability to generate new perspectives of a scene from existing images. It is a core problem in computer vision and has the potential to reshape how we interact with the visual world. Despite its importance, view synthesis has not exhibited the scaling behavior observed in language and 2D generation, even when provided with more data and compute. Traditional reconstruction-based pipelines fail under sparse views or dynamic scenes, while generative models struggle with 3D consistency and camera control. Bridging this gap calls for new methods that combine geometric reasoning with strong generative priors.

1.1 View synthesis in the age of scaling

Scaling is at the heart of recent progress in AI. From language models to image and video generation, performance improves reliably with bigger models, larger datasets, and more compute. These models—typically large transformers—are trained end-to-end via gradient descent on massive corpora, revealing emergent capabilities as scale increases.

This simple recipe, however, has yet to translate to 3D view synthesis. Unlike language and 2D vision, 3D generation must adhere to geometric constraints: each novel view must respect the scene’s structure, appearance, and dynamics. In addition, model architectures for view synthesis are not yet settled. While transformers dominate language and vision, it’s still unclear how to natively perform neural rendering within a transformer-style framework. On the data side, the gap is even wider. The number of 3D scenes available is orders of magnitude smaller than 2D images, let alone the text corpora used for training LLMs. Even when such data is available, it remains unclear what annotations or representations best support generalizable 3D reasoning. Finally, the ecosystem around view synthesis has grown up in a different scaling regime. Neural rendering

pipelines typically succeed with well-captured image sequences and modest training costs. As a result, there’s been little practical or industrial pressure to scale up data or compute, especially in a climate dominated by LLM and video model development.

To tackle these challenges, this thesis adopts three guiding ideas. First, we recast view synthesis as conditional image generation, letting 3D coherence emerge from data instead of explicit rendering. Second, we strip away heavy geometric bias, treating the camera only as a conditioning token so large 2D diffusion backbones can be reused without modification. Third, we tap the latent 3D priors in pre-trained image and video models and refine them with a modest multiview corpus to gain precise camera control. With this recipe, view synthesis inherits the familiar scaling trio of modern generative AI—data diversity, model capacity, and compute.

1.2 Thesis overview

This thesis follows the progression in our investigation: from identifying the core limitations of existing methods, to injecting data-driven priors into classical pipelines, to scaling view synthesis through fully generative modeling.

In Chapter 2, we begin with a diagnostic analysis of reconstruction-based methods. While approaches like NeRF and its dynamic variants perform well under idealized settings, our study reveals that they implicitly rely on multi-view cues often unavailable in monocular video. When these cues are removed, performance drops sharply, exposing the brittleness of purely geometric pipelines. We introduce a new benchmark that includes more diverse real-life motion along with a new experimental protocol that explicitly disentangles multi-view cues from the input data.

In Chapter 3, we next explore hybrid methods that augment traditional reconstruction with learned priors. By integrating monocular depth and long-range 2D point tracks—obtained from off-the-shelf predictors—into a unified dynamic 3D Gaussian representation, we enable the system to reason over the entire video sequence. Each point’s motion is expressed as a combination of shared SE(3) bases, allowing the scene to be softly decomposed into rigidly moving parts. This formulation yields a globally consistent reconstruction of both geometry and motion, even from sparse, casually captured monocular input.

Finally, in Chapter 4, we abandon explicit reconstruction altogether and reframe view synthesis as conditional image generation. In this formulation, the renderer is simply a diffusion model conditioned on camera poses, without any intermediate 3D representation. We show that this setup enables strong 3D consistency to emerge directly from data, rather than being enforced by geometric supervision. By fine-tuning pre-trained image and video diffusion models on multiview datasets, we teach them to synthesize photorealistic, camera-controllable novel views and long, temporally coherent video sequences. This approach inherits the scalability and visual fidelity of large 2D generative models, while achieving strong generalization to in-the-wild scenes—thus unlocking a practical and scalable regime for view synthesis.

Finally, in Chapter 5, we summarize our findings and outline open challenges for future research. Together, these chapters show that 3D vision can scale, just as other domains, when we learn useful priors from data with minimal 3D inductive bias.

Chapter 2

Monocular dynamic view synthesis: A reality check

We study the recent progress on dynamic view synthesis (DVS) from monocular video. Though existing approaches have demonstrated impressive results, we show a discrepancy between the practical capture process and the existing experimental protocols, which effectively leaks in multi-view signals during training. We define *effective multi-view factors* (EMFs) to quantify the amount of multi-view signal present in the input capture sequence based on the relative camera-scene motion. We introduce two new metrics: co-visibility masked image metrics and correspondence accuracy, which overcome the issue in existing protocols. We also propose a new iPhone dataset that includes more diverse real-life deformation sequences. Using our proposed experimental protocol, we show that the state-of-the-art approaches observe a 1-2 dB drop in masked PSNR in the absence of multi-view cues and 4-5 dB drop when modeling complex motion. Code and data can be found at <https://hangg7.com/dycheck>.

2.1 Introduction

Dynamic scenes are ubiquitous in our everyday lives – people moving around, cats purring, and trees swaying in the wind. The ability to capture 3D dynamic sequences in a “casual” manner, particularly through monocular videos taken by a smartphone in an uncontrolled environment, will be a cornerstone in scaling up 3D content creation, performance capture, and augmented reality.

Recent works have shown promising results in dynamic view synthesis (DVS) from a monocular video [1, 2, 3, 4, 5, 6, 7, 8]. However, upon close inspection, we found that there is a discrepancy between the problem statement and the experimental protocol employed. As illustrated in Figure 2.1, the input data to these algorithms either contain frames that “teleport” between multiple camera viewpoints at consecutive time steps, which is impractical to capture from a single camera, or depict quasi-static scenes, which do not represent real-life dynamics. We provide a systematic means of characterizing the aforementioned discrepancy and propose a better set of practices for model fitting and evaluation. Concretely, we introduce *effective multi-view factors* (EMFs) to quantify the amount

of multi-view signal in a monocular sequence based on the relative camera-scene motion. With EMFs, we show that the current experimental protocols operate under an effectively multi-view regime. For example, our analysis reveals that the aforementioned practice of camera teleportation makes the existing capture setup akin to an Olympic runner taking a video of a moving scene without introducing any motion blur.

The reason behind the existing experimental protocol is that monocular DVS is a challenging problem that is also hard to evaluate. Unlike static novel-view synthesis where one may simply evaluate on held-out views of the captured scene, in the dynamic case, since the scene changes over time, evaluation requires another camera that observes the scene from a different viewpoint at the same time. However, this means that the test views often contain regions that were never observed in the input sequence. Camera teleportation, i.e., constructing a temporal sequence by alternating samples from different cameras, addresses this issue at the expense of introducing multi-view cues, which are unavailable in the practical single-camera capture.

We propose two sets of metrics to overcome this challenge without the use of camera teleportation. The first metric enables evaluating only on pixels that were seen in the input sequence by computing the co-visibility of every test pixel. The proposed co-visibility mask can be used to compute masked image metrics (PSNR, SSIM [9] and LPIPS [10]). While the masked image metrics measure the quality of rendering, they do not directly measure the quality of the inferred scene deformation. Thus, we also propose a second metric that evaluates the quality of established point correspondences by the percentage of correctly transferred keypoints (PCK-T) [11]. The correspondences may be evaluated between the input and test frames or even within the input frames, which enable evaluation on sequences that are captured with only a single camera.

We conduct extensive evaluation on existing datasets [5, 7] as well as a new dataset that includes more challenging motion and diverse scenes. When tested on existing datasets without camera teleportation, the state-of-the-art methods observe a 1-2 dB drop in masked PSNR and ~5% drop in PCK-T. When tested on complex motion with the proposed dataset, existing approaches observe another 4-5 dB drop in masked PSNR and ~30% drop in PCK-T, suggesting a large room for improvement. We encourage future works to report EMFs on new data and adopt our experimental protocol to evaluate monocular DVS methods. Code and data are available at our [project page](#).

2.2 Related work

Non-rigid structure from motion (NR-SfM). Traditional NR-SfM tackles the task of dynamic 3D inference by fitting parametric 3D morphable models [12, 13, 14, 15, 16, 17, 18, 19], or fusing non-parametric depth scans of generic dynamic scenes [20, 21, 22, 23, 24]. All of these approaches aim to recover accurate surface geometry at each time step and their performance is measured with ground truth 3D geometry or 2D correspondences with PCK [25] when such ground truth is not available. We analyze recent dynamic view synthesis methods whose goal is to generate a photo-realistic novel view. Due to their goal, these methods do not focus on evaluation against ground truth 3D geometry, but we take inspiration from prior NR-SfM works to evaluate the quality

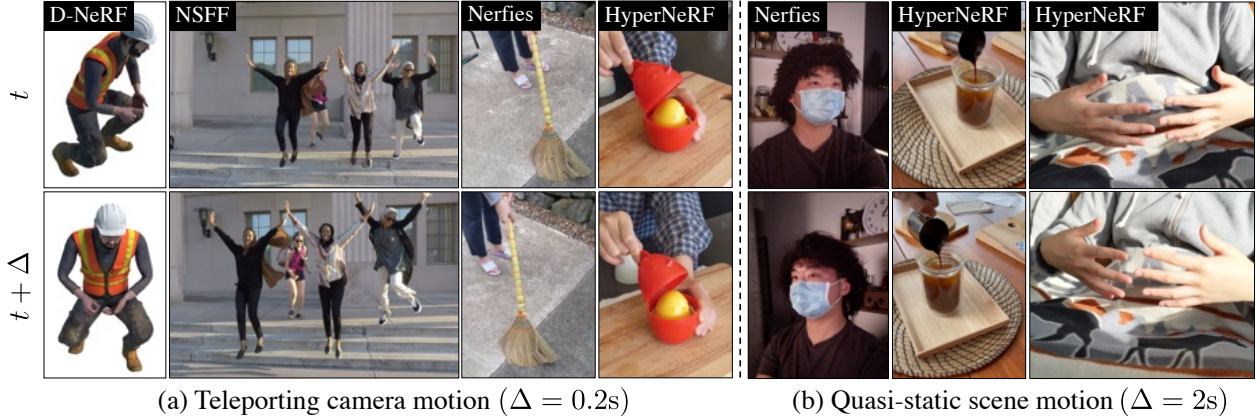


Figure 2.1: **Visualizing the training data of existing benchmarks.** Existing datasets operate under the effective multi-view regime: The sequences either have (a) teleporting camera motion or (b) quasi-static scene motion. These motions leak multi-view cues, e.g., the model can observe the human (1st column) and hands (last column) at roughly the same pose from different viewpoints.

of the inferred 3D dynamic representation based on correspondences. We also draw inspiration from previous NR-SfM work that analyzed camera/object speed and 3D reconstruction quality [26, 27].

Monocular dynamic neural radiance fields (dynamic NeRFs). Dynamic NeRFs reconstruct moving scenes from multi-view inputs or given pre-defined deformation template [28, 29, 30, 31, 32, 33, 34, 35]. In contrast, there is a series of recent works that seek to synthesize high-quality novel views of generic dynamic scenes given a monocular video [1, 2, 3, 4, 5, 6, 7, 8]. These works can be classified into two categories: a deformed scene is directly modeled as a time-varying NeRF in the world space [1, 4, 6] or as a NeRF in canonical space with a time-dependent deformation [2, 3, 5, 7, 8]. The evaluation protocol in these works inherit from the original static-scene NeRF [36] that quantify the rendering quality of held-out viewpoints using image metrics, e.g., PSNR. However, in dynamic scenes, PSNR from an unseen camera view may not be meaningful since the novel view may include regions that were never seen in the training view (unless the method can infer unseen regions using learning based approaches). Existing approaches resolve this issue by incorporating views from multiple cameras during training, which we show results in an effectively multi-view setup. We introduce metrics to measure the difficulties of an input sequence, a monocular dataset with new evaluation protocol and metrics, which show that existing methods have a large room for improvement.

2.3 Effective multi-view in a monocular video

We consider the problem of dynamic view synthesis (DVS) from a monocular video. A monocular dynamic capture consists of a single camera observing a moving scene. The lack of simultaneous multi-view in the monocular video makes this problem more challenging compared to the multi-view setting, such as reconstructing moving people from multiple cameras [30, 34, 37].

Contrary to the conventional perception that the effect of multi-view is binary for a capture (single versus multiple cameras), we show that it can be characterized on a continuous spectrum. Our insight is that a monocular sequence contains *effective* multi-view cues when the camera moves much faster than the scene, though technically the underlying scene is observed only once at each time step.

2.3.1 Characterizing effective multi-view in a monocular video

Although a monocular video only sees the scene from one viewpoint at a time, depending on the capture method, it can still contain cues that are effectively similar to those captured by a multi-view camera rig, which we call as effective multi-view. As shown in Figure 2.2, when the scene moves significantly slower than the camera (to the far right end of the axis), the same scene is observed from multiple views, resulting in multi-view capture. In this case, DVS reduces to a well-constrained multi-view stereo problem at each time step. Consider another case where the camera moves significantly faster compared to the scene so that it observes roughly the same scene from different viewpoints. As the camera motion approaches the infinity this again reduces the monocular capture to a multi-view setup. We therefore propose to characterize the amount of multi-view cues by the relative camera-scene motion.

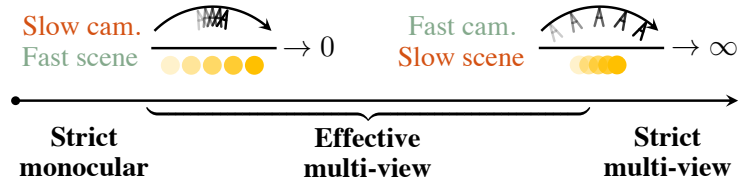


Figure 2.2: **The spectrum of effective multi-view in a monocular video.** A video captured by a single camera can still have multi-view cues when the camera moves much faster than the scene. We disentangle recent advances in DVS given a monocular video from such phenomena.

2.3.2 Quantifying effective multi-view in a monocular video

For practicality, we propose two metrics, referred to as effective multi-view factors (EMFs). The first metric, full EMF Ω is defined as the relative ratio between the motion magnitude of the camera to the scene, which in theory characterizes the effective multi-view perfectly, but in practice can be expensive and challenging to compute. The second metric, angular EMF ω is defined as the camera angular velocity around the scene look-at point, which only considers the camera motion; while approximate, it is easy to compute and characterizes object-centric captures well.

Full EMF Ω : ratio of camera-scene motion magnitude. Consider a monocular video of a moving scene over a set of time steps \mathcal{T} . At each discrete time $t \in \mathcal{T}$, let the camera’s 3D location be \mathbf{o}_t . We consider each point \mathbf{x}_t on the domain of observed scene surface $\mathbb{S}_t^2 \subset \mathbb{R}^3$. We define the

camera-scene motion as the expected relative ratio,

$$\Omega = \mathbb{E}_{t, t+1 \in \mathcal{T}} \left[\mathbb{E}_{\mathbf{x}_t \in \mathbb{S}_t^2} \left[\frac{\|\mathbf{o}_{t+1} - \mathbf{o}_t\|}{\|\mathbf{x}_{t+1} - \mathbf{x}_t\|} \right] \right], \quad (2.1)$$

where the denominator $\mathbf{x}_{t+1} - \mathbf{x}_t$ denotes the 3D scene flow and the numerator $\mathbf{o}_{t+1} - \mathbf{o}_t$ denotes the 3D camera motion, both over one time step forward. The 3D scene flow can be estimated via the 2D dense optical flow field and the metric depth map when available, or monocular depth map from off-the-shelf approaches [38, 39] in the general case. Please see the Appendix for more details. Note that Ω in theory captures the effective multi-view factor for any sequence. However, in practice, 3D scene flow estimation is an actively studied problem and may suffer from noisy or costly predictions.

Angular EMF ω : camera angular velocity. We introduce a second metric ω that is easy to compute in practice. We make an additional assumption that the capture has a single look-at point in world space, which often holds true, particularly for captures involving a single centered subject. Specifically, given a look-at point \mathbf{a} by triangulating the optical-axes of all cameras (as per [5]) and the frame rate N , the camera angular velocity ω is computed as a scaled expectation,

$$\omega = \mathbb{E}_{t, t+1 \in \mathcal{T}} \left[\arccos \left(\frac{\langle \mathbf{a} - \mathbf{o}_t, \mathbf{a} - \mathbf{o}_{t+1} \rangle}{\|\mathbf{a} - \mathbf{o}_t\| \cdot \|\mathbf{a} - \mathbf{o}_{t+1}\|} \right) \right] \cdot N. \quad (2.2)$$

Note that even though ω only considers the camera motion, it is indicative of effective multi-view in the majority of existing captures, which we describe in Section 2.4.1.

For both Ω and ω , the larger the value, the more multi-view cue the sequence contains. For future works introducing new input sequences, we recommend always reporting angular EMF for its simplicity and reporting full EMF when possible. Next we inspect the existing experimentation practices under the lens of effective multi-view.

2.4 Towards better experimentation practice

In this section, we reflect on the existing datasets and find that they operate under the effective multi-view regime, with either teleporting camera motion or quasi-static scene motion. The reason behind the existing protocol is that monocular DVS is challenging from both the modeling and evaluation perspective. While the former challenge is well known, the latter is less studied, as we expand below. To overcome the existing challenge in the evaluation and enable future research to experiment with casually captured monocular video, we propose a better toolkit, including two new metrics and a new dataset of complex motion in everyday lives.

2.4.1 Closer look at existing datasets

We investigate the datasets used for evaluation in D-NeRF [3], HyperNeRF [7], Nerfies [5], and NSFF [4]. Table 2.1 shows their statistics. We evaluate the amount of effective multi-view cues

	#Train cam.	Duration	FPS	Depth	Kpt.	Sequences
D-NeRF [3]	~150	1 – 3s	60	-	-	8 MV
HyperNeRF [7]	2	8 – 15s	15	-	-	3 MV + 13 SV
Nerfies [5]	2	8 – 15s	5/15	-	-	4 MV
NSFF [4]	24	1 – 3s	15/30	Estimated [39]	-	8 MV
iPhone (proposed)	1	8 – 15s	30/60	Lidar	✓	7 MV + 7 SV

Table 2.1: **Summary of the existing and proposed iPhone datasets.** Existing datasets operate under the effective multi-view regime, teleporting between multiple cameras during training to generate a synthetic monocular video. Unlike previous protocols, the proposed iPhone dataset consists of dynamic sequences captured by a *single* smoothly moving camera. It also has accompanying depth maps for training supervision and labeled keypoints for evaluation. “MV” denotes multi-camera capture, and “SV” denotes single-camera capture.

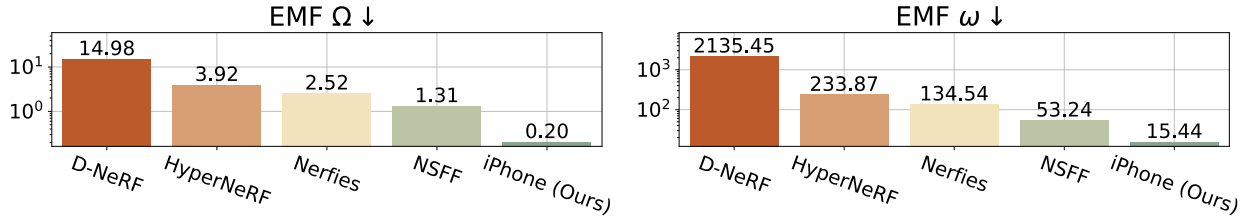


Figure 2.3: **Statistics of effective multi-view factors (EMFs) across different datasets.** Existing datasets have high EMFs, indicating the abundance of multi-view cues during training (note that the y-axis is in log scale). The proposed iPhone dataset features a single camera, capturing the moving scene with a smooth motion, and thus has smaller EMFs. Our results help ground the difficulty of each dataset for DVS from a monocular video.

via the proposed EMFs, shown in Figure 2.3. We find that existing datasets have large EMF values on both metrics. For example, the HyperNeRF dataset has an ω as large as $\sim 200^\circ/\text{s}$. To put these numbers in context, a person imaging an object 3m away has to move at 1m/s to get an $\omega = 20^\circ/\text{s}$ (close to the statistics in the proposed dataset). Some datasets exhibit ω higher than $120^\circ/\text{s}$, which is equivalent to a camera motion faster than the Olympic 100m sprint record, without incurring any motion blur.

Visualizing the actual training data shown in Figure 2.1 reveals that existing datasets feature non-practical captures of either (1) teleporting/fast camera motion or (2) quasi-static/slow scene motion. The former is not representative of practical captures from a hand-held camera, *e.g.*, a smartphone, while the latter is not representative of moving objects in daily life. Note that, out of the 23 multi-camera sequences that these prior works used for quantitative evaluation, 22 have teleporting camera motion, and 1 has quasi-static scene motion – the CURLS sequence shown at the 5th column in Figure 2.1. All 13 single-camera sequences from HyperNeRF [7] used for qualitative

evaluation have quasi-static scene motion. These systemic bases impaired the existing evaluation.

The four datasets also share a similar data protocol for generating effective multi-view input sequences from the original multi-camera rig capture. In Figure 2.4, we illustrate the teleporting protocol used in Nerfies [5] and HyperNeRF [7] as a canonical example. They sample alternating frames from two physical cameras (left and right in this case) mounted on a rig to create the training data. NSFF [4] samples alternating frames from 24 cameras based on the data released from Yoon *et al.* [40]. D-NeRF [3] experiments on synthetic dynamic scenes where cameras are randomly placed on a fixed hemisphere at every time step, in effect teleporting between 100-200 cameras. We encourage you to visit our [project page](#) to view the input videos from these datasets.

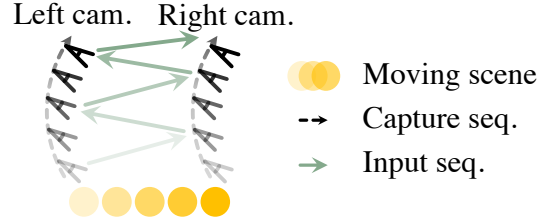


Figure 2.4: **Camera teleportation from a multi-camera rig.** As an example, we show the data protocol used in [5, 7].

Existing works adopt effective multi-view capture for two reasons. First, it makes monocular DVS more tractable. Second, it enables evaluating novel view on the full image, without worrying about the visibility of each test pixel, as all camera views were visible during training. We show this effect in Figure 2.5. When trained with camera teleportation (3rd column), the model can generate a high-quality full image from the test view. However, when trained without camera teleportation (4th column), the model struggles to hallucinate unseen pixels since NeRFs [5, 36] are not designed to predict completely unseen portions of the scene, unless they are specifically trained for generalization [41]. Next, we propose new metrics that enable evaluation without using camera teleportation. Note that when the model is trained without camera teleportation, the rendering quality also degrades, which we also evaluate.

2.4.2 Our proposed metrics

While the existing setup allows evaluating on the full rendered image from the test view, the performance under such evaluation protocol, particularly with teleportation, confounds the efficacy of the proposed approaches and the multi-view signal present in the input sequence. To evaluate with an actual monocular setup, we propose two new metrics that evaluate only on seen pixels and measure the correspondence accuracy of the predicted deformation.

Co-visibility masked image metrics. Existing works evaluate DVS models with image metrics on the *full* image, e.g., PSNR, SSIM [9] and LPIPS [10], following novel-view synthesis evaluation on static scenes [36]. However, in dynamic scenes, particularly for monocular capture with multi-camera validation, the test view contains regions that may not have been observed at all by the training camera. To circumvent this issue without resorting to camera teleportation, for each pixel in the test image, we propose *co-visibility* masking, which tests how many times a test pixel has been observed in the training images.

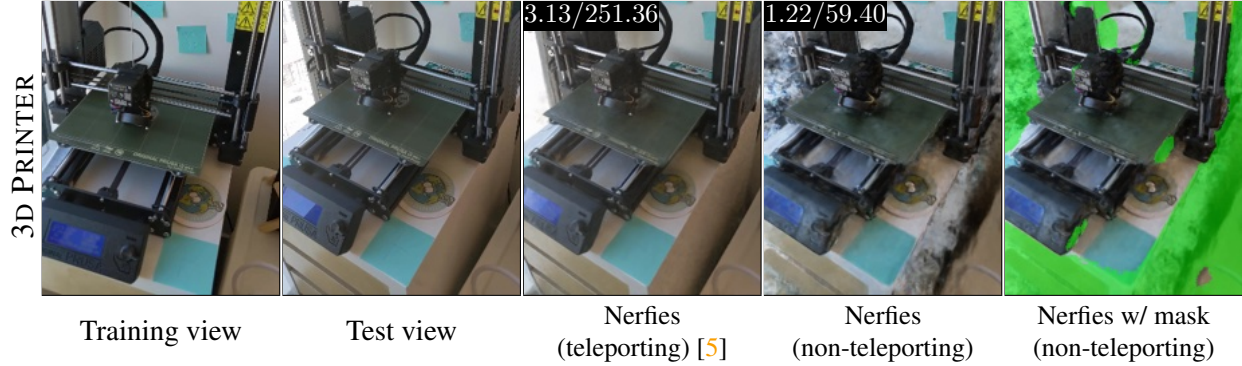


Figure 2.5: **Evaluation without teleportation with the co-visibility mask.** Ω/ω metrics of the input sequence are annotated on the top-left. Existing works avoid evaluating on unseen pixels by camera teleportation (3rd column). Naively training with the non-teleporting (smooth) camera trajectory causes evaluation issues on the full image (4th column), since a NeRF model cannot hallucinate unseen regions. We propose to only evaluate seen regions during training by a co-visibility mask (we show non-visible regions in green at the last column). Note that the model performs poorly on seen regions as well when trained without camera teleportation.

Specifically, we use optical flow to compute correspondences between every test image and the training images, and only keep test pixels that have enough correspondences in the training images via thresholding. This results in a mask, illustrated in Figure 2.5, which we use to confine the image metrics. We follow the common practice from the image generation literature and adopt masked metrics, mPSNR and mLPIPS [42, 43]. Note that NSFF [4] adopts similar metrics but for evaluating the rendering quality on foreground versus background regions. We additionally report mSSIM by partial convolution [44], which only considers seen regions during its computation. More details are in the Appendix. Using masked image metrics, we quantify the performance gap in rendering when a model is trained with or without multi-view cues in Section 2.5.1.

Percentage of correctly transferred keypoints (PCK-T). Correspondences lie at the heart of traditional non-rigid reconstruction [21], which is overlooked in the current image-based evaluation. We propose to evaluate 2D correspondences across training frames with the percentage of correctly transferred keypoints (PCK-T) [11], which directly evaluates the quality of the inferred deformation. Specifically, we sparsely annotate 2D keypoints across input frames to ensure that each keypoint is fully observed during training. For correspondence readout from existing methods, we use either root finding [45] or scene flow chaining. Please see the Appendix for details on our keypoint annotation, correspondence readout, and metric computation. As shown in Figure 2.6, evaluating correspondences reveal that high quality image rendering does not necessarily result in accurate correspondences, which indicates issues in the underlying surface, due to the ambiguous nature of the problem.

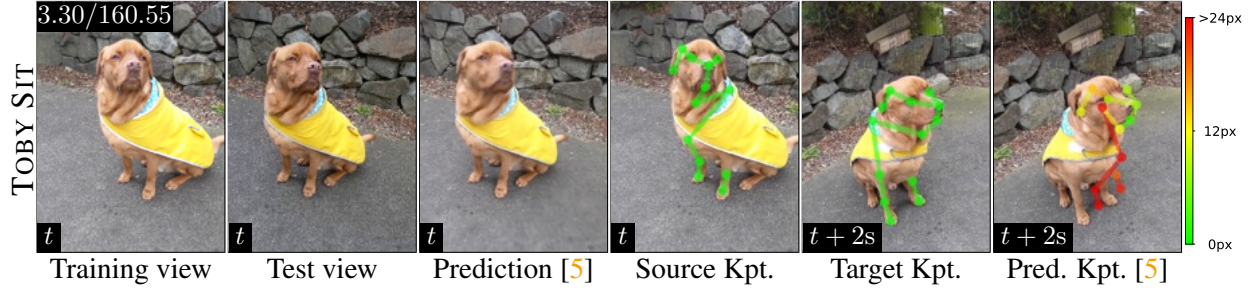


Figure 2.6: **High-quality novel-view synthesis does not imply accurate correspondence modeling.** Ω/ω metrics of the input sequence are shown on the top-left. The time steps of the ground-truth data and predictions are shown on the bottom-left. Using Nerfies [5] as an example, we show that the model renders high-quality results (3rd column) without modeling accurate correspondences (last column). Transferred keypoints are colorized by a heatmap of end-point error, overlaid on the ground-truth target frame.

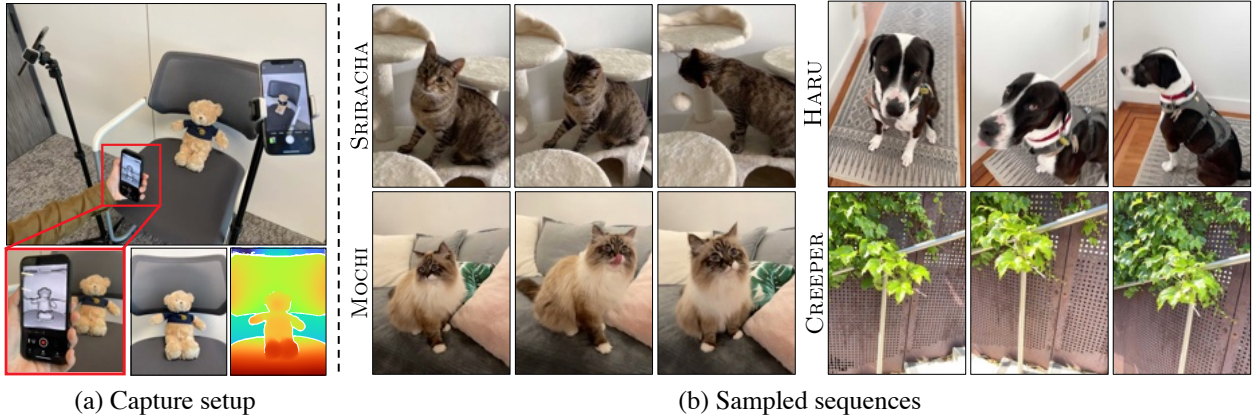


Figure 2.7: **Capture setup and sampled sequences from our proposed iPhone dataset.** The proposed iPhone dataset has a single hand-held camera for training and multiple cameras with a large baseline for validation. It has accompanying depth from the iPhone sensor and features diverse and complex real-life motions.

2.4.3 Proposed iPhone dataset

Existing datasets can be rectified by removing camera teleportation and evaluated using the proposed metrics, as we do in Section 2.5.1. However, even after removing camera teleportation, the existing datasets are still not representative of practical in-the-wild capture. First, the existing datasets are limited in motion diversity. Second, the evaluation baseline in existing datasets is small, which can hide issues in incorrect deformation and resulting geometry. For these reasons, we propose a new dataset called the *iPhone dataset* shown in Figure 2.7. In contrast to existing datasets with repetitive object motion, we collect 14 sequences featuring non-repetitive motion, from various categories

such as generic objects, humans, and pets. We deploy three cameras for multi-camera capture – one hand-held moving camera for training and two static cameras of large baseline for evaluation. Furthermore, our iPhone dataset comes with metric depth from the lidar sensors, which we use to provide ground-truth depth for supervision. In Section 2.5.2, we show that depth supervision, together with other regularizations, is beneficial for training DVS models. Please see the Appendix for details on our multi-camera capture setup, data processing, and more visualizations.

2.5 Reality check: Re-evaluating the state of the art

In this section, we conduct a series of empirical studies to disentangle the recent progress in dynamic view synthesis (DVS) given a monocular video from effective multi-view in the training data. We evaluate current state-of-the-art methods when the effective multi-view factor (EMF) is low.

Existing approaches and baselines. We consider the following state-of-the-art approaches for our empirical studies: NSFF [4], Nerfies [5] and HyperNeRF [7]. We choose them as canonical examples for other approaches [1, 2, 3, 6, 8, 34, 35], discussed in Section 2.2. We also evaluate time-conditioned NeRF (T-NeRF) as a common baseline [1, 3, 4]. Unlike the state-of-the-art methods, it is not possible to extract correspondences from a T-NeRF. A summary of these methods can be found in the Appendix.

Datasets. We evaluate on the existing datasets as well as the proposed dataset. For existing datasets, we use the multi-camera captures accompanying Nerfies [5] and HyperNeRF [7] for evaluation. Due to their similar capture protocol, we consider them as a single dataset in our experiment (denoted as the Nerfies-HyperNeRF dataset). It consists of 7 sequences in total, which we augment with keypoint annotations. Our dataset has 7 multi-camera captures and 7 single-camera captures. We evaluate novel-view synthesis on the multi-camera captures and correspondence on all captures. Our data adopts the data format from the Nerfies-HyperNeRF dataset, with additional support for depth and correspondence labels. All videos are at 480p resolution and all dynamic scenes are inward-facing.

Masked image and correspondence metrics. Following Section 2.4.2, we evaluate co-visibility masked image metrics and the correspondence metric. We report masked image metrics: mPSNR, mSSIM [9, 44], and mLPIPS [4, 10, 42, 43]. We visualize the rendering results with the co-visibility mask. For the correspondence metric, we report the percentage of correctly transferred keypoints (PCK-T) [11] with threshold ratio $\alpha = 0.05$. Additional visualizations of full image rendering and inferred correspondences can be found in the Appendix.

Implementation details. We consolidate Nerfies [5] and HyperNeRF [7] in one codebase using JAX [46]. Compared to the original official code releases, our implementation aligns all training and evaluation details between models and allows correspondence readout. Our implementation

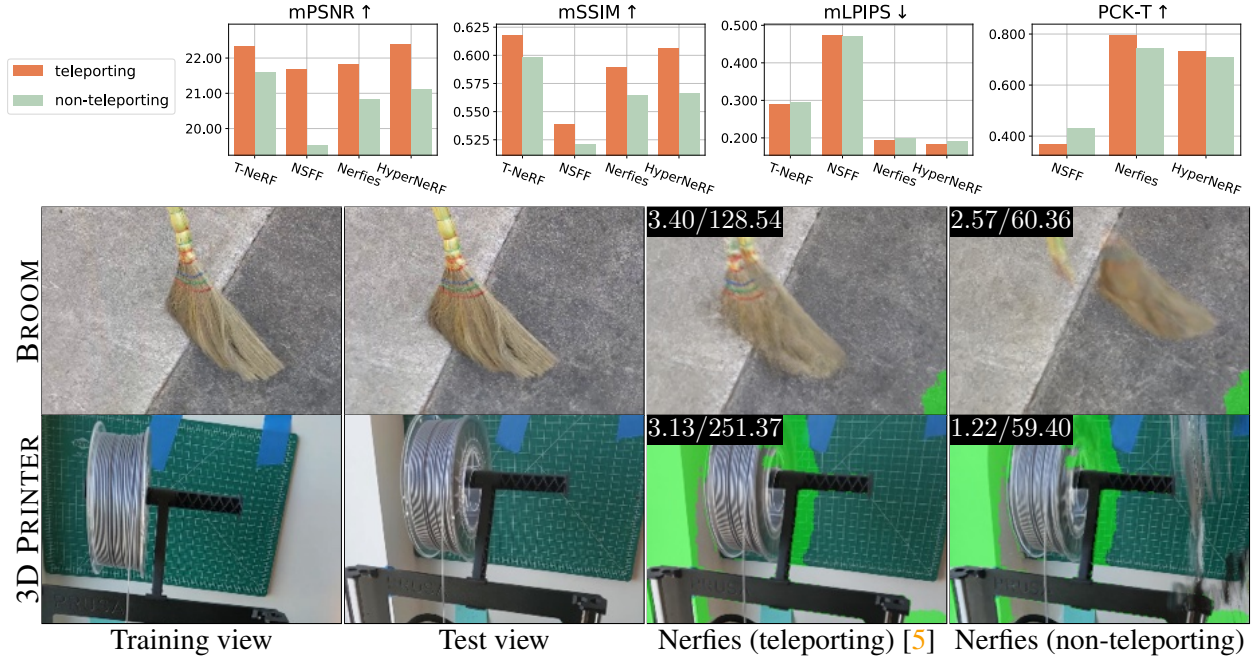


Figure 2.8: **Impact of effective multi-view on the Nerfies-HyperNeRF dataset.** Ω/ω metrics of the input sequence are shown on the top-left. We compare the existing camera teleporting setting and our non-teleporting setting. (Top): Quantitative results of different models using our proposed evaluation metrics. (Bottom): Qualitative comparison using Nerfies as an example. Two settings use the same set of co-visibility masks computed from common training images. Visualizations of other models are in the Appendix.

reproduces the quantitative results in the original papers. We implement T-NeRF in the same codebase. For NSFF [4], we tried both the official code base [47] and a public third-party re-implementation [48], where the former fails to converge on our proposed iPhone dataset while the latter works well. We thus report results using the third-party re-implementation. However, note that both the original and the third-party implementation represent the dynamic scene in normalized device coordinates (NDC). As NDC is designed for forward-facing but not considered inward-facing scenes, layered artifacts may appear due to its log-scale sampling rate in the world space, as shown in Figure 2.9. More details about aligning the training procedure and remaining differences are provided in the Appendix. Code, pretrained models, and data are available on the [project page](#).

2.5.1 Reality check on the Nerfies-HyperNeRF dataset

Impact of effective multi-view. We first study the impact of effective multi-view on the Nerfies-HyperNeRF [5, 7] dataset. In this experiment, we rectify the effective multi-view sequences by only using the left camera during training as opposed to both the left and right cameras, illustrated in Figure 2.4. We denote the original setting as “teleporting” and the rectified sequences as “non-

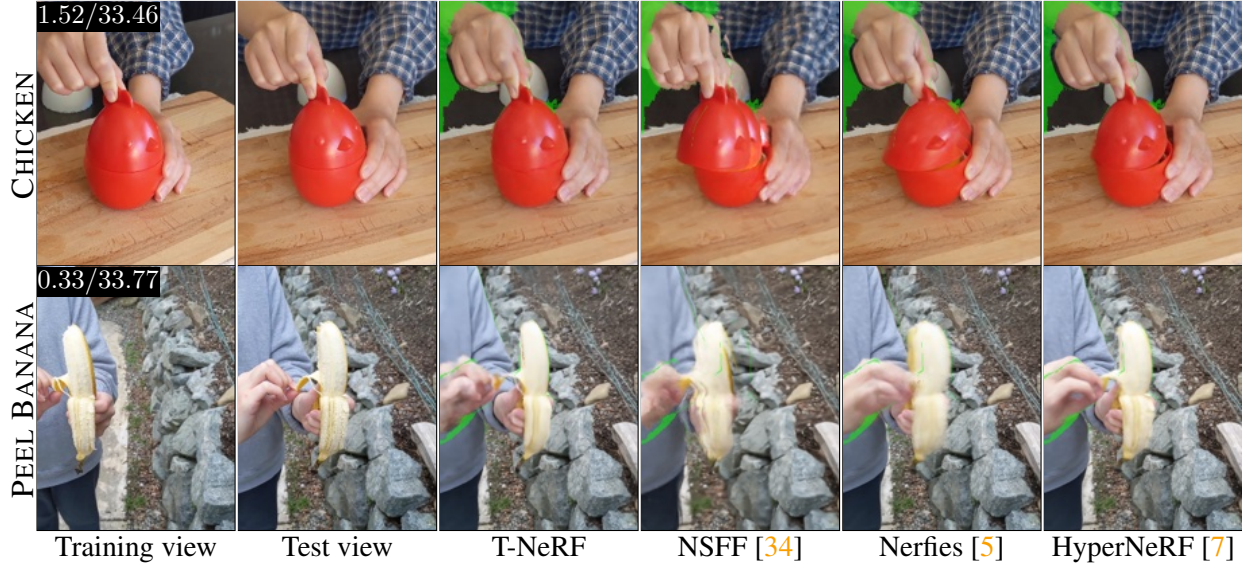


Figure 2.9: **Qualitative results on the Nerfies-HyperNeRF dataset without camera teleportation.** Ω/ω metrics of the input sequence are shown on the top-left. Existing approaches struggle at modeling dynamic regions.

teleporting”. We train all approaches under these two settings with the same held-out validation frames and same set of co-visibility masks computed from common training frames. In Figure 2.8 (Top), all methods perform better across all metrics when trained under the teleporting setting compared to the non-teleporting one, with the exception of PCK-T for NSFF. We conjecture that this is because that NSFF has additional optical flow supervision, which is more accurate without camera teleportation. In Figure 2.8 (Bottom), we show qualitative results using Nerfies (we include visualizations of the other methods in the Appendix). Without effective multi-view, Nerfies fails at modeling physically plausible shape for broom and wires. Our results show that the effective multi-view in the existing experimental protocol inflates the synthesis quality of prior methods, and that truly monocular captures are more challenging.

Benchmark results without camera teleportation. In Table 2.2 and Figure 2.9, we report the quantitative and qualitative results under the non-teleporting setting. Note that our implementation of the T-NeRF baseline performs the best among all four evaluated models in terms of mPSNR and mSSIM. In Figure 2.9, we confirm this result since T-NeRF renders high-quality novel view for both sequences. HyperNeRF produces the most photorealistic renderings, measured by mLPIPS. However it also produces distorted artifacts that do not align well with the ground truth (*e.g.*, the incorrect shape in the CHICKEN sequence).

2.5.2 Reality check on the proposed iPhone dataset

$\Omega = 1.30$ $\omega = 51.53$	mPSNR \uparrow	mSSIM \uparrow	mLPIPS \downarrow	PCK-T \uparrow
T-NeRF	21.55	0.595	0.297	-
NSFF [4]	19.53	0.521	0.471	0.422
Nerfies [5]	20.85	0.562	0.200	0.756
HyperNeRF [7]	21.16	0.565	0.192	0.764

$\Omega = 0.24$ $\omega = 15.18$	mPSNR \uparrow	mSSIM \uparrow	mLPIPS \downarrow	PCK-T \uparrow
T-NeRF	16.96	0.577	0.379	-
NSFF [4]	16.65	0.588	0.369	0.256
Nerfies [5]	16.45	0.570	0.339	0.453
HyperNeRF [7]	16.81	0.569	0.332	0.400

Table 2.2: **Benchmark results on the rectified Nerfies-HyperNeRF dataset.** Please see the Appendix for the breakdown over 7 multi-camera sequences.

Table 2.3: **Benchmark results on the proposed iPhone dataset.** Please see the Appendix for the breakdown over 7 multi-camera sequences of complex motion.

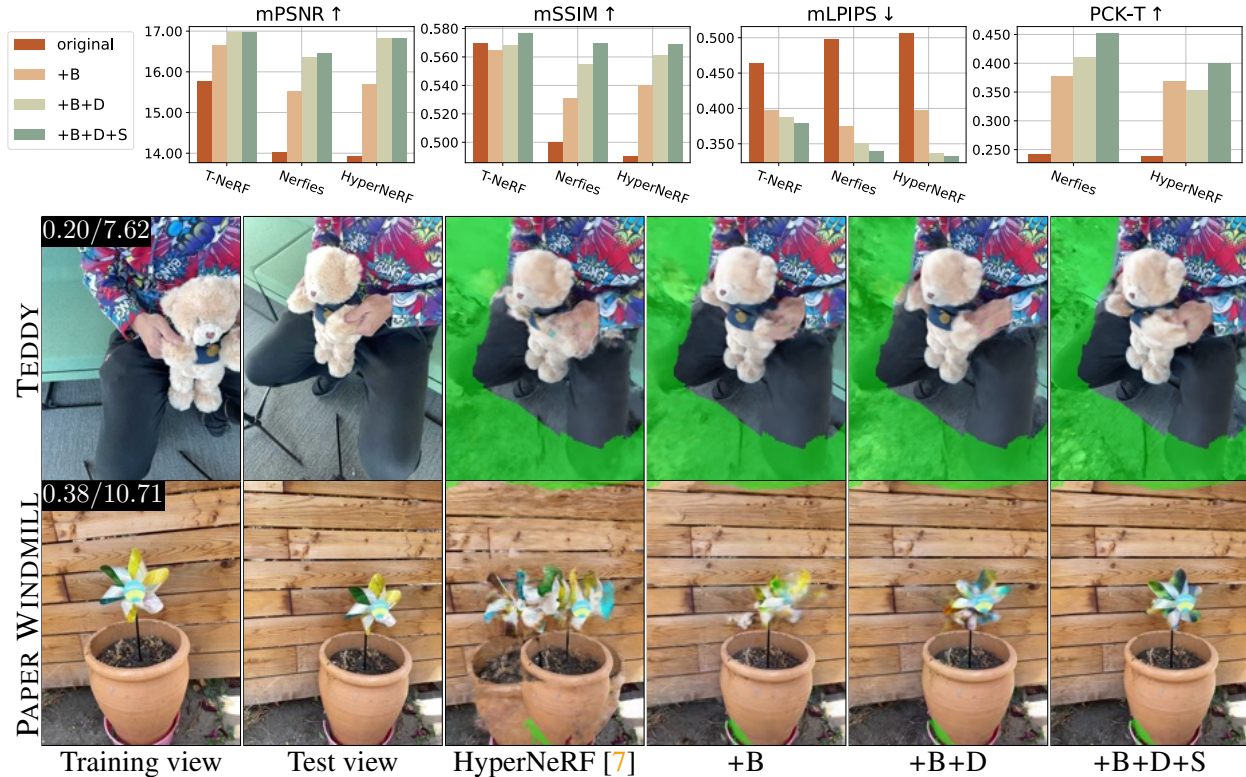


Figure 2.10: **Ablation study on improving the state of the art on the proposed iPhone dataset.** Ω/ω metrics of the input sequence are shown on the top-left. +B, +D, +S denotes random background compositing [32], additional metric depth supervision [1, 4] from iPhone sensor, and surface sparsity regularizer [49], respectively.

Ablation study on improving the state of the art. We find that existing methods perform poorly out-of-the-box on the proposed iPhone dataset with more diverse and complex real-life motions. In Figure 2.10 (Bottom), we demonstrate this finding with HyperNeRF [7] for it achieves the highest

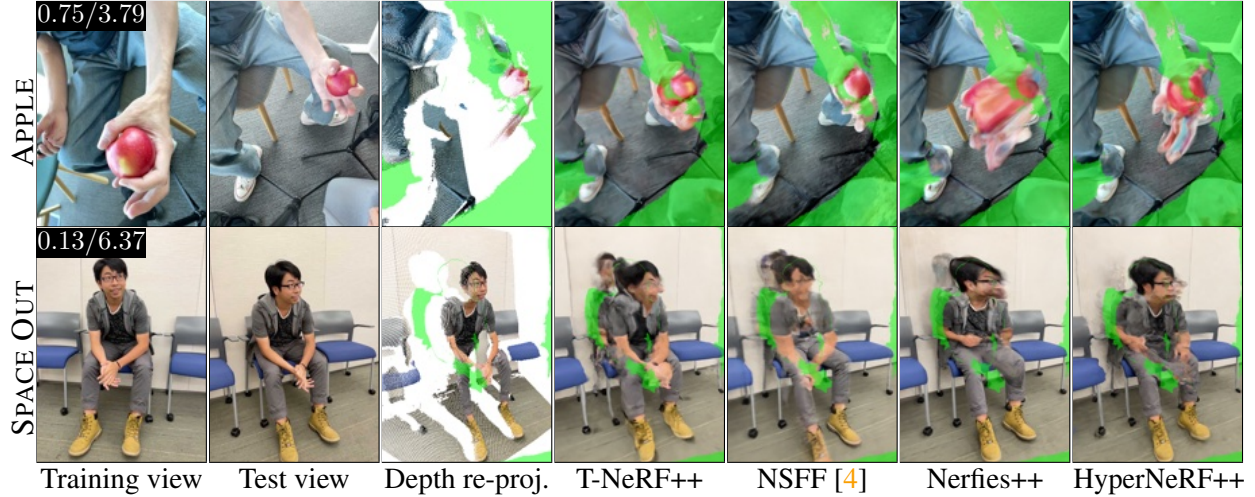


Figure 2.11: **Qualitative results on the proposed iPhone dataset.** Ω/ω metrics of the input sequence are shown on the top-left. The models shown here are trained with all the additional regularizations (+B+D+S) except NSFF. However, existing approaches still struggle to produce high-quality results.

mLPIPS metric on the Nerfies-HyperNeRF dataset. Shown in the 3rd column, HyperNeRF produces visually implausible results with ghosting effects. Thus we explored incorporating additional regularizations from recent advances in neural rendering. Concretely, we consider the following: (+B) random background compositing [32]; (+D) a depth loss on the ray matching distance [1, 4]; and (+S) a sparsity regularization for scene surface [49]. In Figure 2.10 (Top), we show quantitative results from the ablation. In Figure 2.10 (Bottom), we show visualizations of the impact of each regularization. Adding additional regularizations consistently boosts model performance. While we find the random background compositing regularizations particularly helpful, extra depth supervision and surface regularization further improve the quality, *e.g.*, the fan region of the paper windmill.

Benchmarked results. In Figure 2.11, we show qualitative results from our benchmark using the best model settings from the ablation study, denoted as “++”. Note that it is non-trivial to apply the same enhancements to NSFF for its NDC formulation so we keep it as-is. We visualize the lidar depth re-projection from the training view (1st column) to the test view (2nd column), as a reference for qualitative comparison (3rd column). Note that the white region is occluded from the input view, whereas the green region is occluded from the most of input video frames. We observe that existing approaches do not handle complex deformation well. For example, all models fail at fusing a valid human shape on the SPACE OUT sequence. In Table 2.3, we find a similar trend as in the Nerfies-HyperNeRF dataset: the baseline T-NeRF performs the best in terms of mPSNR and mSSIM while HyperNeRF produces the most photorealistic renderings in terms of mLPIPS. The overall synthesis quality and correspondence accuracy of all methods drop considerably compared to the results on the Nerfies-HyperNeRF dataset. Taking Nerfies as an example, it drops 4.4 dB

in mPSNR, 69.6% in mLPIPS, and 40.1% in PCK-T. Our study suggests an opportunity for large improvement when modeling complex motion.

2.6 Discussion and recommendation for future works

In this work, we expose issues in the common practice and establish systematic means to calibrate performance metrics of existing and future works, in the spirit of papers like [50, 51, 52]. We provide initial attempts toward characterizing the difficulty of a monocular video for dynamic view synthesis (DVS) in terms of effective multi-view factors (EMFs). In practice, there are other challenging factors such as variable appearance, lighting condition, motion complexity and more. We leave their characterization for future works. We recommend future works to visualize the input sequences and report EMFs when demonstrating the results. We also recommend future works to evaluate the correspondence accuracy and strive for establishing better correspondences for DVS.

Chapter 3

Shape of Motion: 4D reconstruction from a single video

Monocular dynamic reconstruction is a challenging and long-standing vision problem due to the highly ill-posed nature of the task. Existing approaches are limited in that they either depend on templates, are effective only in quasi-static scenes, or fail to model 3D motion explicitly. In this work, we introduce a method capable of reconstructing generic dynamic scenes, featuring explicit, full-sequence-long 3D motion, from casually captured monocular videos. We tackle the under-constrained nature of the problem with two key insights: First, we exploit the low-dimensional structure of 3D motion by representing scene motion with a compact set of $\mathbb{SE}(3)$ motion bases. Each point’s motion is expressed as a linear combination of these bases, facilitating soft decomposition of the scene into multiple rigidly-moving groups. Second, we utilize a comprehensive set of data-driven priors, including monocular depth maps and long-range 2D tracks, and devise a method to effectively consolidate these noisy supervisory signals, resulting in a globally consistent representation of the dynamic scene. Experiments show that our method achieves state-of-the-art performance for both long-range 3D/2D motion estimation and novel view synthesis on dynamic scenes.

3.1 Introduction

Reconstructing the persistent geometry and their 3D motion across a video is crucial for understanding and interacting with the underlying physical world. While recent years have seen impressive progress in modeling static 3D scenes [36, 53], recovering the geometry and motion of complex dynamic 3D scenes, especially from a single video, remains an open challenge. A number of prior dynamic reconstruction and novel view synthesis approaches have attempted to tackle this problem. However, most methods rely on synchronized multi-view videos [54, 55, 55, 56, 57] or additional LIDAR/depth sensors [58, 59, 60, 61, 62]. Recent monocular approaches can operate on regular dynamic videos, but they typically model 3D scene motion as short-range scene flow between consecutive times [34, 63, 64] or deformation fields that maps between canonical and view space [5, 7, 65], failing to capture 3D motion trajectories persistent over a video.

The longstanding challenge for more general in-the-wild videos lies in the poorly constrained nature of the reconstruction problem. In this work, we tackle this challenge with two key insights. The first is that, while the image space dynamics can be complex and discontinuous, the underlying 3D motion is a composition of continuous simple rigid motions. Our second insight is that data-driven priors provide complementary, though noisy cues, that aggregate well into a globally coherent representation of the 3D scene geometry and motion.

Motivated by these two insights, we represent the dynamic scene as a set of persistent 3D Gaussians, and represent their motion across the video in terms of a compact set of shared $\mathbb{SE}(3)$ motion bases. Unlike traditional scene flow, which computes 3D correspondence between consecutive frames, our representation recovers a persistent 3D trajectory over the whole video, enabling long-range 3D tracking. As the 3D trajectories produced by our method capture the geometric patterns that trace each point’s movement through 3D space and time, we refer to our approach “Shape of Motion”, as shown in Figure 3.1. We show how to fit our explicit scene representation to a general video in-the-wild, by fusing together complementary cues from two main sources: monocular depth estimates per-frame, and 2D track estimates across frames.

We conduct extensive evaluations on both synthetic and real-world dynamic video datasets, and show that our proposed approach significantly outperforms prior monocular dynamic novel view synthesis methods and 3D tracking baselines in both long-range 2D and 3D tracking accuracy. Moreover, we achieve state-of-the-art novel view synthesis quality among all existing methods. In summary, our key contributions include: (1) A new dynamic scene representation enabling both real-time novel view synthesis and globally consistent 3D tracking for any point at any time. (2) A carefully designed framework that optimizes the representation on posed monocular video by leveraging physical motion priors and data-driven priors.

3.2 Related work

3.2.1 Correspondences and tracking

Monocular 3D long-range tracking has not been explored broadly in the literature, but there exist many approaches to perform tracking in 2D image space. A typical way for determining 2D point correspondences relies on optical flows. This involves estimating dense motion fields between image pairs [66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 38, 76, 74, 77, 78, 79, 80, 81]. While effective for consecutive frames, accurate long-term tracking in videos remains a challenge with optical flow methods. Sparse keypoint matching methods can enable long trajectory generation [82, 83, 84, 85, 86], but these methods are primarily intended for sparse 3D reconstruction. Long-range 2D trajectory estimation for arbitrary points has been explored in earlier works, which relied on hand-crafted priors to generate motion trajectories [87, 88, 89, 90, 91, 92]. Recently, there has been a resurgence of interest in this problem, with several works showcasing impressive long-term 2D tracking results on challenging, in-the-wild videos. These approaches employ either test-time optimization where models consolidate noisy short-range motion estimates into a global representation for producing long-term correspondences [93, 94], or data-driven strategies [95, 96,

97], where neural networks learn long-term correspondence estimates from synthetic data [98, 99]. While these methods effectively track any 2D point throughout a video, they lack the knowledge of underlying 3D scene geometry and motions.

Scene flow or 3D motion trajectory is a common representation to model 3D scene motion and point correspondences. Most prior work estimates scene flow directly from Lidar point clouds [58, 59, 60, 61, 62] or RGBD images [100, 101, 102, 103, 104, 105]. In monocular setting, a few recent works proposed to estimate 3D motions through self-supervised learning or test-time optimization strategies [106, 107, 34, 63, 108, 109], but these approaches either focus on a single object, require template priors, or only produce short-range motion correspondences. In contrast, our method does not rely on template priors and is capable of producing long-range 3D trajectories, making it suitable for modeling complex scenes with multiple moving objects.

3.2.2 Dynamic reconstruction and view synthesis

Our work also relates to dynamic 3D scene reconstruction and novel view synthesis problems. In non-rigid reconstruction, early methods often required RGBD sensors [110, 111, 112, 113, 114] or strong hand-crafted priors [115, 116, 117]. Recent work has demonstrated progress toward the reconstruction of dynamic scenes in the wild by integrating monocular depth priors [118, 119, 120, 121, 122]. Recently, Neural Radiance Fields (NeRF)[36] and Gaussian Splat [123] based approaches have achieved state-of-the-art results. Most of these methods [124, 55, 54, 125, 55, 126, 127, 128, 129, 130] require simultaneous multi-view video observations or predefined templates [131, 37, 32] for high-quality novel view outputs. Template-free monocular approaches model dynamic scenes with different types of representations such as video depth maps [40], time-dependent NeRFs [35, 34, 132, 5, 133, 7, 1, 63], and dynamic 3D Gaussians [57, 134, 135, 65]. While significant progress has been made, as DyCheck [136] pointed out, many approaches focus on scenarios with camera teleportation [137, 65] or quasi-static scenes, which do not represent real-world monocular videos. In this work, we focus on modeling casual videos captured by a single camera, a more practical and challenging setup.

3.3 Method

Our method takes as input a sequence of T video frames $\{I_t \in \mathbb{R}^{H \times W \times 3}\}$ of a dynamic scene, the camera intrinsics $\mathbf{K}_t \in \mathbb{R}^{3 \times 3}$, and world-to-camera extrinsics $\mathbf{E}_t \in \mathbb{SE}(3)$ of each input frame I_t . From these inputs, we aim to recover the geometry of the entire dynamic scene and the full-length 3D motion trajectory of every point in the scene.

Unlike most prior dynamic NeRFs methods [118, 63, 136, 1, 133] which render the scene contents through volumetric ray casting and represent the motion implicitly at fixed 3D locations, we model the dense scene elements as a set of canonical 3D Gaussians, but allow them to translate and rotate over entire video through full-length motion trajectories. We adopt explicit point-based representation because it simultaneously allows for both (1) high-fidelity rendering in real-time and (2) full-length 3D tracking of any surface point from any input time.

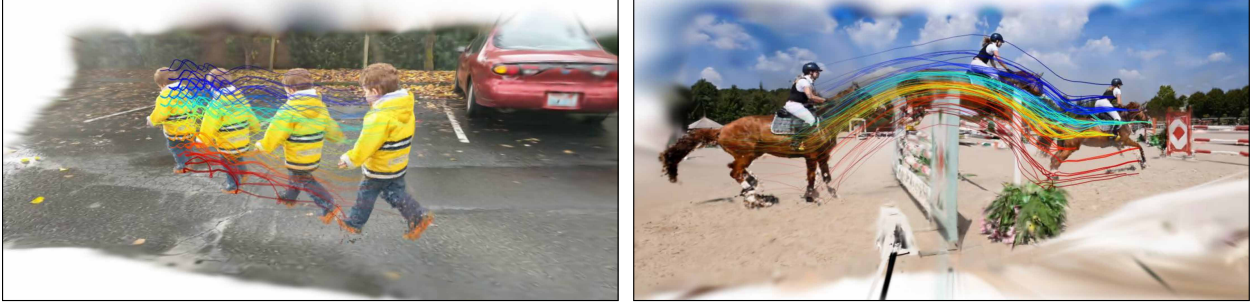


Figure 3.1: **Shape of Motion.** Our approach enables joint 3D long-range tracking and novel view synthesis from a monocular video of a complex dynamic scene. Here, we demonstrate our ability to render moving scene elements at fixed viewpoint with different moments in time. Additionally, we visualize the estimated long-range 3D motion as colorful trajectories. These trajectories reveal distinct geometric patterns that encapsulate the movement of each point through 3D space and time, which leads us to the term “Shape of Motion”.

Optimizing an explicit representation of dynamic 3D Gaussians from a single video is severely ill-posed – at each point in time, the moving subjects in the scene are observed at those poses from only a single viewpoint. In order to overcome this ambiguity, we make two insights: First, we note that, while the projected 2D dynamics might be complex in the video, the underlying 3D motion in the scene is low-dimensional, and composed of simpler units of rigid motion. Second, powerful data-driven priors, namely monocular depth estimates and long-range 2D tracks, provide complementary but noisy signals of the underlying 3D scene. We propose a system that allows us to fuse these noisy estimates together into a globally coherent representation of both the scene geometry and motion.

The following sections introduce our framework. We represent the scene contents as a globally consistent set of 3D Gaussians that live in a canonical space (Section 3.3.1). To recover consistent motion trajectories, we bind the 3D Gaussians to a compact and low-dimensional motion parameterization. In particular, we represent the full-length motion trajectories of each dynamic scene element with a set of compact and low-dimensional $\mathbb{SE}(3)$ motion bases (Section 3.3.2). Finally, we fit these motion bases to the input video frames, constraining our optimization with structural priors and data-driven priors (Section 3.3.3). We show a schematic of our pipeline in Figure 3.2.

3.3.1 Preliminaries: 3D Gaussian splatting

We represent the appearance and geometry of dynamic scene contents with a global set of 3D Gaussians, an explicit and expressive differentiable scene representation [53] for efficient optimization and rendering. We define parameters of each 3D Gaussian in the canonical frame t_0 as $\vec{g}_0 \equiv (\mu_0, \mathbf{R}_0, s, o, \mathbf{c})$, where $\mu_0 \in \mathbb{R}^3$, $\mathbf{R}_0 \in \mathbb{SO}(3)$ are the 3D mean and orientation in the canonical frame, and $s \in \mathbb{R}^3$ the scale, $o \in \mathbb{R}$ the opacity, and $\mathbf{c} \in \mathbb{R}^3$ the color, are persistent properties shared across time.

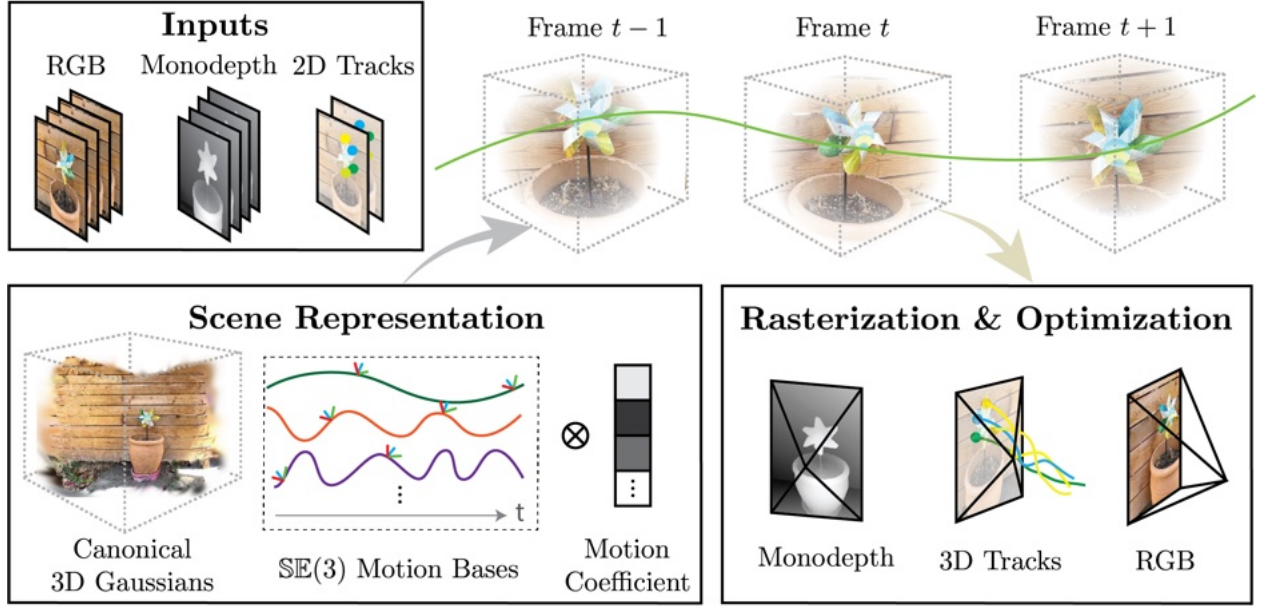


Figure 3.2: **System overview.** Given a single RGB video sequence with known camera poses, along with monocular depth maps and 2D tracks computed from off-the-shelf models [138, 97] as input, we optimize a dynamic scene representation as a set of persistent 3D Gaussians that translate and rotate over time. To capture the low-dimensional nature of scene motion, we model the motion with a set of compact $\text{SE}(3)$ motion bases shared across all scene elements. Each 3D Gaussian’s motion is represented as a linear combination of these global $\text{SE}(3)$ motion bases, weighted by motion coefficients specific to each Gaussian. We supervise our scene representation (canonical 3D Gaussian parameters, per-Gaussian motion coefficients, and global motion bases) by comparing the rendered outputs (RGB, depths and 2D tracks) with the corresponding input signals. This results in a dynamic 3D representation of the scene with explicit long-range 3D scene motion.

To render a set of 3D Gaussians from a camera with world-to-camera extrinsics \mathbf{E} and intrinsics \mathbf{K} , the projections of the 3D Gaussians in the image plane are obtained by 2D Gaussians parameterized as $\boldsymbol{\mu}'_0 \in \mathbb{R}^2$ and $\boldsymbol{\Sigma}'_0 \in \mathbb{R}^{2 \times 2}$ via affine approximation

$$\boldsymbol{\mu}'_0(\mathbf{K}, \mathbf{E}) = \Pi(\mathbf{K}\mathbf{E}\boldsymbol{\mu}_0) \in \mathbb{R}^2, \quad \boldsymbol{\Sigma}'_0(\mathbf{K}, \mathbf{E}) = \mathbf{J}_{\mathbf{KE}}\boldsymbol{\Sigma}_0\mathbf{J}_{\mathbf{KE}}^\top \in \mathbb{R}^{2 \times 2}, \quad (3.1)$$

where Π is perspective projection, and $\mathbf{J}_{\mathbf{KE}}$ is the Jacobian of perspective projection with \mathbf{K} and \mathbf{E} at the point $\boldsymbol{\mu}_0$.

The 2D Gaussians can then be efficiently rasterized into RGB image and depth map via alpha compositing as

$$\hat{\mathbf{I}}(\mathbf{p}) = \sum_{i \in H(\mathbf{p})} T_i \alpha_i \mathbf{c}_i, \quad \hat{\mathbf{D}}(\mathbf{p}) = \sum_{i \in H(\mathbf{p})} T_i \alpha_i \mathbf{d}_i, \quad (3.2)$$

where $\alpha_i = o_i \cdot \exp(-\frac{1}{2}(\mathbf{p} - \boldsymbol{\mu}'_0)^T \boldsymbol{\Sigma}'_0 (\mathbf{p} - \boldsymbol{\mu}'_0))$, and $T_i = \prod_{j=1}^{i-1} (1 - \alpha_j)$. $H(\mathbf{p})$ is the set of Gaussians that intersect the ray shoot from the pixel \mathbf{p} . This process is fully differentiable, and enables direct optimization of the 3D Gaussian parameters.

3.3.2 Dynamic scene representation

Scene motion parameterization. To model a dynamic 3D scene, we keep track of N canonical 3D Gaussians and vary their positions and orientations over time with a per frame rigid transformation. In particular, for a moving 3D Gaussian at time t , its pose parameters $(\boldsymbol{\mu}_t, \mathbf{R}_t)$ are rigidly transformed from the canonical frame t_0 to t via $\mathbf{T}_{0 \rightarrow t} = [\mathbf{R}_{0 \rightarrow t} \quad \mathbf{t}_{0 \rightarrow t}] \in \mathbb{SE}(3)$:

$$\boldsymbol{\mu}_t = \mathbf{R}_{0 \rightarrow t} \boldsymbol{\mu}_0 + \mathbf{t}_{0 \rightarrow t}, \quad \mathbf{R}_t = \mathbf{R}_{0 \rightarrow t} \mathbf{R}_0. \quad (3.3)$$

Rather than modeling the 3D motion trajectories independently for each Gaussian, we define a set of $B \ll N$ learnable basis trajectories $\{\mathbf{T}_{0 \rightarrow t}^{(b)}\}_{b=1}^B$ that are globally shared across all Gaussians [137]. The transformation $\mathbf{T}_{0 \rightarrow t}$ at each time t is then computed by a weighted combination of this global set of basis trajectories through per-point basis coefficients $\mathbf{w}^{(b)}$ via

$$\mathbf{T}_{0 \rightarrow t} = \sum_{b=0}^B \mathbf{w}^{(b)} \mathbf{T}_{0 \rightarrow t}^{(b)}, \quad (3.4)$$

where $\|\mathbf{w}^{(b)}\| = 1$. In our implementation, we parameterize $\mathbf{T}_{0 \rightarrow t}^{(b)}$ as 6D rotation [139] and translation, and perform the weighted combination separately on each with the same weight $\mathbf{w}^{(b)}$. During optimization, we jointly learn the set of global motion bases and motion coefficients of each 3D Gaussian. These compact motion bases explicitly regularize the trajectories to be low-dimensional, encouraging the 3D Gaussians that move similarly to each other to be represented by similar motion coefficients.

Rasterizing 3D trajectories. Given this representation, we now describe how we obtain pixelwise 3D motion trajectory at any query frame I_t . we take a similar approach to Wang *et al.* [93] and rasterize the motion trajectories of 3D Gaussians into query frame I_t . Namely, for a query camera at time t with intrinsics \mathbf{K}_t and extrinsics \mathbf{E}_t , we perform rasterization to obtain a map ${}^w\hat{\mathbf{X}}_{t \rightarrow t'} \in \mathcal{R}^{H \times W \times 3}$ that contains the expected 3D world coordinates of the surface points corresponding to each pixel at target time t'

$${}^w\hat{\mathbf{X}}_{t \rightarrow t'}(\mathbf{p}) = \sum_{i \in H(\mathbf{p})} T_i \alpha_i \boldsymbol{\mu}_{i,t'}, \quad (3.5)$$

where $H(\mathbf{p})$ is the set of Gaussians that intersect the pixel \mathbf{p} at query time t .

The 2D correspondence location at time t' for a given pixel \mathbf{p} , $\hat{\mathbf{U}}_{t \rightarrow t'}(\mathbf{p})$, and the corresponding depth value at time t' , $\hat{\mathbf{D}}_{t \rightarrow t'}(\mathbf{p})$ can then be written as

$$\hat{\mathbf{U}}_{t \rightarrow t'}(\mathbf{p}) = \Pi(\mathbf{K}_{t'} {}^c\hat{\mathbf{X}}_{t \rightarrow t'}(\mathbf{p})), \quad \hat{\mathbf{D}}_{t \rightarrow t'}(\mathbf{p}) = \left({}^c\hat{\mathbf{X}}_{t \rightarrow t'}(\mathbf{p}) \right)_{[3]} \quad (3.6)$$

where ${}^c\hat{\mathbf{X}}_{t \rightarrow t'}(\mathbf{p}) = \mathbf{E}_{t'} {}^w\hat{\mathbf{X}}_{t \rightarrow t'}(\mathbf{p})$, Π is a perspective projection operation, and $(\cdot)_{[3]}$ is the third element of a vector.

3.3.3 Optimization

We prepare the following estimates using off-the-shelf methods in our optimization: 1) masks for the moving objects for each frame $\{\mathbf{M}_t\}$, which can be easily obtained using Track-Anything [140, 141] with a few user clicks, 2) monocular depth maps $\{\mathbf{D}_t\}$ computed using state-of-the-art relative depth estimation method Depth Anything [138] and 3) long-range 2D tracks $\{\mathbf{U}_{t \rightarrow t'}\}$ for foreground pixels from state-of-the-art point tracking method TAPIR [97]. We align the relative depth maps with the metric depth maps by computing a per-frame global scale and shift and use them for our optimization, as we found relative depth maps tend to contain finer details. We treat the lifted 2D tracks unprojected with the aligned depth maps as noisy initial 3D track observations $\{\mathbf{X}_t\}$ for the moving objects. For the static part of the scene, we model them using standard static 3D Gaussians and initialize their 3D locations by unprojecting them into the 3D space using the aligned depth maps. The static and dynamic Gaussians are jointly optimized and rasterized together to form an image. We focus on describing the optimization process for dynamic Gaussians below.

Initialization. We first select the canonical frame t_0 to be the frame in which the most 3D tracks are visible, and initialize the Gaussian means in the canonical frame μ_0 as N randomly sampled 3D tracks locations from this set of initial observations. We then perform k-means clustering on the vectorized velocities of the noisy 3D tracks $\{\mathbf{X}_t\}$, and initialize the motion bases $\{\mathbf{T}_{0 \rightarrow t}^{(b)}\}_{b=1}^B$ from these B clusters of tracks. Specifically, for the set of trajectories $\{\mathbf{X}_t\}_b$ belonging to cluster b , we initialize the basis transform $\mathbf{T}_{0 \rightarrow \tau}^{(b)}$ using weighted Procrustes alignment between the point sets $\{\mathbf{X}_0\}_b$ and $\{\mathbf{X}_\tau\}_b$ for all $\tau = 0, \dots, T$, where the weights are computed using uncertainty and visibility scores from TAPIR predictions. We initialize $\mathbf{w}^{(b)}$ of each Gaussian to decay exponentially with its distance from the center of cluster b in the canonical frame.

We then optimize μ_0 , $\mathbf{w}^{(b)}$, and set of basis functions $\{\mathbf{T}_{0 \rightarrow t}^{(b)}\}_{b=1}^B$ to fit the observed 3D tracks with an ℓ_1 -loss under temporal smoothness constraints.

Training. We supervise the dynamic Gaussians with two sets of losses. The first set of losses comprise our reconstruction loss to match the per-frame pixelwise color, depth, and masks inputs. The second set of losses enforce consistency of correspondences across time. Specifically, during each training step, we render the image $\hat{\mathbf{I}}_t$, depth $\hat{\mathbf{D}}_t$, and mask $\hat{\mathbf{M}}_t$ from their corresponding training cameras $(\mathbf{K}_t, \mathbf{E}_t)$ according to Equation 3.2.

We supervise these predictions with a reconstruction loss applied independently per-frame

$$L_{\text{recon}} = \|\hat{\mathbf{I}} - \mathbf{I}\|_1 + \lambda_{\text{depth}} \|\hat{\mathbf{D}} - \mathbf{D}\|_1 + \lambda_{\text{mask}} \|\hat{\mathbf{M}} - \mathbf{1}\|_1. \quad (3.7)$$

The second set of losses supervises the motion of the Gaussians between frames. Specifically, we additionally render the 2D tracks $\hat{\mathbf{u}}_{t \rightarrow t'}$ and reprojected depths $\hat{\mathbf{D}}_{t \rightarrow t'}$, for a pair of randomly sampled query time t and target time t' . We supervise these rendered correspondences with the lifted long-range 2D track estimates via

$$L_{\text{track-2d}} = \|\mathbf{U}_{t \rightarrow t'} - \hat{\mathbf{U}}_{t \rightarrow t'}\|_1, \text{ and } L_{\text{track-depth}} = \|\hat{\mathbf{d}}_{t \rightarrow t'} - \hat{\mathbf{D}}(\mathbf{U}_{t \rightarrow t'})\|_1. \quad (3.8)$$

Finally, we enforce a distance-preserving loss between randomly sampled dynamic Gaussians and their k-nearest neighbors. Let $\hat{\mathbf{X}}_t$ and $\hat{\mathbf{X}}_{t'}$ denote the location of a Gaussian at time t and t' , and

$\mathcal{C}_k(\hat{\mathbf{X}}_t)$ denote the set of k -nearest neighbors of $\hat{\mathbf{X}}_t$, we define

$$L_{\text{rigidity}} = \left\| \text{dist}(\hat{\mathbf{X}}_t, \mathcal{C}_k(\hat{\mathbf{X}}_t)) - \text{dist}(\hat{\mathbf{X}}_{t'}, \mathcal{C}_k(\hat{\mathbf{X}}_{t'})) \right\|_2^2, \quad (3.9)$$

where $\text{dist}(\cdot, \cdot)$ measures Euclidean distance.

Implementation details. For in-the-wild videos, we obtain their camera parameters with the following procedure: we first run UniDepth [142] to get the camera intrinsics and metric depth maps, and then run Droid-SLAM [143] in RGB-D mode with UniDepth’s depth maps to obtain the camera poses. This process is efficient and provides accurate camera parameters. For evaluating our methods on public benchmark, we use the camera annotations that come with the datasets (e.g., from COLMAP [144] or simulation).

We optimize our model using Adam Optimizer [145]. We perform 1000 iterations of optimization for the initial fitting and 500 epochs for joint optimization, respectively. The number of $\mathbb{SE}(3)$ bases B is set to 20 for all of our experiments. We initialize 40k dynamic Gaussians for the dynamic part and 100k static Gaussians for the static part of the scene, respectively. We perform the same adaptive Gaussian controls for dynamic and static Gaussians as per 3D-GS [123]. Training on a sequence of 300 frames of 960×720 resolution takes about 2 hours to finish on an A100 GPU. Our rendering FPS is around 40 fps.

3.4 Experiments

We evaluate our performance quantitatively and qualitatively on a broad range of tasks: long range 3D point tracking, long-range 2D point tracking, and novel view synthesis. We focus our evaluation in particular on datasets that exhibit substantial scene motion. In particular, the iPhone dataset [136] features casual captures of real-world scenes that closely match our target scenarios. It provides comprehensive annotations, including simultaneous validation views, lidar depth, sparse 2D point correspondences across the entire video, and can be used to evaluate our performance on all three tasks. Given the challenge of obtaining precise 3D track annotations for real data, we also evaluate performance using the scenes from the synthetic MOVi-F Kubric dataset [146].

3.4.1 Task specification

Long-range 3D tracking. Our primary task is estimating 3D scene motion for any pixel over long period of time (up to over 10 seconds). For this purpose, we extend the metrics for scene flow evaluation introduced in RAFT-3D [101] to evaluate long-range 3D tracking. Specifically, we report the 3D end-point-error (EPE), which measures the Euclidean distance between the ground truth 3D tracks and predicted tracks at each target time step. In addition, we report the percentage of points that fall within a given threshold of the ground truth 3D location $\delta_{3D}^{0.5} = 5\text{cm}$ and $\delta_{3D}^{1.0} = 10\text{cm}$ in metric scale.

Method	3D Tracking			2D Tracking			View Synthesis		
	EPE ↓	$\delta_{3D}^{0.05}$ ↑	δ_{3D}^{10} ↑	AJ ↑	$<\delta_{avg}$ ↑	OA ↑	PSNR ↑	SSIM ↑	LPIPS ↓
T-NeRF [136]	-	-	-	-	-	-	15.60	0.55	0.55
HyperNeRF [7]	0.182	28.4	45.8	10.1	19.3	52.0	15.99	0.59	0.51
DynIBaR [63]	0.252	11.4	24.6	5.4	8.7	37.7	13.41	0.48	0.55
Deformable-3D-GS [65]	0.151	33.4	55.3	14.0	20.9	63.9	11.92	0.49	0.66
CoTracker [96]+DA [138]	0.202	34.3	57.9	24.1	33.9	73.0	-	-	-
TAPIR [97]+DA [138]	0.114	38.1	63.2	27.8	41.5	67.4	-	-	-
Ours	0.082	43.0	73.3	34.4	47.0	86.6	16.72	0.63	0.45
Ours + 2DGS [147]	0.097	47.3	71.3	35.8	47.0	87.3	16.75	0.65	0.40

Table 3.1: **Evaluation on iPhone dataset.** Our method achieves SOTA performance on all tasks of 3D point tracking, 2D point tracking, and novel view synthesis. The baselines that perform best on 2D and 3D tracking (TAPIR [97]+DA [138] and CoTracker [96]+DA [138]) are unable to synthesize novel viewpoints of the scene, while the methods that perform best in novel view synthesis struggle with or fail to produce 2D and 3D tracks. Our method achieves a significant boost in all three tasks above baselines. We can also substitute the underlying 3DGS with 2DGS [147] to improve scene geometry. We include details about 2DGS training in the Supplemental.

Long-range 2D tracking. Our globally consistent 3D motion representation can be easily projected onto image plane to get long-range 2D tracks. We therefore also evaluate 2D tracking performance in terms of both position accuracy and occlusion accuracy following the metrics introduced in the TAP-Vid benchmark [98], and report the Average Jaccard (AJ), average position accuracy ($<\delta_{avg}$), and Occlusion Accuracy (OA).

Novel view synthesis. We measure our method’s novel view synthesis quality as a comprehensive assessment for geometry, appearance, and motion. We evaluate on the iPhone dataset [136] which provides validation views and report co-visibility masked image metrics [136]: mPSNR, mSSIM [9] and mLPIPS [42, 43].

3.4.2 Baselines

Our method represents dynamic 3D scene comprehensively with explicit long-range 3D scene motion estimation, which also allows for novel view synthesis. While no existing method achieves the exact same goals as ours, there are methods that focus on sub-tasks related to our problem, such as dynamic novel view synthesis, 2D tracking, or monocular depth estimation. We therefore adapt existing methods as our baselines which are introduced below.

While dynamic novel view synthesis approaches focus primarily on the photometric reconstruction quality and do not explicitly output 3D point tracks, we can adapt some of their representations to derive 3D point tracks for our evaluation. For HyperNeRF [7], we compose the learned inverse mapping (from view space to canonical space) and a forward mapping solved via root-finding [136, 45] to produce 3D tracks at query points. DynIBaR [63] produces short-range view-to-view

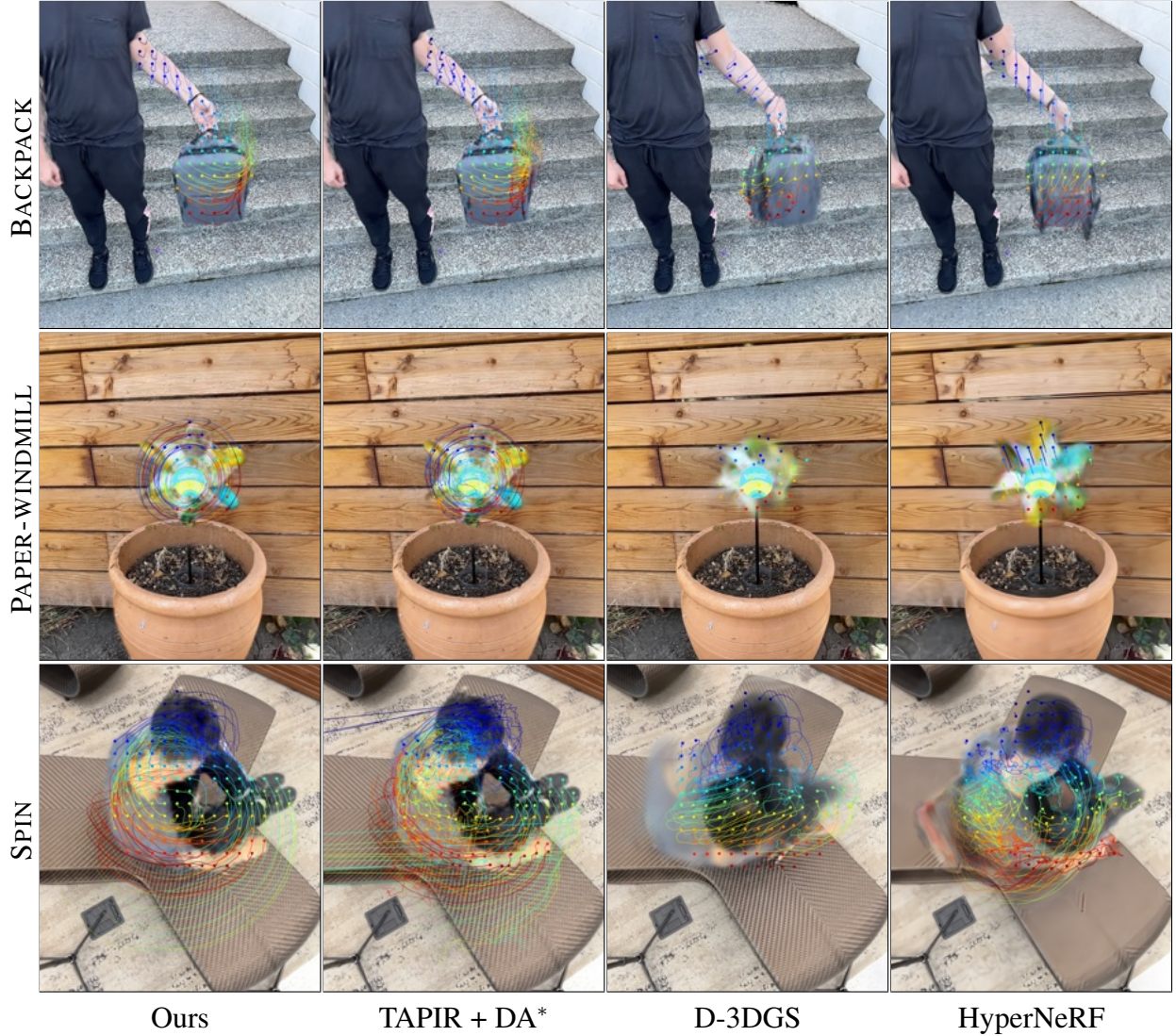


Figure 3.3: **3D tracking visualization on iPhone dataset.** We render novel views for each method and overlay their predicted 3D tracks on top of the rendered images. For clarity, we only display a segment of the trails spanning 50 frames for a specific set of grid query points. However, it is important to note that our method can generate dense, full-length 3D tracks. *Note that TAPIR + DA cannot produce novel view rendering, and we overlay their tracking results with our rendering to make it easier to interpret.

scene flow, which we chain into long-range 3D tracks for our evaluation. Deformable-3D-GS (D-3DGS) [65] represents dynamic scenes using 3D Gaussians [53] in the canonical space and a deformation MLP network that deforms the canonical 3D Gaussians into each view, which naturally allows 3D motion computation. Finally, T-NeRF [136] models dynamic scenes using time as MLP

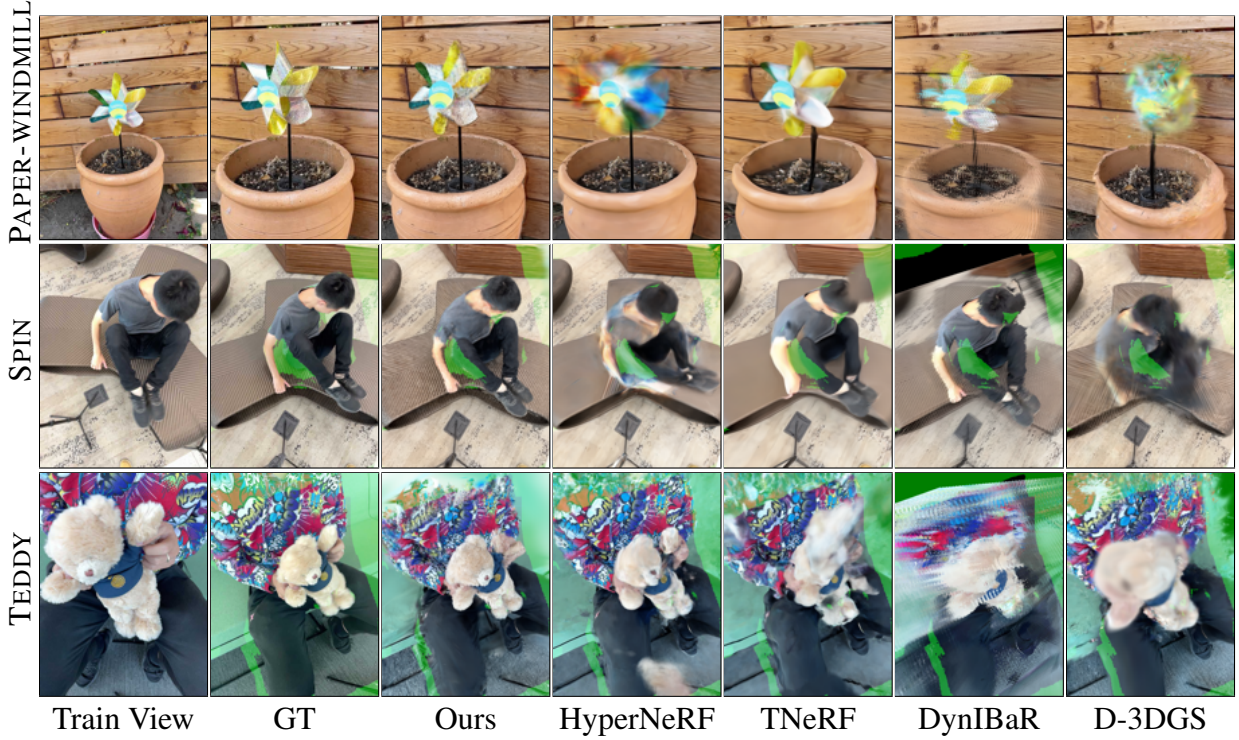


Figure 3.4: **Qualitative comparison of novel-view synthesis on iPhone dataset.** The leftmost image in each row shows the training view at the same time step as the validation view. The regions highlighted in green indicate areas excluded from evaluation due to the lack of co-visibility between training and validation views. Please see the supplemental video for more qualitative results and comparisons.

input in addition to 3D locations, which does not provide a method for extracting 3D motion, and hence they are not considered for 2D/3D tracking evaluation. We evaluate the 3D tracks of these methods only on the iPhone dataset, because we found none can handle the Kubric scene motion.

In addition to dynamic view synthesis baselines, we would also like to include baselines that focus on estimating 3D tracks. However, we did not find prior work that produces long-range 3D tracks from generic monocular videos, hence we adapt existing SOTA methods for long-range 2D tracking and monocular depth estimation. Specifically, we take the state-of-art long-range 2D tracks from TAPIR [97] and CoTracker [96] and lift them into 3D scene motion using monocular depth maps produced by Depth Anything (DA) [138]. We compute the correct scale and shift for each relative depth map from Depth Anything to align them with the scene. The two resulting baselines are called “CoTracker [96] + DA [138]” and “TAPIR [97] + DA [138]”. Note that these baselines can only produce 3D tracks for visible regions, as the depth values for occluded points are unknown from such a 2.5D representation. In contrast, our global representation allows for modeling 3D motion through occlusions.

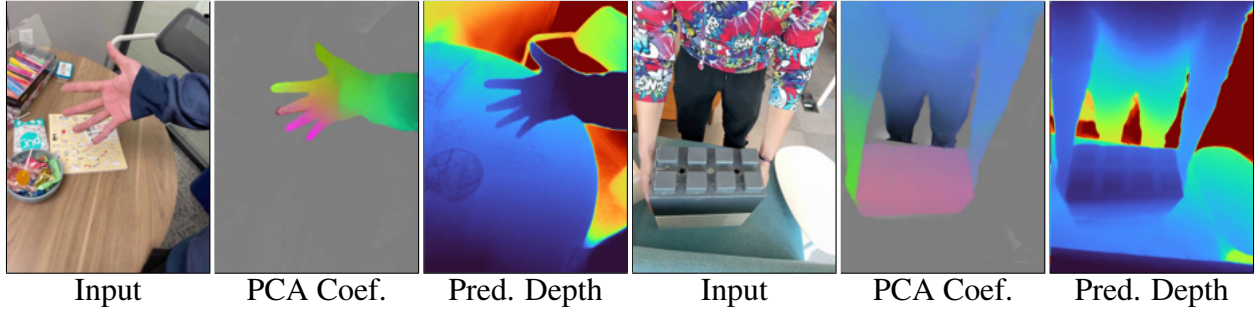


Figure 3.5: **Visualization of motion coefficients after PCA and predicted depth maps on iPhone dataset.** The motion coefficients encode information regarding rigid moving components. For instance, our motion coefficient PCA produces constant color for the block in the second example which exhibits rigid motion.

3.4.3 Evaluation on iPhone dataset

iPhone dataset [136] contains 14 sequences of 200-500 frames featuring various types of challenging real world scene motion. All sequences are recorded using a handheld moving camera in a casual manner, with 7 of them additionally featuring two synchronized static cameras with a large baseline for novel view synthesis evaluation. It also comes with metric depth from the lidar sensors and annotations of 5 to 15 keypoints at ten equally spaced time steps for each sequence. For 3D tracking evaluation, we generate the groundtruth 3D tracks by lifting the 2D keypoint annotations into 3D using lidar depth, and masking out points that are occluded or with invalid lidar depth values.

All experiments are conducted with the original instead of half resolution as in [136] given that our method can handle high-res video input. We also discard SPACE-OUT and WHEEL scenes due to camera and lidar error. We find that the original camera annotations from ARKit is not accurate enough, and refine them using global bundle adjustment from COLMAP [148].

We report the quantitative comparison in Table 3.1 which shows that our method outperforms all baselines on all tasks by a substantial margin. On 3D and 2D tracking, we show clear improvement over naive baselines of 2D tracks plus depth maps, and significant improvement over all novel view synthesis methods, e.g., nearly halving the EPE of the second best method Deformable-3D-GS [65]. Our method also achieves the best novel view synthesis quality across dynamic NeRF, IBR, and 3D-GS-based baselines.

Figure 3.3 shows qualitative comparison of the 3D tracking results. To illustrate 3D tracking quality, we render the novel views and plot the predicted 3D tracks of the given query points onto the novel views. Since “TAPIR + DA” cannot perform novel view synthesis, we overlay their track predictions onto our renderings to aid interpretation. D-3DGS [65] and HyperNeRF [7] fail to capture the significant scene motion in the paper-windmill and spin sequence, resulting in structure degradation and blurry rendering. “TAPIR + DA” can track large motions, but their 3D tracks tend to be noisy and erroneous. In contrast, our method not only generates the highest-quality novel views but also the most smooth and accurate 3D tracks.

Figure 3.4 provides additional novel view synthesis comparison on the validation views. In

Method	EPE↓	$\delta_{3D}^{05} \uparrow$	$\delta_{3D}^{10} \uparrow$
CoTracker [96]+DA [138]	0.19	34.4	56.5
TAPIR [97]+DA [138]	0.20	34.0	56.2
Ours	0.16	39.8	62.2

Table 3.2: **3D tracking evaluation on Kubric dataset.** Evaluation of 3D tracking performance comparing our method against baselines using CoTracker and TAPIR with depth estimation. Our approach achieves the best performance across all metrics including End Point Error (EPE) and delta-3D accuracy thresholds.

Figure 3.5, we provide additional visualization of the outputs from our representation. We visualize the rendering of the first three components of PCA decomposition for the motion coefficients, which correlates well with the rigid groups of the moving object.

3.4.4 Evaluation on the Kubric dataset

The Kubric MOVi-F dataset contains short 24-frame videos of scenes of 10-20 objects, rendered with linear camera movement and motion blur. Multiple rigid objects are tossed onto the scene, at a speed that far exceeds the speed of the moving camera, making it a similar capture scenario to in-the-wild capture scenarios. It provides dense comprehensive annotations, including ground truth depth maps, camera parameters, segmentation masks, and point correspondences that are dense across time. We use 30 scenes from the MOVi-F validation set to evaluate the accuracy of our long-range 3D point tracks for all points in time.

We demonstrate our method on the synthetic Kubric MOVi-F dataset in long-range 3D point tracking across time. Of the above baselines, only “CoTracker[96]+DA [138]” and “TAPIR[96]+DA [138]” yield 3D tracks for these scenes. For all baselines, we provide the ground truth camera intrinsics and extrinsics, and monocular depth estimates that have been aligned to the ground truth depth map. We obtain point tracks for all non-background pixels for each method.

We report our quantitative 3D point tracking metrics in Table 3.2, and find that across all metrics, our method outperforms the baselines. Moreover, we find qualitatively that the optimized motion coefficients of the scene representation are coherently grouped with each moving object in the scene. We demonstrate this in Figure 3.6, where we show the first 3 PCA components of our optimized motion coefficients of evaluation scenes.

3.4.5 Ablation studies

We ablate various components of our method on the iPhone dataset in 3D tracking in Table 3.3.

We first validate our choices of motion representation, namely the $\mathbb{SE}(3)$ motion bases parameterization, with two ablations: 1) “Per-Gaussian Transl.”: replacing our motion representation with naive per-Gaussian translational motion trajectories, and 2) “Transl Bases.”: keeping the motion

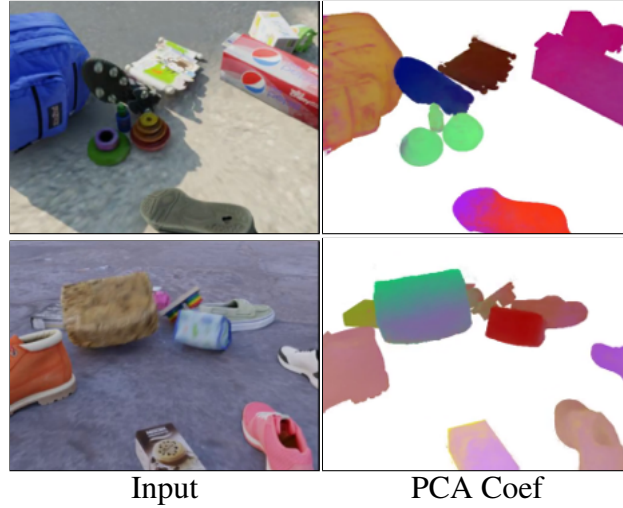


Figure 3.6: **First three PCA components of the optimized motion coefficients.** Our method recovers coherent motion patterns on the Kubric dataset, where different moving objects are clearly distinguished by the motion coefficients.

Methods	$\mathbb{SE}(3)$	Motion Basis	2D tracks	Initialization	EPE↓	$\delta_{3D}^{.05}$ ↑	$\delta_{3D}^{.10}$ ↑
Ours (Full)	✓	✓	✓	✓	0.082	43.0	73.3
Transl. Bases		✓	✓	✓	0.093	42.3	69.9
Per-Gaussian Transl.			✓	✓	0.087	41.2	69.2
No $\mathbb{SE}(3)$ Init.	✓	✓	✓		0.111	39.3	65.7
No 2D Tracks	✓	✓			0.141	30.4	57.8

Table 3.3: **Ablation studies on iPhone dataset.** We systematically evaluate the contribution of each component in our method: $\mathbb{SE}(3)$ motion parameterization, motion basis functions, 2D track initialization, and $\mathbb{SE}(3)$ initialization. Each component contributes to improved 3D tracking performance as measured by EPE and delta-3D metrics.

bases representation but only using translational bases instead of $\mathbb{SE}(3)$. We find $\mathbb{SE}(3)$ bases lead to a noticeable improvement in 3D tracking compared to both translational bases and per-Gaussian translational motion representation.

Next, we ablate our training strategies including initialization and supervision signal. We conduct an ablation of “No $\mathbb{SE}(3)$ Init.”, where instead of performing our initial $\mathbb{SE}(3)$ fitting stage, we initialize the translational part of the motion bases with randomly selected noisy 3D tracks formed by directly lifting input 2D tracks using depth maps into 3D, and the rotation part as identity. We find that skipping this initialization noticeably hurts performance. Lastly, we remove the 2D track supervision entirely (“No 2D Tracks”) and find it to lead to significant drop in performance, which verifies the importance of the 2D track supervision for 3D tracking.

3.5 Discussion and conclusion

Limitations. Although our approach demonstrates promising steps towards long-range 3D tracking and accurate reconstruction of complex 3D dynamic scenes, several limitations remain. Similar to most prior monocular dynamic view synthesis methods, our system still requires per-scene test-time optimization, which hinders its use in streamable applications. In addition, it cannot handle large changes in camera viewpoint, which require some kind of generative capabilities or data-driven priors to hallucinate unseen regions. Moreover, our approach relies on accurate camera parameters obtained from off-the-shelf algorithms, suggesting potential failure in scenes dominated by texture-less regions or moving objects. Finally, our method relies on user input to generate the mask of moving objects. A promising research direction would be to design a feed-forward network approach for jointly estimating camera poses, scene geometry, and motion trajectories from an unconstrained monocular video.

Please note that since we wrote this paper, there have been several concurrent works posted on arXiv [149, 150, 151, 152, 153] that also address this setup of monocular 4D reconstruction from causal videos. As far as we know, all of them are optimization-based approaches, also utilizing strong off-the-shelf data-driven priors. We leave it to future work to compare these approaches.

Conclusion. We presented a new method for joint long-range 3D tracking and novel view synthesis from dynamic scene captured by a monocular video. Our approach models dense dynamic scene elements with a global set of 3D Gaussians that translate and rotate over time. We regularize the full-length 3D motion trajectories of each Gaussian, ensuring smoothness and low dimensionality, by parameterizing them as linear combinations of a compact set of rigid motion bases. To overcome the ill-posed nature of this problem in monocular capture settings, we further design the model to regularize by consolidating noisy data-driven observations into a globally consistent estimate of scene appearance, geometry, and motion. Extensive evaluations on both synthetic and real benchmarks demonstrate that our approach significantly improves upon prior state-of-the-art methods in both 2D/3D long-range tracking and novel view synthesis tasks.

Chapter 4

Stable Virtual Camera: Generative view synthesis with diffusion models

We present STABLE VIRTUAL CAMERA (SEVA), a generalist diffusion model that creates novel views of a scene, given any number of input views and target cameras. Existing works struggle to generate either large viewpoint changes or temporally smooth samples, while relying on specific task configurations. Our approach overcomes these limitations through simple model design, optimized training recipe, and flexible sampling strategy that generalize across view synthesis tasks at test time. As a result, our samples maintain high consistency without requiring additional 3D representation-based distillation, thus streamlining view synthesis in the wild. Furthermore, we show that our method can generate high-quality videos lasting up to half a minute with seamless loop closure. Extensive benchmarking demonstrates that SEVA outperforms existing methods across different datasets and settings.

4.1 Introduction

Novel view synthesis (NVS) aims to generate realistic, 3D-consistent images of a scene from arbitrary camera viewpoints given any number of camera-posed input views. Traditional methods, which rely on dense input views, treat NVS as a 3D reconstruction and rendering problem [36, 154, 123], but this approach fails with sparse inputs. Generative view synthesis addresses this limitation by leveraging modern deep network priors [155, 156], enabling immersive 3D interactions in uncontrolled environments without the need to capture large image sets per scene. In this work, we focus on generative view synthesis and, unless otherwise specified, refer to it simply as NVS for clarity.

Despite recent progress [157, 158, 159, 160, 161, 162, 163], NVS in the wild remains limited due to two key challenges: First, existing methods struggle to generate both large viewpoint changes [161, 160] and temporally smooth samples [157, 164, 158, 159] while being constrained by rigid task configurations, such as a fixed number of input and target views [158, 162, 160, 163], reviewed in Table 4.1. Second, their sampling consistency is often insufficient, necessitating

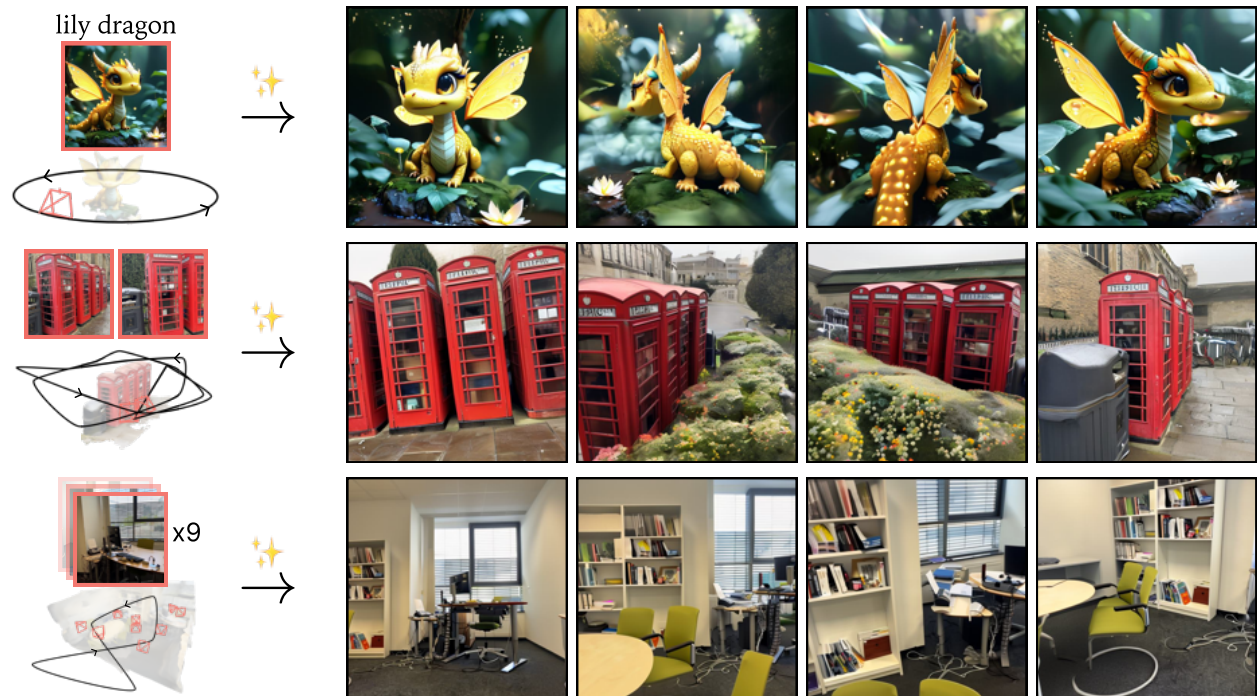


Figure 4.1: **Generative view synthesis.** STABLE VIRTUAL CAMERA generates novel views from any number of input views and target cameras, which the user can specify anywhere. We show three examples: single view with simple orbit camera trajectory (top); two views with long camera trajectory (middle); and nine views with large spatial range (bottom). Please visit our website for video samples.

additional NeRF distillation to fuse inconsistent results into a coherent representation [164, 158, 159]. These limitations hinder their applicability across diverse NVS tasks, which we address in this work.

We present STABLE VIRTUAL CAMERA¹ (SEVA), a diffusion-based NVS model that generalizes across a spectrum of view synthesis tasks without requiring NeRF distillation. With a single network, SEVA generates high-quality novel views that strike both large viewpoint changes and temporal smoothness, while supporting any number of input and target views. Our approach simplifies the NVS pipeline without requiring distillation from a 3D representation, thus streamlining it for real-world applications. For the first time, we demonstrate high-quality videos lasting up to half a minute with precise camera control and seamless loop closure in 3D. We highlight these results in Figure 4.1 and showcase more examples of camera control in Figure 4.2.

To achieve this, we carefully design our pipeline in three key aspects: model design, training recipe, and sampling method at inference. First, SEVA avoids explicit 3D representations within the

¹We are naming this model in tribute to the Virtual Camera [165] cinematography technology, a pre-visualization technique to simulate real-world camera movements.

network, allowing the model to inherit strong priors from pre-trained 2D models. Second, during training, we carefully craft our view selection strategy to cover both small and large viewpoint changes, ensuring strong generalization to diverse NVS tasks. Third, at inference, we introduce a two-pass procedural sampling approach that supports flexible input-target configurations. Together, these design choices create a versatile 3D “virtual camera simulation system” capable of synthesizing novel views along arbitrary camera trajectories with any number of input and target views, without using a 3D representation.

We conducted a unified benchmark across 10 datasets and a variety of experimental settings, including both open-source and proprietary models. Our benchmark reflects the diversity of real-world NVS tasks across the board and systematically evaluates existing methods beyond their comfort zones. We find that SEVA consistently outperforms previous works, achieving +1.5 dB PSNR over the state of the art CAT3D [159] in its own setup. Moreover, our method generalizes well to in-the-wild user captures, with input views ranging from 1 to 32.

In summary, our key contributions with the SEVA model include: (1) a training strategy for jointly modeling large viewpoint changes and temporal smoothness, (2) a two-pass procedural sampling method for smooth video generation along arbitrary long camera trajectories, (3) a comprehensive benchmark that evaluates NVS methods across different datasets and settings, and (4) an open-source release of model weights to support future research.

4.2 Background

We consider the evaluation of an NVS model across three key criteria: (1) generation capacity—the ability to synthesize missing regions for large viewpoint changes; (2) interpolation smoothness—the ability to produce seamless transitions between views; and (3) input flexibility—the ability to handle a variable number of input and target views; We review existing NVS models based on these criteria in Table 4.1, including the types of training data.

4.2.1 Types of NVS tasks

Given M input view images $\mathbf{I}^{\text{inp}} \in \mathbb{R}^{M \times H \times W \times 3}$ of $H \times W$ resolution, along with their corresponding cameras π^{inp} , NVS involves predicting N targets views $\mathbf{I}^{\text{tgt}} \in \mathbb{R}^{N \times H \times W \times 3}$, specified by their respective cameras π^{tgt} . For each camera, we assume we know both intrinsics and extrinsics. Based on the number of input views, we define the “sparse-view regime” as having up to 8 input views, and the “semi-dense-view regime” as an intermediate state bridging the sparse-view regime and dense captures, which typically involve hundreds of views. Based on the nature of their target views, we bucket a broad range of NVS tasks into “set NVS” and “trajectory NVS”, as shown in Figure 4.3.

Set NVS considers a set of target views in arbitrary order, usually across a large spatial range. The order of views is often not helpful here, and a good NVS model requires great generation capacity to excel at this task. We note that some works address only this task (e.g., ReconFusion [158]).

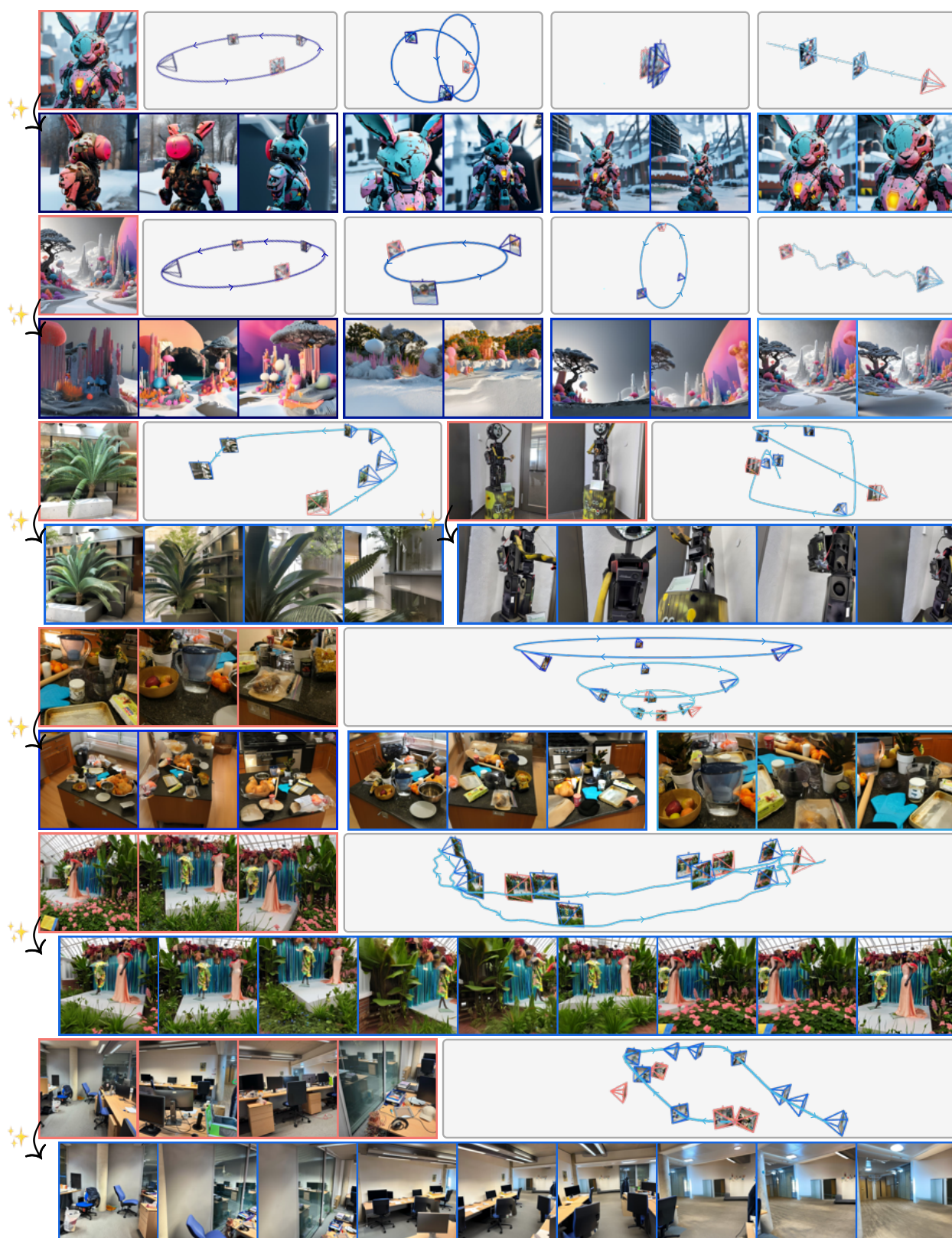


Figure 4.2: **Diverse camera control.** SEVA generates photorealistic novel views following diverse camera trajectories. This includes orbit, spiral, zoom out, dolly zooms, and any user-specified trajectories. Please visit our website for more visual results.






















model	training data	generation capacity	interpolation smoothness	input flexibility
Regression-based				
pixelNeRF [155]		✗	✓	sparse (1)
pixelSplat [166]		✗	✓	sparse (2)
MVSplat [167]		✗	✓	sparse (2)
Long-LRM [168]		✗	✓	semi-dense ({16, 32})
LVSM [162]	 / 	✗	✓	sparse ({2, 4})
Diffusion-based: image models				
Zero123 [157]		✓	✗	sparse (1)
ZeroNVS [164]	 	✓	✗	sparse (1)
ReconFusion [158]	 	✓	✗	sparse (3)
CAT3D [159]	 	✓	✗	sparse ([1, 9])
Diffusion-based: video models				
SV3D [169]		✗	✓	sparse (1)
MotionCtrl [161]		✗	✓	sparse (1)
ViewCrafter [160]		✗	✓	sparse (2)
4DiM [163]		✓	✓	sparse ({1, 2, 8})
SEVA	 	✓	✓	sparse ([1, 8]), semi-dense ([9, 32*])

Table 4.1: **Comparison of existing NVS models** based on the source of training data and key attributes. SEVA is trained on both object-level () and scene-level () data, offering flexible input conditioning, strong generation capacity, and smooth view interpolation. We define generation capacity and interpolation smoothness of each work based on their evaluation setting and our benchmark results. *This upper-bound can be up to hundreds for dense captures, we test our model up to 32 views in practice.

Trajectory NVS regards target views along a smooth camera trajectory, such that they form a video sequence. However, they are often sampled within a small spatial range in a shorter video. To solve this task, a good NVS model requires great interpolation smoothness to produce consistent and non-flickering results. We note that some existing works address only this task (e.g., ViewCrafter [160]).

4.2.2 Existing models

We group existing approaches into regression- and diffusion-based models based on their high-level design choices. A more detailed discussion of related works can be found in Appendix C.3.

Regression-based models learn a deterministic mapping:

$$f_{\theta}(\mathbf{I}^{\text{inp}}, \boldsymbol{\pi}^{\text{inp}}, \boldsymbol{\pi}^{\text{tgt}}),$$



Figure 4.3: **Set NVS versus trajectory NVS.** Set NVS generates target views as an image set, whereas trajectory NVS produces them as a trajectory video.

to directly generate \mathbf{I}^{tgt} deterministically from $\mathbf{I}^{\text{inp}}, \boldsymbol{\pi}^{\text{inp}}, \boldsymbol{\pi}^{\text{tgt}}$. f_θ can be either an end-to-end network parameterized by θ , or a composition of a feed-forward prediction of an intermediate 3D representation and then a neural renderer (e.g., NeRF [170] or 3DGS [123]). For the latter case, set NVS and trajectory NVS are solved in the same way since there exists a persistent 3D representation.

Diffusion-based models capture the conditional distribution:

$$p_\theta(\mathbf{I}^{\text{tgt}} \mid \mathbf{I}^{\text{inp}}, \boldsymbol{\pi}^{\text{inp}}, \boldsymbol{\pi}^{\text{tgt}}),$$

from which \mathbf{I}^{tgt} are sampled [171] iteratively. We highlight two types of models within this scope: Image and Video models. Image models are trained on unordered image sets, such that $(\mathbf{I}^{\text{inp}}, \mathbf{I}^{\text{tgt}}) \sim \mathcal{I}$, where $\mathcal{I} = \{\mathbf{I}_{\sigma(1)}, \mathbf{I}_{\sigma(2)}, \dots, \mathbf{I}_{\sigma(M+N)}\}$ is an image batch, and $\sigma(\cdot)$ is a random permutation function, where camera parameters are omitted for simplicity. Image models usually thrive at set NVS, but struggle in trajectory NVS since they are designed to generate images and not videos. Additionally, the unordered nature of all views solicits flexible input conditioning. Video models are instead trained on ordered views, such that $(\mathbf{I}^{\text{inp}}, \mathbf{I}^{\text{tgt}}) \sim \mathcal{V}$, where $\mathcal{V} = \{\mathbf{I}_1, \mathbf{I}_2, \dots, \mathbf{I}_{M+N}\}$ is a randomly sampled video batch with ordering preserved. Additional temporal operators may also be used to improve the temporal smoothness, such as temporal positional encoding and temporal attention. In contrast with image models, video models thrive at trajectory NVS, but struggle in set NVS. Moreover, all existing video models require both input and target views to be ordered (input views followed by target ones), constraining their input flexibility [172, 169, 161, 173, 174].

4.2.3 Remarks and motivation

Existing tasks pose critical challenges to our design choices. Specifically, our design choices are made to achieve high generation capacity, smooth view interpolation, and flexible input conditioning, as compared in Table 4.1. In this way, we can employ a single model for both tasks, described next.

4.3 Method

We describe our model design and training strategy in Section 4.3.1, then our sampling process at test time in Section 4.3.2 and Section 4.3.3. A system overview is provided in Figure 4.4.

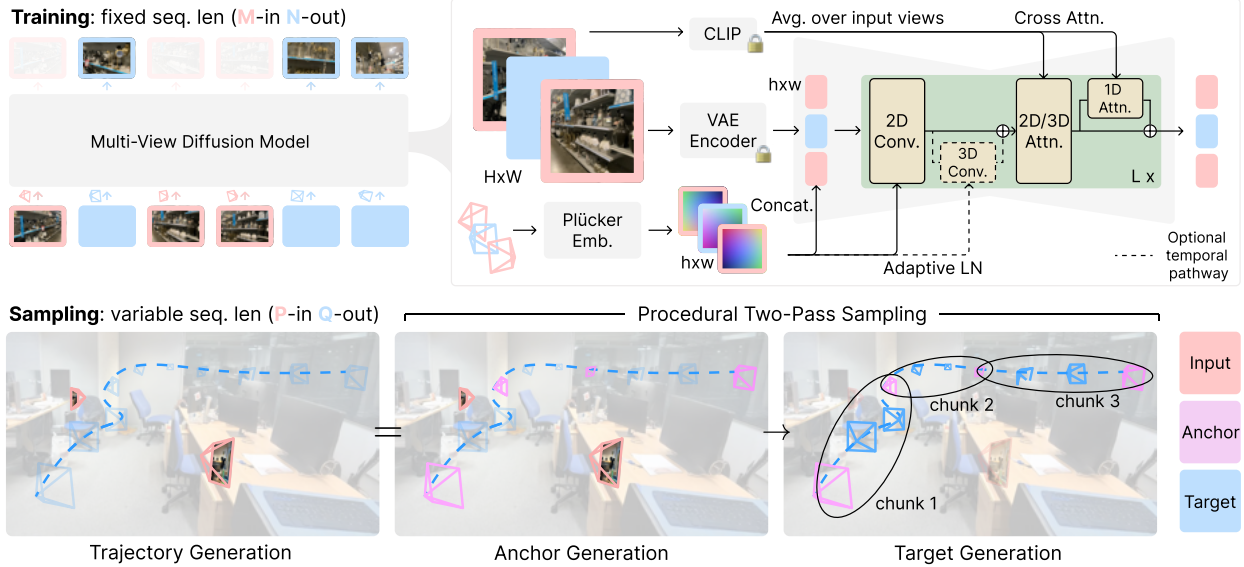


Figure 4.4: **Method.** SEVA is trained with fixed sequence length as a “ M -in N -out” multi-view diffusion model with standard architecture. It conditions on CLIP embeddings, VAE latents of the input views, and their corresponding camera poses. During sampling, SEVA can be cast as a generative “ P -in Q -out” renderer that works with variable sequence length, where P and Q need not be equal to M and N . To enhance temporal and 3D consistency across generated views, especially when generating along a trajectory, we present procedural two-pass sampling as a general strategy.

4.3.1 Model design and training

We consider a “ M -in N -out” multi-view diffusion model p_θ , as notated in Section 4.2.2. We formulate this learning problem as a standard diffusion process [171] without any change.

Architecture. Our model is based on the publicly available SD 2.1 [175], which consists of an auto-encoder and a latent denoising U-Net. Following [159], we inflate the 2D self-attention of each low-resolution residual block into 3D self-attention [176] within the U-Net. To improve model capacity, we add 1D self-attention along the view axis after each self-attention block via skip connection [177, 178], bumping the model parameters from 870M to 1.3B. Optionally, we further tame this model into a video model by introducing 3D convolutions after each residual block via skip connection, similar to [179, 172], yielding 1.5B total parameters. The temporal pathway can be enabled during inference when frames within one forward pass are known to be spatially ordered, enhancing output’s smoothness.

Conditioning. To fine-tune our base model into a multi-view diffusion model, we add camera conditioning as Plücker embedding [180] via concatenation [159] and adaptive layer normalization [181]. We normalize π^{inp} and π^{tgt} by first computing the relative pose with respect to the first input camera and then normalizing the scene scale such that all camera positions are within

a $[-2, 2]^3$ cube. For each input frame, we first encode its latent then concatenate with its Plücker embedding and a binary mask [159, 172] differentiating between input and target views. For each target frame, we use the noisy state of its latent instead. Additionally, we find it helpful [169] to also inject high-level semantic information via CLIP [182] image embedding. We zero initialize new weights for additional channels in the first layer. In our experiment, we found that our model can quickly adapt to these conditioning changes and produce realistic images with as few as 5K iterations.

Training. Let us define the training context window length $T = |\mathbf{I}^{\text{inp}}| + |\mathbf{I}^{\text{tgt}}| = M + N$. One natural goal is to support large T such that we can generate a larger set of frames. However, we find that naive training is prone to divergence, and we thus employ a two-stage training curriculum. During the first stage, we train our model with $T = 8$ with a batch size of 1472 for 100K iterations. In the second stage, we train our model with $T = 21$ with a batch size of 512 for 600K iterations. Given a training video sequence, we randomly sample the number of input frames $M \in [1, T - 1]$ and the frames $(\mathbf{I}^{\text{inp}}, \mathbf{I}^{\text{tgt}})$. We find it important to jointly sample \mathbf{I} with a smaller subsampling stride to ensure sufficient temporal granularity and avoid missing critical transitions with a small probability (0.2 is used in practice). In the optional video training stage, we only train temporal weights with data sampled with a small subsampling stride and a batch size of 512 for 200K iterations. We shift the signal-to-noise ratio (SNR) in all stages as more noise is necessary to destroy the information when training with more frames, corroborating findings from [183, 184, 159]. The model is trained with squared images with $H = W = 576$.

4.3.2 Sampling novel views

Once the diffusion model is trained, we can sample it for a wide range of NVS tasks during test time. Formally, let us consider a “ P -in Q -out” NVS task during testing, where we are given $P = |\mathbf{I}^{\text{inp}}|$ input frames and aim to produce $Q = |\mathbf{I}^{\text{tgt}}|$ target frames. Our goal is to design a generic sampling strategy that works for all P and Q configurations, where P and Q need not be equal to M and N .

We make two key observations: First, within a single forward pass, predictions are 3D consistent, provided the model is well-trained. Second, when $P + Q > T$, \mathbf{I}^{tgt} must be split into smaller chunks of size Q_i such that $P + Q_i \leq T$ for the i^{th} forward pass. We term this practice *one-pass sampling*. However, predictions across these forward passes would be inconsistent unless they share common frames to maintain local consistency within a spatial neighborhood. Building on these observations, we summarize our sampling process under two scenarios: $P + Q \leq T$ and $P + Q > T$.

$P + Q \leq T$. We fit the task within one forward pass for simplicity and consistency. As shown in Appendix C.5, we find it works better to pad the forward pass to have exactly T frames by repeating the first input image, compared to changing the context window T zero-shot.

$P + Q > T$. We propose *procedural two-pass sampling*: In the first pass, we generate anchor frames \mathbf{I}^{acr} using all input frames \mathbf{I}^{inp} . In the second pass, we divide \mathbf{I}^{tgt} into chunks and generate them using \mathbf{I}^{acr} (and optionally \mathbf{I}^{inp}) according to the spatial distribution of \mathbf{I}^{acr} and \mathbf{I}^{tgt} . Given the

distinct nature of the two tasks of interest—set NVS and trajectory NVS—e.g., differences in the availability of views’ ordering, we design tailored chunking strategies for each task.

For set NVS, we consider nearest procedural sampling. We first generate \mathbf{I}^{acr} based on pre-defined trajectory priors, similar to [159], e.g., 360 trajectories for object-centric scenes, or spiral trajectories for forward-facing scenes. We then divide \mathbf{I}^{tgt} into chunks w.r.t. \mathbf{I}^{acr} using nearest neighbor. Specifically, the i^{th} forward pass involves:

$$\text{nearest} : \{\mathbf{I}_i^{\text{acr}}\} \cup \{\mathbf{I}_j^{\text{tgt}} \mid \text{NN}(\mathbf{I}_j^{\text{tgt}}, \mathbf{I}^{\text{acr}}) = \mathbf{I}_i^{\text{acr}}\}.$$

We considered two strategies of procedural sampling: nearest as described above, and gt + nearest strategy by appending \mathbf{I}^{inp} into each forward pass. We find that the gt + nearest strategy performs better than nearest and thus default to it instead. In the absence of trajectory priors, we revert to one-pass sampling. In practice, employing nearest anchors enhances qualitative consistency, albeit on a limited scale.

For trajectory NVS, we consider interp procedural sampling. We first generate a subset of target frames as \mathbf{I}^{acr} by uniformly spanning the target camera path with a stride $\Delta = \lfloor \frac{Q}{T-2} \rfloor$. We then generate the rest of \mathbf{I}^{tgt} as segments between those anchors:

$$\text{interp} : \{\mathbf{I}_i^{\text{acr}}, \mathbf{I}_{i \cdot \Delta + 1}^{\text{tgt}}, \dots, \mathbf{I}_{(i+1) \cdot \Delta - 1}^{\text{tgt}}, \mathbf{I}_{i+1}^{\text{acr}}\}.$$

Since the input to the model is ordered, we can leverage temporal weights to further improve smoothness (Section 4.4.3). Similarly, gt + interp is possible by appending \mathbf{I}^{inp} with $\Delta = \lfloor \frac{Q}{T-P-2} \rfloor$. We find that interp is sufficiently robust, and choose it as the default option. The interp strategy drastically outperforms its counterparts (e.g., one-pass, or gt + nearest procedural sampling) in terms of temporal smoothness.

4.3.3 Scaling sampling for large P and Q

Next, we examine two special cases when $P + Q > T$: $P > T$ and $Q \gg T$. Here, we make a tailored design for anchor generation in the first pass, while keeping target generation in the second pass unchanged.

$P > T$. In the semi-dense-view regime (e.g., $P = 32$), we extend the context window length T zero-shot to accommodate all P input views and anchor views in one pass during anchor generation. Empirically, T can even be extended up to hundreds without severe degradation in photorealism in the generated outputs. We find that the diffusion model generalizes well in this case as long as the input views cover the majority of the scene, shifting the task from generation to primarily interpolation. In the sparse-view regime (i.e., $P \leq 8$), we observe similar performance degradation caused by zero-shot extension of T compared to what we have found when $P + Q \leq T$. Refer to Section 4.4.5 for a detailed discussion.

$Q \gg T$. When the number of target views Q is large, e.g., in large-set NVS or long-trajectory NVS, even anchors will be chunked into different forwards in the first pass, leading to the inconsistency of anchors. To this end, we maintain a *memory bank* of anchor views previously generated,

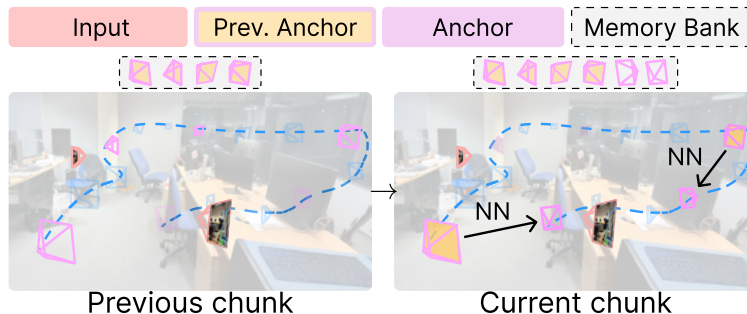


Figure 4.5: **Anchor generation when $Q \gg T$.** We introduce a memory bank composed of previously-generated anchor views and their corresponding camera poses. The lookup of spatial neighbors helps improve long-term 3D consistency.

as shown in Figure 4.5. We generate anchors auto-regressively by retrieving their spatially nearest ones from the memory bank, similar to the nearest strategy introduced above for the second pass. In Section 4.4.4, we show that this strategy drastically outperforms the standard practice of reusing temporally nearest anchors previously generated in long video literature [174], in terms of long-range 3D consistency, especially for hard trajectories.

4.4 Experiments

We employ a single model for a spectrum of settings and find that SEVA model generalizes well under the three criteria (Table 4.1). We cover different NVS tasks (set NVS and trajectory NVS) and examine one special task of interest—long trajectory NVS. We also cover different input regimes (single-view, sparse-view, and semi-dense-view). A discussion about several key properties is presented in Section 4.4.5.

4.4.1 Benchmark

Datasets, splits, and the number of input views. We consider (1) object datasets, e.g., OmniObject3D [185] (OO3D) and GSO [186]; (2) object-centric scene datasets, e.g., LLFF [187], DTU [188], CO3D [189], and WildRGBD [190] (WRGBD); and (3) scene datasets, e.g., RealEstate10K [191] (RE10K), Mip-NeRF 360 [49] (Mip360), DL3DV140 [192] (DL3DV), and Tanks and Temples [193] (T&T). We consider a wide range of the number of input views P , ranging from sparse-view regime to semi-dense-view regime, evaluating models’ input flexibility. To establish a comprehensive and rigorous comparison with baselines, we consider different dataset splits utilized in prior works with the same input-view configuration, unless specified as our split (O). These include splits used in 4DiM [163] (D), ViewCrafter [160] (V), pixelSplat [166] (P), ReconFusion [158] (R), SV3D [169] (S), and Long-LRM [168] (L). For example, the 4DiM [163] (D) split on the RE10K dataset is 128 out of all 6711 test scenes with $P = 1$.

Method	dataset	OO3D GSO		RE10K				LLFF		DTU		CO3D		WRGBD		Mip360	DL3DV		T&T		
	split	O	O	D [163]	V [160]	P [166]	R [158]	R [158]	R [158]	R [158]	V [160]	R [158]	O _e	O _h	R [158]	O	L [168]	V [160]	L [168]		
	<i>P</i>	3	3	1	1	2	1	3	1	3	1	3	1	3	3	6	6	6	32	1	32
Regression-based models																					
Long-LRM [168]		-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	23.86	-	18.20	
MVSplat [167]		14.78	15.21	20.42	20.32	26.39	21.56	25.64	11.23	12.50	13.87	15.52	12.52	13.52	14.56	12.54	13.56	14.34	16.24	13.22	12.63
DepthSplat [194]		15.67	16.52	20.90	19.24	27.44	21.87	22.54	12.07	12.62	14.15	16.24	13.23	13.77	15.93	14.23	14.01	15.72	16.78	14.35	13.12
LVSM [162]		-	-	-	-	29.67	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
Diffusion-based models																					
MotionCtrl [195]		-	-	12.74	16.29	-	-	-	-	-	-	15.46	-	-	-	-	-	-	13.29	-	
4DiM [163]		-	-	17.08	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
ViewCrafter [160]		14.64	15.93	20.43	22.04	21.42	20.88	22.81	10.53	13.52	12.66	16.40	18.96	14.72	16.42	12.66	14.59	13.78	-	18.07	
SEVA		30.30	31.53	17.99	18.56	25.66	18.11	27.57	14.03	19.48	14.47	20.82	18.40	19.25	19.75	18.91	16.70	17.80	20.96	15.16	16.50

Table 4.2: **PSNR \uparrow on small-viewpoint set NVS.** *P* denotes the number of input views. For all results with *P* = 1, we sweep the unit length for camera normalization due to the model’s scale ambiguity. *O_e* and *O_h* denote the easy and hard split of our split, respectively. Underlined numbers are run by us using the official released code.

Small-viewpoint versus large-viewpoint NVS. Sweeping across all datasets, splits, and input-view configurations reveals a diverse benchmark of setups. To better evaluate models’ generation capacity and interpolation smoothness (Section 4.2.1), we propose to categorize these setups into two groups—*small-viewpoint NVS* and *large-viewpoint NVS*—depending on the disparity between \mathbf{I}^{gt} and \mathbf{I}^{inp} . small-viewpoint NVS with smaller disparities emphasizes interpolation smoothness and continuity with nearby input views, whereas large-viewpoint NVS with larger disparities requires a model to generate prominent unseen areas from input observations, predominantly assessing models’ generation capacity. See Table C.1 for the complete list. Refer to Appendix C.4 for the detailed choice of datasets, splits, the number of input views, and the way to measure disparity.

Baselines. We consider a range of proprietary models, including ReconFusion [158], CAT3D [159], 4DiM [163], LVSM [162], and Long-LRM [168]. We also consider various open-source models, including SV3D [169], MVSplat [167], depthSplat [194], MotionCtrl [195], and ViewCrafter [160]. As outlined in Section 4.2.2, these baselines encompass both regression-based and diffusion-based approaches, providing a comprehensive framework for comparison.

4.4.2 Set NVS

In this section, we focus on comparing our model against prior works, given that set NVS is a task that has been extensively explored.

Quantitative comparison. The input and target views are chosen following splits used in previous methods. The order of target views is not preserved, i.e., $\mathbf{I}^{\text{gt}} \sim \mathcal{I}$. We use standard metrics of peak signal-to-noise ratio (PSNR), learned perceptual image patch similarity [197] (LPIPS), and structural similarity index measure [9] (SSIM). Only PSNR is showcased here due to space limits, with the rest deferred to Appendix C.5.2. Empirically, our method shows a greater performance improvement on LPIPS, reflecting the photorealism of our results.

Method	dataset	OO3D	GSO	CO3D	WRGBD	Mip360	DL3DV	T&T						
	split	S [169]	S [169]	R [158]	O _h	R [158]	O	O						
	P	1	1	1	1	3	1	3	1	3	6	9		
SV3D [169]		19.28	20.38	-	-	-	-	-	-	-	-	-		
DepthSplat [194]		11.56	12.32	10.42	9.35	13.53	10.49	12.54	9.63	12.52	8.63	9.78	10.12	11.20
CAT3D [159]		-	-	-	-	-	-	15.15	-	-	-	-	-	-
ViewCrafter [160]		10.56	11.42	10.11	9.12	13.45	9.79	10.34	8.97	11.50	9.23	9.88	10.32	11.08
SEVA		19.25	20.65	15.30	14.37	17.28	12.93	15.78	13.01	15.95	11.28	12.65	13.80	14.72

Table 4.3: **PSNR \uparrow on large-viewpoint set NVS.** For all results with $P = 1$, we sweep the unit length for camera normalization due to the model’s scale ambiguity. Underlined numbers are run by us using the officially released code.

Method	small-viewpoint				large-viewpoint
	RE10K	LLFF	DTU	CO3D	Mip360
ZipNeRF [196]	20.77	17.23	9.18	14.34	12.77
ZeroNVS [164]	19.11	15.91	16.71	17.13	14.44
ReconFusion [158]	25.84	21.34	20.74	19.59	15.50
CAT3D [159]	26.78	21.58	22.02	20.57	16.62
SEVA	27.95	21.88	22.68	21.88	17.82

Table 4.4: **PSNR \uparrow on 3DGS renderings for set NVS.** Results are reported on the ReconFusion [158] split with $P = 3$.

For small-viewpoint set NVS, Table 4.2 shows that SEVA sets state-of-the-art results in the majority of splits. In the sparse-view regime (i.e., $P \leq 8$), SEVA excels across different datasets when $P > 1$. For example, a performance gain of +6.0 dB PSNR is achieved on LLFF with $P = 3$. In the semi-dense-view regime (e.g., $P = 32$), SEVA surprisingly performs favorably against the specialized model [168], despite not being specifically designed for this setup. For example, SEVA lags behind the state-of-the-art method [168] by only 1.7 dB on T&T. On object datasets OO3D and GSO, SEVA achieves a significantly higher state-of-the-art PSNR compared to all other methods.

Notably, for small-viewpoint set NVS on the RealEstate10K [191] dataset, SEVA underperforms when in the single-view regime (i.e., $P = 1$). This issue arises from scale ambiguity in the model due to two factors: (1) it always takes in unit-normalized cameras during training, and (2) it is trained on multiple datasets with diverse scales. This challenge is most pronounced on RE10K, where panning motion dominates. Additionally, the absence of a second input view negates any scale relativity. To address this, for all results with $P = 1$, we sweep the *unit length for camera normalization* from 0.1 to 2.0 (with 2.0 used during training), selecting the best scale for each scene. On P split with $P = 2$, we observe diffusion models lag behind regression-based models that are advantageous in small-viewpoint interpolation. SEVA bridges this gap by improving upon the state-of-the-art diffusion-based model by +4.2 dB. On R split with $P = 3$, the advantage of SEVA is pronounced exceeding the previously best result by +1.9 dB. Notably, ViewCrafter excels on V split due to capacity taking in wide-aspect-ratio images and thus more input pixels than others with square images. The advantage of ViewCrafter on V split diminishes on CO3D since the majority of informative pixels are centrally located.

For large-viewpoint set NVS, Table 4.3 shows that SEVA’s quantitative advantages are even more prominent here, revealing clear benefits of SEVA in terms of generation capacity when the camera spans a large spatial range. On Mip360 with $P = 3$, SEVA improves over previous state-of-the-art method CAT3D [159] by +0.6 dB PSNR. On harder scenes like DL3DV and T&T with different input-view configurations, SEVA obtains a clear performance lead. On OO3D and GSO with $P = 1$, although the performances of SEVA and previous state-of-the-art method [169] are similar, we qualitatively observe more photorealistic and sharper output from our model.



Figure 4.6: **SOTA comparison** on set NVS (top) and trajectory NVS (bottom) across varying numbers of input views. We compare with open-source approaches—ViewCrafter [160] (VC) and DepthSplat [194] (DS)—as well as proprietary ones including LVSM [162], Long-LRM [168] (LLRM), 4DiM [163], and CAT3D [159]. When the input comprises multiple views, we arrange them so that the view closest to the target is placed at the top of each set.

Qualitative comparison. Figure 4.6 top panel shows a qualitative comparison with diverse baselines. For small-viewpoint set NVS, the output from SEVA with the best scale exhibits desirable alignment with the ground truth while being more photorealistic in details. Compared with LVSM [162] on the P split of RE10K, SEVA produces sharper images, also corroborating that lower PSNR arises from scale ambiguity rather than interpolation quality. Similar trends hold when

compared to Long-LRM [168] on DL3DV with $P = 32$. For large-viewpoint set NVS, we compare with DepthSplat [194] on DL3DV with $P = 3$. DepthSplat fails to produce reasonable results when the viewpoint change is too large and falls short in overall visual quality.

Comparison of 3D reconstruction. To enable a direct quantitative comparison with prior works [158, 159], we adopt the few-view 3D reconstruction pipeline described in [159]. For each scene, we first generate 8 videos conditioned on the same input views following different camera paths, summing into 720 generated views. Then, both the input views and generated views are distilled into a 3DGS-MCMC [198] representation without point cloud initialization. We optimize the camera parameters and apply LPIPS loss [10] during the distillation. Finally, we render the distilled 3D model on the test views and report the performance in Table 4.4. SEVA shows a consistent performance lead.

4.4.3 Trajectory NVS

In this section, we focus on qualitative demonstration, given that trajectory NVS is an underexplored task. We then compare against prior arts both qualitatively and quantitatively.

Qualitative results. Figure 4.2 presents qualitative results, illustrating trajectories of varying complexities with different numbers of input views across diverse types, including object-centric scene-level, scene-level, real-world, and text-prompted from image diffusion models [175], etc..

In the single-view regime (i.e., $P = 1$), we manually craft a set of common camera movements/effects, e.g., look-at 360, spiral, panning, zoom-in, zoom-out, dolly zoom, etc.. We observe that SEVA generalizes to a wide range of images and demonstrates accurate camera-following capacity. Excitingly, our model derives reasonable output with a dolly zoom effect (the second row of Figure 4.2). In the FERN scene from the third row of Figure 4.2, our model demonstrates its ability to generate plausible outputs even when moving close to or passing through an object—despite never being explicitly trained for such scenarios. This highlights the expressiveness of our model. An extensive sweeping of camera movements on 4 types of images is provided in Figure C.3, Figure C.4, Figure C.5, and Figure C.6.

In the sparse-view regime with few input views (i.e., $1 < P \leq 8$), we observe that SEVA demonstrates strong generalization to in-the-wild real-world images and versatility in adapting to different numbers of input views. The output forms a smooth trajectory video with subtle temporal flickering, revealing its capacity to interpolate between views smoothly. In the last row of Figure 4.2, our model generates plausible results at the end of the trajectory—an area unseen in the input observations—demonstrating its strong generation capacity. In the semi-dense-view regime (i.e., $P > 9$), we similarly find that SEVA is surprisingly able to produce a smooth trajectory video with minimal artifacts. Please check the website for video results.

Qualitative comparison. Figure 4.6 bottom panel presents a qualitative comparison with diverse baselines. In the single-view regime (i.e., $P = 1$), we compare to 4DiM [163] and CAT3D [159]. We observe more photo-realistic and sharper output from our model, especially in the background area for object-centric scenes. 4DiM outputs tend to be cartoonish and over-simplistic, given that

Method	split	small-viewpoint					large-viewpoint		
	dataset	V [160]			O				
		RE	CO3D	T&T	RE	DTU	WR	DL	T&T
MotionCtrl [195]		16.29	15.46	13.29	-	-	-	-	-
DepthSplat [194]		19.24	13.23	14.35	25.23	14.68	12.45	11.32	9.11
ViewCrafter [160]		22.04	18.96	18.07	26.54	18.99	13.44	11.45	9.68
SEVA		18.56	18.40	15.16	27.34	19.99	17.79	15.76	11.92
SEVA (+ temp.)		18.62	18.43	15.13	27.36	20.19	17.93	15.78	11.99

Table 4.5: **PSNR \uparrow on trajectory NVS.** *temp.* denotes optional temporal pathway. RE, WR, and DL denotes RE10K, WRGBD, and DL3DV, respectively. For the V [160] split, $P = 1$ with unit length swept; for the O split, $P = 3$. Underlined numbers are run by us using the officially released code.

Method	samples		3DGS	video
	PSNR \uparrow	TSED \downarrow	PSNR \uparrow	MS \uparrow
SEVA (one-pass)	15.73	115.1	16.03	95.39
SEVA (two-pass: nearest)	13.74	120.9	14.21	94.71
SEVA (two-pass: gt + nearest)	15.58	116.2	15.96	95.22
SEVA (two-pass: gt + interp)	15.66	120.1	15.98	<u>95.56</u>
SEVA	15.76	116.7	16.11	95.76
SEVA (+ temp.)	15.78	109.0	16.17	95.77

Table 4.6: **3D consistency (TSED \downarrow and PSNR \uparrow) and temporal quality (MS \uparrow) on trajectory NVS.** SEVA uses interp procedural sampling by default. *temp.* denotes the optional temporal pathway. *MS* denotes motion smoothness from VBench [199]. Results are reported on our split of DL3DV with $P = 3$.

the model is only trained on RE10K. In the sparse-view regime with few input views (i.e., $P = 3$), we compare with CAT3D and observe that our model demonstrates more photo-realistic textures, especially in the background. For start-end-view interpolation considered in ViewCrafter [160] with $P = 2$, our model produces smooth transitions across trajectories, although it exhibits slight flickering between adjacent frames, particularly in regions with significant high-frequency detail.

Quantitative comparison. We use the same input views as in the set NVS for each split. We use all frames from each scene as target views such that they form a smoothly transitioning trajectory video, i.e., $\mathbf{I}^{\text{tgt}} \sim \mathcal{V}$. We use PSNR as metrics and compare with baselines in Table 4.5.

For small-viewpoint trajectory NVS, Table 4.5 compares SEVA with baselines on PSNR. SEVA performs favorably against other methods in V split with $P = 1$. The performance lead of ViewCrafter is mainly attributed to its training on high-resolution images. For large-viewpoint trajectory NVS with $P = 3$, SEVA consistency sets new state-of-the-art results. Applying the

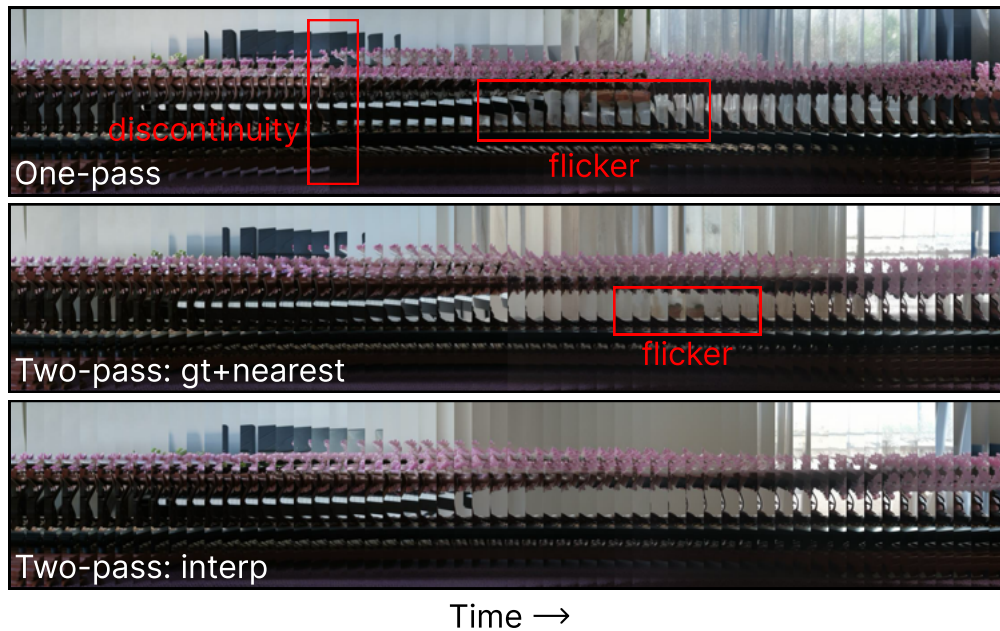


Figure 4.7: **Temporal quality.** Vertical slices of a rendered novel camera path on the BONSAI scene from Mip-NeRF360 [49] illustrate the temporal quality across adjacent viewpoints. One-pass or gt + nearest procedural sampling results in notable flickering, whereas interp procedural sampling ensures temporally smooth rendering.

temporal pathway further boosts performance and improves smoothness, indicating the benefits of the gated architecture.

Ablation on two-pass procedural sampling. We conduct an ablation study comparing the default interp procedural sampling with one-pass sampling and alternative procedural sampling strategies.

Quantitatively, beyond evaluating PSNR on individual views, we assess 3D consistency using the PSNR on 3D renderings of that same view and SED [163, 200] score. To compute the SED score, we first apply SIFT [201] to detect keypoints in two images. For each keypoint in the first image, we determine its corresponding epipolar line in the second image and measure the shortest distance to its match. Additionally, we report Motion Smoothness (MS) from VBench [199], a benchmark designed to evaluate temporal coherence in video generative models. As shown in Table 4.6, interp procedural sampling demonstrates a clear advantage over its alternatives, with the integration of the temporal pathway further reinforcing its superiority.

Qualitative comparisons in Figure 4.7 show that one-pass sampling introduces visible temporal flickering and abrupt visual changes. In contrast, interp produces the smoothest transitions, outperforming gt + nearest and mitigating noticeable flickering.

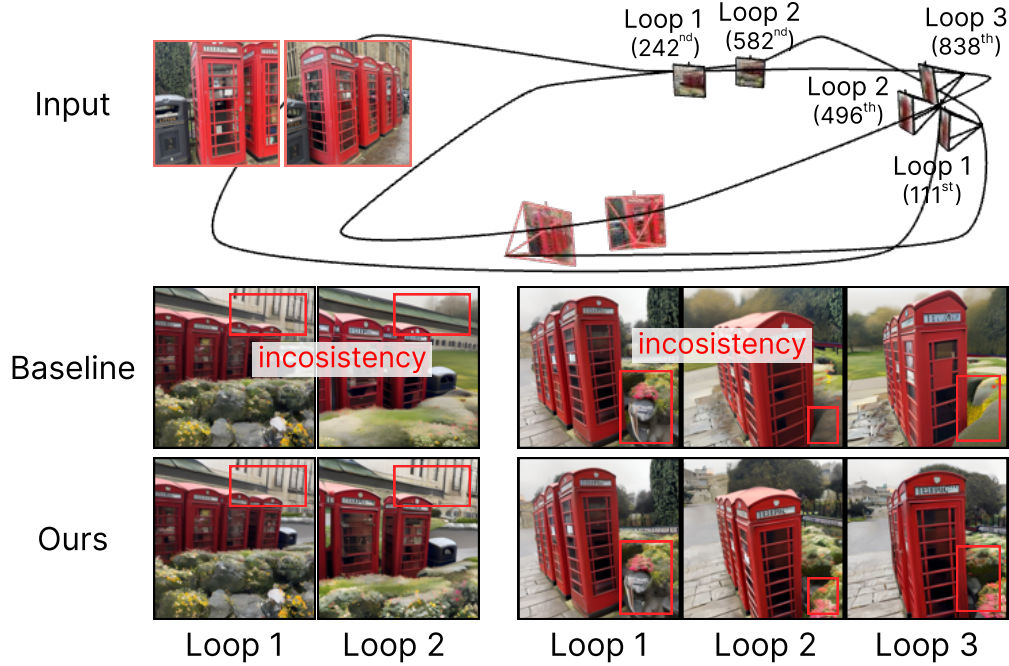


Figure 4.8: **Long-range 3D consistency.** We visualize samples following a camera path looping three times around the TELEPHONE-BOOTH scene. Lookup using spatial neighbors from the memory bank (ours) notably improves view consistency and reduces artifacts in recurring locations across different loops, compared to lookup using temporal neighbors (baseline).

4.4.4 Long-trajectory NVS

Figure 4.8 presents a qualitative demonstration of NVS over a long trajectory of up to 1000 frames. As the camera orbits the TELEPHONE BOOTH for multiple rounds, the generated views in each round from similar viewpoints can be drastically different since they are far away from each other temporally. With the memory bank maintaining previously generated anchors, SEVA achieves robust 3D consistency for long-trajectory NVS, e.g., the building in front of and the plantation after the booth. Comparing it to using temporal nearest anchors previously generated, using spatially nearest ones demonstrates a clear advantage. The memory mechanism has been concurrently explored in previous works [160, 202], leveraging explicit intermediate 3D representations such as dense point clouds predicted by DUST3R [203]. In contrast, our model demonstrates greater robustness and generalizability to in-the-wild data, as it is not constrained by the quality of DUST3R’s output, which often becomes unreliable in quality for data outside of its training domain, e.g., text-prompted images.

4.4.5 Discussions

Zero-shot generalization of context window length T . We surprisingly find our model, though only trained on $T = 21$ frames, can generalize reasonably to larger T during sampling *in the*

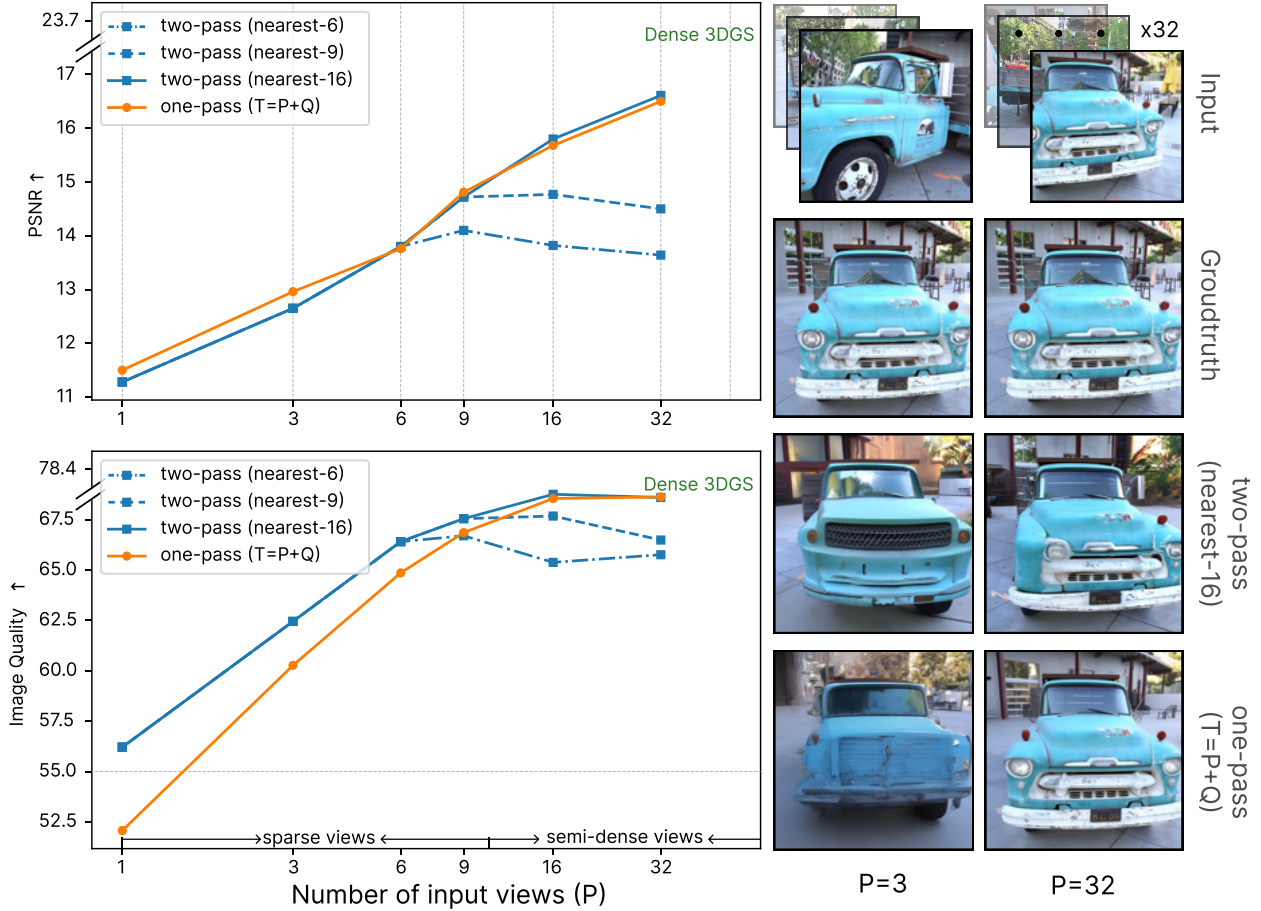


Figure 4.9: **Generation quality on the number of input views.** PSNR↑ (top) and Image Quality↑ (bottom) on set NVS. Results are reported on our split of T&T. Extending T to more input views in a zero-shot manner produces more consistent samples in the semi-dense-view regime. Dense 3DGS denotes results of [123] with full views.

semi-dense-view regime. On our split of T&T for set NVS, we evaluate the predictions against ground truth in both sparse-view (i.e., $1 \leq P \leq 8$) and semi-dense-view regime (i.e., $9 \leq P$) using PSNR↑ and Image Quality↑ [199]. Image Quality refers to the distortion (e.g., over-exposure, noise, blur) presented in the generated image. We experiment with different sampling strategies: one-pass sampling zero-shot extending the context window length T ; two-pass procedural sampling by first generating anchor views using nearest- K ($K < T$) input views and then interpolating anchor views into target views.

Our results are shown in Figure 4.9. Procedural sampling with the nearest- K anchor views plateau after taking K views as input, indicating inefficiencies in procedural sampling and an inability to effectively utilize all available input views when $P > T$. Conversely, the metrics steadily improve with respect to the number of input frames for one-pass sampling with T extending to $P + Q$ in a zero-shot manner. However, we observe that this generalization fails in the sparse-

view regime, resulting in blurry samples, as indicated by the low Image Quality when $P < 9$ and qualitative samples when $P = 3$. In the semi-dense-view setting, although quantitative metrics show minimal differences between one-pass and procedural sampling, we consistently observe that one-pass produces more 3D-consistent samples, as illustrated in the bottom-right figure.

Zero-shot generalization of image resolution. Surprisingly, we find our model, despite being trained only on square images with $H = W = 576$, generalizes well to different image resolution during sampling, similar to [204]. As shown in Figure 4.10, SEVA can produce high-quality results in both portrait (16 : 9) and landscape (9 : 16) orientations of different image resolutions.

Guidance scale on generation uncertainty. We employ classifier-free guidance [205] (CFG) to enhance sampling quality. Empirically, we find that the CFG scale, a hyperparameter at test time, has a significant impact on the final result [172], as shown in Figure 4.11. Specifically, the optimal CFG scale is strongly correlated with the inherent uncertainty of the generation. When uncertainty is high (top row), a higher CFG scale (e.g., 5) is preferable to prevent excessive blurriness in the generated samples. Conversely, when uncertainty is low (bottom row), a lower CFG scale (e.g., 3) helps avoid oversaturation. In practice, setting the CFG scale between 2 and 5 consistently produces high-quality results across all our samples.

Sampling diversity of unseen areas. Figure 4.12 demonstrates the capability of the model to generate diverse and plausible predictions for unseen regions of input observations. In the first row, the input view depicts a frontal view of a classical statue. We sample multiple back views by varying the random seeds, producing distinct yet coherent interpretations of the unseen geometry and texture while preserving fidelity to the input. Similarly, in the second row, the model generates multiple plausible continuations of the scene given an input view of a scenic road, each reflecting unique variations in environmental and structural details. These results highlight the model’s ability to synthesize realistic and diverse outputs for occluded or ambiguous regions.

4.5 Conclusion

We present STABLE VIRTUAL CAMERA (SEVA), a generalist diffusion model for novel view synthesis that balances large viewpoint changes and smooth interpolation while supporting flexible input and target configurations. By designing a diffusion-based architecture without 3D representation, a structured training strategy, and a two-pass procedural sampling approach, SEVA achieves 3D consistent rendering across diverse NVS tasks. Extensive benchmarking demonstrates its superiority over existing methods, with strong generalization to real-world scenes. For broader impact and limitations, please refer to the appendix.

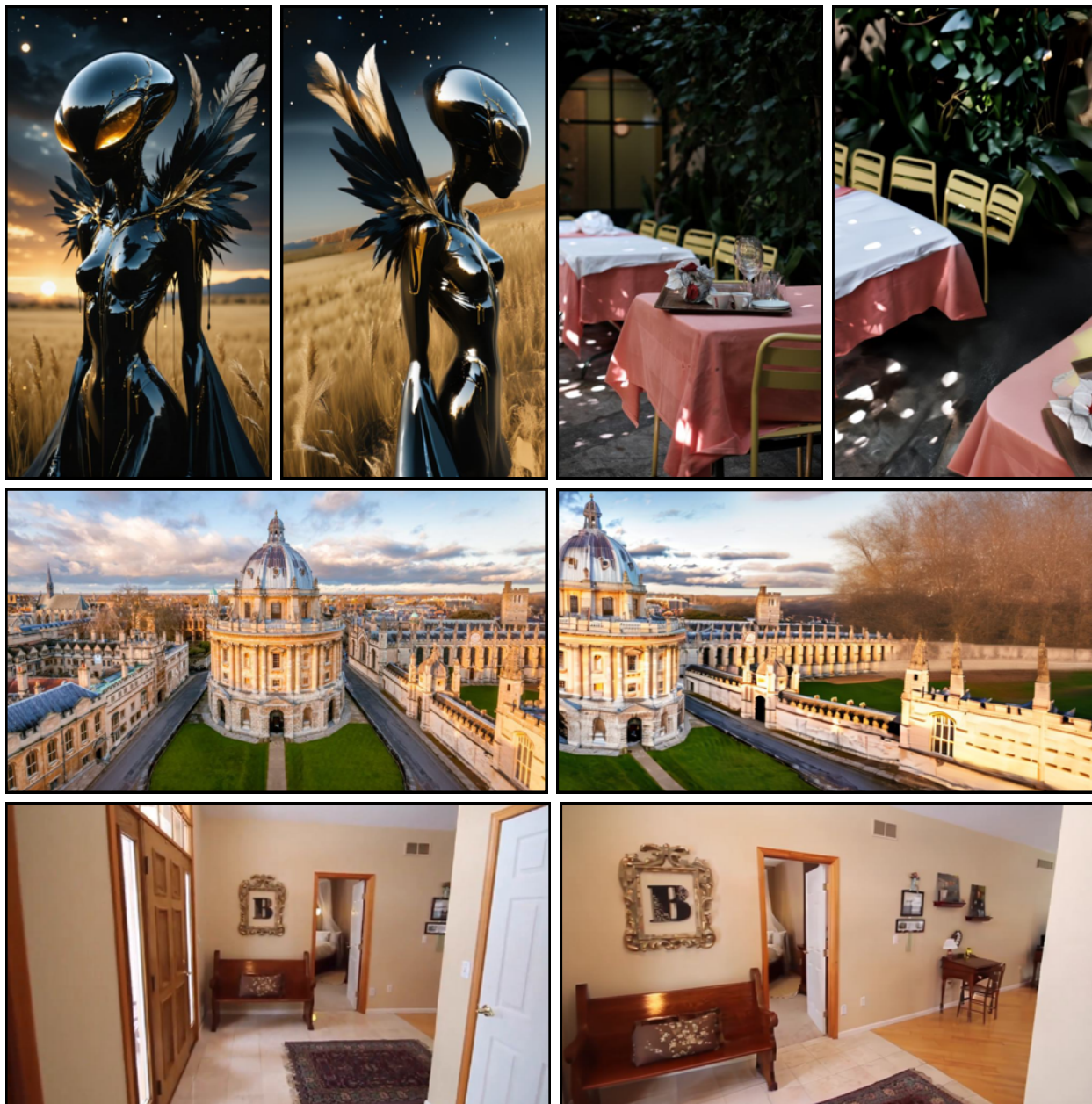


Figure 4.10: **Generation quality on different image resolutions.** Our model generalizes to different image resolution of varying aspect ratios, including both portrait (top) and landscape orientations (bottom). Results are presented as a pair of the input view and the target views.

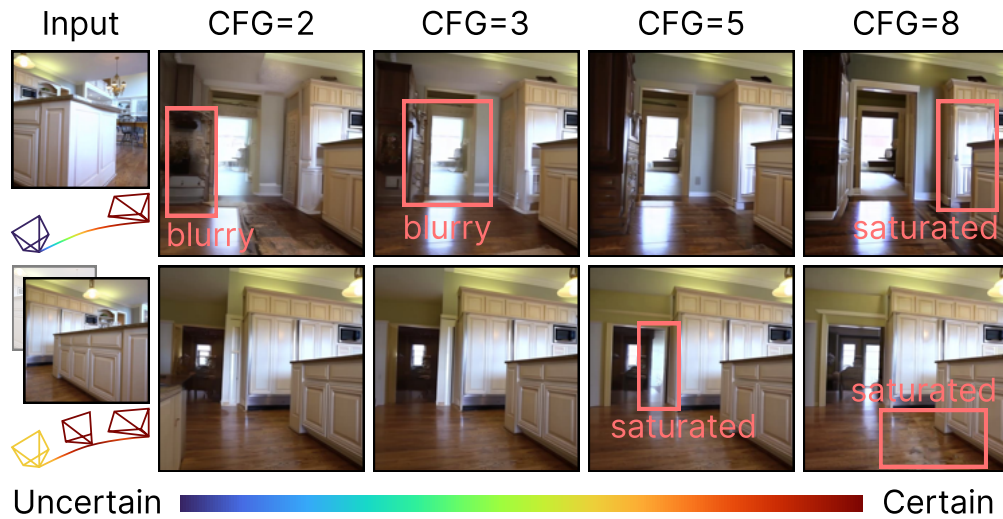


Figure 4.11: **Generation uncertainty on CFG.** The CFG scale should be increased as generation uncertainty rises. For single-view conditioning (top), a higher CFG scale is typically required, whereas few-view conditioning (bottom) benefits from a lower scale.



Figure 4.12: **Generation diversity in unseen regions.** Our model generates diverse samples by varying randomization seeds during the sampling process.

Chapter 5

Conclusion

In this thesis, we chart a data-driven path for scaling view synthesis. Chapter 2 diagnoses the hidden multi-view assumptions in state-of-the-art dynamic NeRF pipelines and establishes a benchmark that exposes their limits under monocular capture. Chapter 3 remedies these limits with a hybrid system that fuses monocular depth and long-range tracks into a dynamic 3D Gaussian scene representation, recovering globally consistent geometry and motion from a single video. Chapter 4 completes the progression by reframing view synthesis as camera-conditioned diffusion: a fully generative model that synthesizes photorealistic, 3D-consistent images and minute-long videos from sparse inputs while inheriting the scalability of large 2D backbones.

5.1 Limitations and future work

While this thesis outlines a practical path for scaling view synthesis, several limitations remain.

First, our diagnostic benchmark in Chapter 2 only accounts for view sparsity; other factors such as lighting, texture, sequence length, and global motion remain unquantified. Systematically isolating and evaluating these variables would provide a more complete understanding of the problem space.

Second, the 4D reconstruction system introduced in Chapter 3 remains optimization-based. This makes it slow to train and dependent on a heavy preprocessing pipeline. Developing a feedforward or transformer-based 4D reconstruction system would improve scalability and usability.

Third, our generative exploration in Chapter 4 is still constrained in model and data scale. The current model is 2B parameters—far smaller than proprietary systems like Veo 3—and trained only on academic multiview datasets. Scaling to larger models and training directly on internet-scale video collections would likely yield further gains in generalization and fidelity. Moreover, current generation remains bounded in length and consistency. Extending view synthesis to produce open-ended, 3D-consistent videos over long camera trajectories remains a key challenge. Future research could explore autoregressive or recurrent architectures that maintain both local photorealism and global scene coherence over time.

5.2 The future of 3D

This thesis was written during a time of explosive progress in AI. In contrast, 3D research has advanced more slowly—constrained by data scarcity, architectural uncertainty, and the unique demands of geometric consistency. As 3D researchers, we are often asked: why 3D? Is it fundamentally important for the future of AI? These questions push us beyond our technical comfort zones to consider the broader relevance of our work.

While the verdict will ultimately be left to future generations, this thesis offers a few personal hunches about where 3D is heading.

Multimodality. Compared to text or audio, 3D may not seem “special.” But in practice, modalities often bootstrap one another. Our own experiments show that strong 2D generative models can transfer useful priors to 3D view synthesis. As foundation models evolve, we expect the boundaries between modalities to blur: general-purpose models will natively handle images, video, 3D, text, and more—allowing them to perceive, imagine, and act in grounded environments. Because our physical world is inherently 3D, we believe 3D will remain a vital input and output interface.

Geometry as interaction. The boundary between geometry and photorealism is fading. In the past, artists needed clean 3D assets—meshes, textures, depth—to drive applications. But as neural representations become more powerful and interactive, users increasingly care less about how a 3D model is structured and more about how it behaves: Is it view-consistent? Can I move through it? Does it respond to my edits? For many applications, implicit representations can now satisfy these requirements without ever producing explicit geometry. Of course, task-specific constraints will always matter, but the line between “rendering” and “reasoning” is becoming less clear.

3D data growth. 3D data is poised to grow. Today, most 3D datasets are constructed from image or video collections where camera poses can be extracted. But the landscape is evolving: immersive virtual worlds, wearable sensors, and embodied agents are all generating massive volumes of spatial data. We expect this co-evolution—between representation, supervision, and environment—to be one of the key enablers of scalable 3D learning.

World modeling. We are just beginning to model the world. View synthesis is only a small step toward the broader goal of world modeling: constructing representations that support prediction, imagination, interaction, and reasoning. Future models will need to synthesize not just how the world looks, but how it changes, responds, and unfolds over time. In that future, 3D is not the goal—but it is an essential foundation.

Bibliography

- [1] Wenqi Xian et al. “Space-time Neural Irradiance Fields for Free-Viewpoint Video”. In: *CVPR*. 2021.
- [2] Edgar Tretschk et al. “Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a dynamic scene from monocular video”. In: *ICCV*. 2021.
- [3] Albert Pumarola et al. “D-NeRF: Neural Radiance Fields for Dynamic Scenes”. In: *CVPR*. 2021.
- [4] Zhengqi Li et al. “Neural Scene Flow Fields for Space-Time View Synthesis of Dynamic Scenes”. In: *CVPR*. 2021.
- [5] Keunhong Park et al. “Nerfies: Deformable Neural Radiance Fields”. In: *ICCV*. 2021.
- [6] Chen Gao et al. “Dynamic view synthesis from dynamic monocular video”. In: *ICCV*. 2021.
- [7] Keunhong Park et al. “HyperNeRF: A Higher-Dimensional Representation for Topologically Varying Neural Radiance Fields”. In: *ACM TOG* (2021).
- [8] Tianhao Wu et al. “D²NeRF: Self-Supervised Decoupling of Dynamic and Static Objects from a Monocular Video”. In: *NeurIPS*. 2022.
- [9] Zhou Wang et al. “Image quality assessment: from error visibility to structural similarity”. In: *IEEE TIP* (2004).
- [10] Richard Zhang et al. “The unreasonable effectiveness of deep features as a perceptual metric”. In: *CVPR*. 2018.
- [11] Nilesch Kulkarni, Abhinav Gupta, and Shubham Tulsiani. “Canonical surface mapping via geometric cycle consistency”. In: *ICCV*. 2019.
- [12] Volker Blanz and Thomas Vetter. “A morphable model for the synthesis of 3D faces”. In: *CVPR*. 1999.
- [13] Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. “Reconstructing 3d human pose from 2d image landmarks”. In: *ECCV*. 2012.
- [14] Federica Bogo et al. “Keep it SMPL: Automatic estimation of 3D human pose and shape from a single image”. In: *ECCV*. 2016.
- [15] Abhishek Kar et al. “Category-specific object reconstruction from a single image”. In: *CVPR*. 2015.

- [16] Angjoo Kanazawa et al. “Learning Category-Specific Mesh Reconstruction from Image Collections”. In: *ECCV*. 2018.
- [17] Nilesch Kulkarni et al. “Articulation-aware Canonical Surface Mapping”. In: *CVPR*. 2020.
- [18] Gengshan Yang et al. “LASR: Learning Articulated Shape Reconstruction from a Monocular Video”. In: *CVPR*. 2021.
- [19] Gengshan Yang et al. “BANMo: Building Animatable 3D Neural Models from Many Casual Videos”. In: *CVPR*. 2022.
- [20] Brian Curless and Marc Levoy. “A volumetric method for building complex models from range images”. In: *ACM CGIT*. 1996.
- [21] Richard Newcombe, Dieter Fox, and Steve Seitz. “DynamicFusion: Reconstruction and Tracking of Non-rigid Scenes in Real-Time”. In: *CVPR*. 2015.
- [22] Hao Li, Robert W. Sumner, and Mark Pauly. “Global Correspondence Optimization for Non-Rigid Registration of Depth Scans”. In: *SGP* (2008).
- [23] Wei Gao and Russ Tedrake. “SurfelWarp: Efficient Non-Volumetric Single View Dynamic Reconstruction”. In: *RSS*. 2018.
- [24] Aljaz Bozic et al. “Neural Non-Rigid Tracking”. In: *NeurIPS*. 2020.
- [25] Yi Yang and Deva Ramanan. “Articulated human detection with flexible mixtures of parts”. In: *IEEE TPAMI* (2012).
- [26] Hyun Soo Park et al. “3D Reconstruction of a Moving Point from a Series of 2D Projections”. In: *ECCV*. 2010.
- [27] Kaan Yucer et al. “Reconstruction of Articulated Objects from a Moving Camera”. In: *ICCVW*. 2015.
- [28] Youngjoong Kwon et al. “Neural human performer: Learning generalizable radiance fields for human performance rendering”. In: *NeurIPS* (2021).
- [29] Atsuhiko Noguchi et al. “Neural Articulated Radiance Field”. In: *ICCV*. 2021.
- [30] Sida Peng et al. “Neural Body: Implicit Neural Representations with Structured Latent Codes for Novel View Synthesis of Dynamic Humans”. In: *CVPR*. 2021.
- [31] Shih-Yang Su et al. “A-NeRF: Articulated Neural Radiance Fields for Learning Human Shape, Appearance, and Pose”. In: *NeurIPS*. 2021.
- [32] Chung-Yi Weng et al. “Humannerf: Free-viewpoint rendering of moving people from monocular video”. In: *CVPR*. 2022.
- [33] Guy Gafni et al. “Dynamic Neural Radiance Fields for Monocular 4D Facial Avatar Reconstruction”. In: *CVPR*. 2021.
- [34] Tianye Li et al. “Neural 3d Video Synthesis from Multi-view Video”. In: *CVPR*. 2022.
- [35] Yilun Du et al. “Neural radiance flow for 4d view synthesis and video processing”. In: *ICCV*. 2021.

- [36] Ben Mildenhall et al. “NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis”. In: *ECCV*. 2020.
- [37] Ruilong Li et al. “Tava: Template-free animatable volumetric actors”. In: *CVPR*. 2022.
- [38] Zachary Teed and Jia Deng. “Raft: Recurrent all-pairs field transforms for optical flow”. In: *ECCV*. 2020.
- [39] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. “Vision transformers for dense prediction”. In: *ICCV*. 2021.
- [40] Jae Shin Yoon et al. “Novel view synthesis of dynamic scenes with globally coherent depths from a monocular camera”. In: *CVPR*. 2020.
- [41] Alex Yu et al. “pixelNeRF: Neural Radiance Fields from One or Few Images”. In: *CVPR*. 2021.
- [42] Leon A Gatys et al. “Controlling perceptual factors in neural style transfer”. In: *CVPR*. 2017.
- [43] Minyoung Huh et al. “Transforming and projecting images into class-conditional generative networks”. In: *ECCV*. 2020.
- [44] Guilin Liu et al. “Image inpainting for irregular holes using partial convolutions”. In: *ECCV*. 2018.
- [45] Xu Chen et al. “SNARF: Differentiable Forward Skinning for Animating Non-Rigid Neural Implicit Shapes”. In: *ICCV*. 2021.
- [46] James Bradbury et al. *JAX: composable transformations of Python+NumPy programs*. 2018.
- [47] Zhengqi Li et al. *Neural Scene Flow Fields for Space-Time View Synthesis of Dynamic Scenes: Official Code Release*. 2021.
- [48] kweal23. *Neural Scene Flow Fields for Space-Time View Synthesis of Dynamic Scenes: Pytorch Re-implementation*. 2021.
- [49] Jonathan T Barron et al. “Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields”. In: *CVPR*. 2022.
- [50] Santosh K Divvala et al. “An empirical study of context in object detection”. In: *CVPR*. 2009.
- [51] Maxim Tatarchenko et al. “What do single-view 3d reconstruction networks learn?” In: *CVPR*. 2019.
- [52] Kevin Musgrave, Serge Belongie, and Ser-Nam Lim. “A metric learning reality check”. In: *ECCV*. 2020.
- [53] Bernhard Kerbl et al. “3D Gaussian Splatting for Real-Time Radiance Field Rendering”. In: *ACM Transactions on Graphics* 42.4 (2023).
- [54] Michael Broxton et al. “Immersive light field video with a layered mesh representation”. In: *ACM TOG* (2020).

- [55] Sara Fridovich-Keil et al. “K-planes: Explicit radiance fields in space, time, and appearance”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 12479–12488.
- [56] Jonathon Luiten et al. “Dynamic 3D Gaussians: Tracking by Persistent Dynamic View Synthesis”. In: *ArXiv abs/2308.09713* (2023).
- [57] Guanjun Wu et al. “4d gaussian splatting for real-time dynamic scene rendering”. In: *arXiv preprint arXiv:2310.08528* (2023).
- [58] Xiuye Gu et al. “HPLFlowNet: Hierarchical Permutohedral Lattice FlowNet for Scene Flow Estimation on Large-Scale Point Clouds”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), pp. 3249–3258.
- [59] Chaoyang Wang et al. “Neural Prior for Trajectory Estimation”. In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2022), pp. 6522–6532.
- [60] Xingyu Liu, C. Qi, and Leonidas J. Guibas. “Learning Scene Flow in 3D Point Clouds”. In: *ArXiv abs/1806.01411* (2018).
- [61] Gilles Puy, Alexandre Boulch, and Renaud Marlet. “FLOT: Scene Flow on Point Clouds Guided by Optimal Transport”. In: *European Conference on Computer Vision*. 2020.
- [62] Zirui Wang et al. “FlowNet3D++: Geometric Losses For Deep Scene Flow Estimation”. In: *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)* (2019), pp. 91–98.
- [63] Zhengqi Li et al. “Dynibar: Neural dynamic image-based rendering”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 4273–4284.
- [64] Chen Gao et al. “Dynamic View Synthesis from Dynamic Monocular Video”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2021.
- [65] Ziyi Yang et al. “Deformable 3D Gaussians for High-Fidelity Monocular Dynamic Scene Reconstruction”. In: *arXiv preprint arXiv:2309.13101* (2023).
- [66] Berthold K. P. Horn and Brian G. Schunck. “Determining Optical Flow”. In: *Other Conferences*. 1981.
- [67] Michael J Black and Padmanabhan Anandan. “A framework for the robust estimation of optical flow”. In: *1993 (4th) International Conference on Computer Vision*. IEEE. 1993, pp. 231–236.
- [68] Bruce D. Lucas and Takeo Kanade. “An Iterative Image Registration Technique with an Application to Stereo Vision”. In: *International Joint Conference on Artificial Intelligence*. 1981.
- [69] Andrés Bruhn, Joachim Weickert, and Christoph Schnörr. “Lucas/Kanade meets Horn/Schunck: Combining local and global optic flow methods”. In: *International journal of computer vision* 61 (2005), pp. 211–231.

- [70] Thomas Brox, Christoph Bregler, and Jitendra Malik. “Large displacement optical flow”. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition* (2009), pp. 41–48.
- [71] Thomas Brox et al. “High Accuracy Optical Flow Estimation Based on a Theory for Warping”. In: *European Conference on Computer Vision*. 2004.
- [72] Deqing Sun et al. “PWC-Net: CNNs for Optical Flow Using Pyramid, Warping, and Cost Volume”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2017), pp. 8934–8943.
- [73] Alexey Dosovitskiy et al. “FlowNet: Learning Optical Flow with Convolutional Networks”. In: *2015 IEEE International Conference on Computer Vision (ICCV)* (2015), pp. 2758–2766.
- [74] Zhaoyang Huang et al. “Flowformer: A transformer architecture for optical flow”. In: *European Conference on Computer Vision*. Springer. 2022, pp. 668–685.
- [75] Shihao Jiang et al. “Learning to estimate hidden motions with global motion aggregation”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 9772–9781.
- [76] Eddy Ilg et al. “FlowNet 2.0: Evolution of optical flow estimation with deep networks”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 2462–2470.
- [77] Hao-fei Xu et al. “Gmflow: Learning optical flow via global matching”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 8121–8130.
- [78] Shihao Jiang et al. “Learning optical flow from a few matches”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021, pp. 16592–16600.
- [79] Joel Janai et al. “Unsupervised learning of multi-frame optical flow with occlusions”. In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 690–706.
- [80] Zhile Ren et al. “A fusion approach for multi-frame optical flow estimation”. In: *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE. 2019, pp. 2077–2086.
- [81] Xiaoyu Shi et al. “Videoflow: Exploiting temporal cues for multi-frame optical flow estimation”. In: *arXiv preprint arXiv:2303.08340* (2023).
- [82] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. “SURF: Speeded Up Robust Features”. In: *European Conference on Computer Vision*. 2006.
- [83] Ethan Rublee et al. “ORB: An efficient alternative to SIFT or SURF”. In: *2011 International Conference on Computer Vision* (2011), pp. 2564–2571.
- [84] David G. Lowe. “Distinctive Image Features from Scale-Invariant Keypoints”. In: *International Journal of Computer Vision* 60 (2004), pp. 91–110.

- [85] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. “SuperPoint: Self-Supervised Interest Point Detection and Description”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* (2017), pp. 337–33712.
- [86] Ce Liu, Jenny Yuen, and Antonio Torralba. “SIFT Flow: Dense Correspondence across Scenes and Its Applications”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33 (2011), pp. 978–994.
- [87] Michael Rubinstein and Ce Liu. “Towards Longer Long-Range Motion Trajectories”. In: *British Machine Vision Conference*. 2012.
- [88] Stanley T Birchfield and Shrinivas J Pundlik. “Joint tracking of features and edges”. In: *2008 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2008, pp. 1–6.
- [89] P Sand. “Long-range video motion estimation using point trajectories”. PhD thesis. Ph. D. dissertation, Cambridge, MA, USA, 2006, adviser-Teller, Seth.
- [90] Peter Sand and Seth Teller. “Particle video: Long-range motion estimation using point trajectories”. In: *International journal of computer vision* 80 (2008), pp. 72–91.
- [91] Josef Sivic, Frederik Schaffalitzky, and Andrew Zisserman. “Object Level Grouping for Video Shots”. In: *International Journal of Computer Vision* 67 (2004), pp. 189–210.
- [92] Heng Wang and Cordelia Schmid. “Action Recognition with Improved Trajectories”. In: *2013 IEEE International Conference on Computer Vision* (2013), pp. 3551–3558.
- [93] Qianqian Wang et al. “Tracking Everything Everywhere All at Once”. In: *International Conference on Computer Vision*. 2023.
- [94] Michal Neoral, Jonáš Šerých, and Jiří Matas. “MFT: Long-Term Tracking of Every Pixel”. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. Jan. 2024, pp. 6837–6847.
- [95] Adam W Harley, Zhaoyuan Fang, and Katerina Fragkiadaki. “Particle video revisited: Tracking through occlusions using point trajectories”. In: *European Conference on Computer Vision*. Springer. 2022, pp. 59–75.
- [96] Nikita Karaev et al. “CoTracker: It is Better to Track Together”. In: 2023.
- [97] Carl Doersch et al. “TAPIR: Tracking Any Point with per-frame Initialization and temporal Refinement”. In: *arXiv preprint arXiv:2306.08637* (2023).
- [98] Carl Doersch et al. “Tap-vid: A benchmark for tracking any point in a video”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 13610–13626.
- [99] Yang Zheng et al. “Pointodyssey: A large-scale synthetic dataset for long-term point tracking”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, pp. 19855–19865.
- [100] Deqing Sun, Erik B. Sudderth, and Hanspeter Pfister. “Layered RGBD scene flow estimation”. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015), pp. 548–556.

- [101] Zachary Teed and Jia Deng. “RAFT-3D: Scene Flow using Rigid-Motion Embeddings”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021.
- [102] Mariano Jaimez et al. “A primal-dual framework for real-time dense RGB-D scene flow”. In: *2015 IEEE International Conference on Robotics and Automation (ICRA)* (2015), pp. 98–104. URL: <https://api.semanticscholar.org/CorpusID:16757067>.
- [103] Julian Quiroga et al. “Dense Semi-rigid Scene Flow Estimation from RGBD Images”. In: *European Conference on Computer Vision*. 2014. URL: <https://api.semanticscholar.org/CorpusID:10240196>.
- [104] Sundar Vedula et al. “Three-dimensional scene flow”. In: *IEEE transactions on pattern analysis and machine intelligence* 27.3 (2005), pp. 475–480.
- [105] Lukas Mehl et al. “M-FUSE: Multi-frame fusion for scene flow estimation”. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2023, pp. 2020–2029.
- [106] Gengshan Yang et al. “ViSER: Video-Specific Surface Embeddings for Articulated 3D Shape Reconstruction”. In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato et al. Vol. 34. Curran Associates, Inc., 2021, pp. 19326–19338. URL: https://proceedings.neurips.cc/paper_files/paper/2021/file/a11f9e533f28593768ebf87075ab34f2-Paper.pdf.
- [107] Junhwa Hur and Stefan Roth. “Self-Supervised Monocular Scene Flow Estimation”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2020), pp. 7394–7403.
- [108] Gengshan Yang et al. “LASR: Learning Articulated Shape Reconstruction from a Monocular Video”. In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2021), pp. 15975–15984. URL: <https://api.semanticscholar.org/CorpusID:234093797>.
- [109] Gengshan Yang et al. “BANMo: Building Animatable 3D Neural Models from Many Casual Videos”. In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2021), pp. 2853–2863. URL: <https://api.semanticscholar.org/CorpusID:245425066>.
- [110] Michael Zollhöfer et al. “Real-time non-rigid reconstruction using an RGB-D camera”. In: *ACM Transactions on Graphics (TOG)* 33 (2014), pp. 1–12.
- [111] Richard A. Newcombe, Dieter Fox, and Steven M. Seitz. “DynamicFusion: Reconstruction and tracking of non-rigid scenes in real-time”. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015), pp. 343–352.
- [112] Aljaz Bozic et al. “DeepDeform: Learning Non-Rigid RGB-D Reconstruction With Semi-Supervised Data”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), pp. 7000–7010.

- [113] Mingsong Dou et al. “Fusion4D”. In: *ACM Transactions on Graphics (TOG)* 35 (2016), pp. 1–13.
- [114] Matthias Innmann et al. “VolumeDeform: Real-Time Volumetric Non-rigid Reconstruction”. In: *European Conference on Computer Vision*. 2016.
- [115] Chris Russell, Rui Yu, and Lourdes Agapito. “Video pop-up: Monocular 3d reconstruction of dynamic scenes”. In: *European conference on computer vision*. Springer. 2014, pp. 583–598.
- [116] Suryansh Kumar, Yuchao Dai, and Hongdong Li. “Monocular Dense 3D Reconstruction of a Complex Dynamic Scene from Two Perspective Frames”. In: *2017 IEEE International Conference on Computer Vision (ICCV)* (2017), pp. 4659–4667.
- [117] Rene Ranftl et al. “Dense Monocular Depth Estimation in Complex Dynamic Scenes”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016.
- [118] Zhengqi Li et al. “Learning the depths of moving people by watching frozen people”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 4521–4530.
- [119] Zhoutong Zhang et al. “Consistent depth of moving objects in video”. In: *ACM Transactions on Graphics (TOG)* 40.4 (2021), pp. 1–12.
- [120] Xuan Luo et al. “Consistent video depth estimation”. In: *ACM Transactions on Graphics (ToG)* 39.4 (2020), pp. 71–1.
- [121] Johannes Kopf, Xuejian Rong, and Jia-Bin Huang. “Robust consistent video depth estimation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 1611–1621.
- [122] Zhoutong Zhang et al. “Structure and motion from casual videos”. In: *European Conference on Computer Vision*. Springer. 2022, pp. 20–37.
- [123] Bernhard Kerbl et al. “3D Gaussian Splatting for Real-Time Radiance Field Rendering”. In: *ACM Transactions on Graphics* 42.4 (July 2023). URL: <https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/>.
- [124] Aayush Bansal et al. “4d visualization of dynamic events from unconstrained multi-view videos”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 5366–5375.
- [125] Tianye Li et al. “Neural 3d video synthesis from multi-view video”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 5521–5531.
- [126] Ang Cao and Justin Johnson. “Hexplane: A fast representation for dynamic scenes”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 130–141.

- [127] Liangchen Song et al. “Nerfplayer: A streamable dynamic scene representation with decomposed neural radiance fields”. In: *IEEE Transactions on Visualization and Computer Graphics* 29.5 (2023), pp. 2732–2742.
- [128] Timo Stich et al. “View and time interpolation in image space”. In: *Computer Graphics Forum*. Vol. 27. 7. Wiley Online Library. 2008, pp. 1781–1787.
- [129] Liao Wang et al. “Fourier plenotrees for dynamic radiance field rendering in real-time”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 13524–13534.
- [130] Zhan Li et al. “Spacetime Gaussian Feature Splatting for Real-Time Dynamic View Synthesis”. In: *arXiv preprint arXiv:2312.16812* (2023).
- [131] Mustafa Işık et al. “Humanrf: High-fidelity neural radiance fields for humans in motion”. In: *arXiv preprint arXiv:2305.06356* (2023).
- [132] Albert Pumarola et al. “D-nerf: Neural radiance fields for dynamic scenes”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 10318–10327.
- [133] Chaoyang Wang et al. “Neural trajectory fields for dynamic novel view synthesis”. In: *arXiv preprint arXiv:2105.05994* (2021).
- [134] Yuanxing Duan et al. “4D Gaussian Splatting: Towards Efficient Novel View Synthesis for Dynamic Scenes”. In: *arXiv preprint arXiv:2402.03307* (2024).
- [135] Zeyu Yang et al. “Real-time photorealistic dynamic scene representation and rendering with 4d gaussian splatting”. In: *arXiv preprint arXiv:2310.10642* (2023).
- [136] Hang Gao et al. “Dynamic Novel-View Synthesis: A Reality Check”. In: *NeurIPS*. 2022.
- [137] Agelos Kratimenos, Jiahui Lei, and Kostas Daniilidis. “DynMF: Neural Motion Factorization for Real-time Dynamic View Synthesis with 3D Gaussian Splatting”. In: *arXiv* (2023).
- [138] Lihe Yang et al. “Depth Anything: Unleashing the Power of Large-Scale Unlabeled Data”. In: *arXiv:2401.10891* (2024).
- [139] Thorsten Hempel, Ahmed A Abdelrahman, and Ayoub Al-Hamadi. “6d rotation representation for unconstrained head pose estimation”. In: *2022 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2022, pp. 2496–2500.
- [140] Alexander Kirillov et al. “Segment Anything”. In: *2023 IEEE/CVF International Conference on Computer Vision (ICCV)* (2023), pp. 3992–4003.
- [141] Jinyu Yang et al. *Track Anything: Segment Anything Meets Videos*. 2023. arXiv: [2304.11968 \[cs.CV\]](#).
- [142] Luigi Piccinelli et al. “UniDepth: Universal Monocular Metric Depth Estimation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2024.

- [143] Zachary Teed and Jia Deng. “Droid-slam: Deep visual slam for monocular, stereo, and rgb-d cameras”. In: *Advances in neural information processing systems* 34 (2021), pp. 16558–16569.
- [144] Johannes L Schonberger and Jan-Michael Frahm. “Structure-from-motion revisited”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 4104–4113.
- [145] Diederik P Kingma, J Adam Ba, and J Adam. “A method for stochastic optimization. arXiv 2014”. In: *arXiv preprint arXiv:1412.6980* 106 (2020).
- [146] Klaus Greff et al. “Kubric: A scalable dataset generator”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 3749–3761.
- [147] Binbin Huang et al. “2D Gaussian Splatting for Geometrically Accurate Radiance Fields”. In: *SIGGRAPH 2024 Conference Papers*. Association for Computing Machinery, 2024. DOI: [10.1145/3641519.3657428](https://doi.org/10.1145/3641519.3657428).
- [148] Johannes Lutz Schönberger and Jan-Michael Frahm. “Structure-from-Motion Revisited”. In: *CVPR*. 2016.
- [149] Jiahui Lei et al. “MoSca: Dynamic Gaussian Fusion from Casual Videos via 4D Motion Scaffolds”. In: *arXiv preprint arXiv:2405.17421* (2024).
- [150] Shizun Wang et al. “GFlow: Recovering 4D World from Monocular Video”. In: *arXiv preprint arXiv:2405.18426* (2024).
- [151] Colton Stearns et al. “Dynamic Gaussian Marbles for Novel View Synthesis of Casual Monocular Videos”. In: *arXiv preprint arXiv:2406.18717* (2024).
- [152] Qingming Liu et al. “MoDGS: Dynamic Gaussian Splatting from Causally-captured Monocular Videos”. In: *arXiv preprint arXiv:2406.00434* (2024).
- [153] Wen-Hsuan Chu, Lei Ke, and Katerina Fragkiadaki. “DreamScene4D: Dynamic Multi-Object Scene Generation from Monocular Videos”. In: *arXiv preprint arXiv:2405.02280* (2024).
- [154] Thomas Müller et al. “Instant Neural Graphics Primitives with a Multiresolution Hash Encoding”. In: *ACM Trans. Graph.* (2022).
- [155] Alex Yu et al. “pixelNeRF: Neural Radiance Fields from One or Few Images”. In: *CVPR*. 2021.
- [156] Ben Poole et al. “DreamFusion: Text-to-3D using 2D Diffusion”. In: *arXiv* (2022).
- [157] Ruoshi Liu et al. “Zero-1-to-3: Zero-shot one image to 3D object”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023.
- [158] Rundi Wu et al. “Reconfusion: 3d reconstruction with diffusion priors”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 21551–21561.

- [159] Ruiqi Gao* et al. “CAT3D: Create Anything in 3D with Multi-View Diffusion Models”. In: *arXiv* (2024).
- [160] Wangbo Yu et al. “ViewCrafter: Taming video diffusion models for high-fidelity novel view synthesis”. In: *arXiv preprint arXiv:2409.02048* (2024).
- [161] Zhouxia Wang et al. “Motionctrl: A unified and flexible motion controller for video generation”. In: *ACM SIGGRAPH 2024 Conference Papers*. 2024, pp. 1–11.
- [162] Haian Jin et al. *LVSM: A Large View Synthesis Model with Minimal 3D Inductive Bias*. 2024. arXiv: 2410.17242 [cs.CV]. URL: <https://arxiv.org/abs/2410.17242>.
- [163] Daniel Watson et al. “Controlling space and time with diffusion models”. In: *arXiv preprint arXiv:2407.07860* (2024).
- [164] Kyle Sargent et al. “ZeroNVS: Zero-Shot 360-Degree View Synthesis from a Single Real Image”. In: *arXiv preprint arXiv:2310.17994* (2023).
- [165] *Virtual Camera System*. https://en.wikipedia.org/wiki/Virtual_camera_system. Accessed: 5-Mar-2025.
- [166] David Charatan et al. “pixelSplat: 3D Gaussian Splats from Image Pairs for Scalable Generalizable 3D Reconstruction”. In: *arXiv*. 2023.
- [167] Yuedong Chen et al. “Mvsplat: Efficient 3d gaussian splatting from sparse multi-view images”. In: *European Conference on Computer Vision*. Springer. 2025, pp. 370–386.
- [168] Chen Ziwen et al. “Long-lrm: Long-sequence large reconstruction model for wide-coverage gaussian splats”. In: *arXiv preprint arXiv:2410.12781* (2024).
- [169] Vikram Voleti et al. “SV3D: Novel Multi-view Synthesis and 3D Generation from a Single Image using Latent Video Diffusion”. In: *European Conference on Computer Vision*. 2024.
- [170] Ben Mildenhall et al. “NeRF: Representing scenes as neural radiance fields for view synthesis”. In: *Communications of the ACM* (2021).
- [171] Jonathan Ho, Ajay Jain, and Pieter Abbeel. “Denoising diffusion probabilistic models”. In: *Advances in neural information processing systems* 33 (2020), pp. 6840–6851.
- [172] Andreas Blattmann et al. “Stable video diffusion: Scaling latent video diffusion models to large datasets”. In: *arXiv preprint arXiv:2311.15127* (2023).
- [173] Hao He et al. “Cameractrl: Enabling camera control for text-to-video generation”. In: *arXiv preprint arXiv:2404.02101* (2024).
- [174] Yingqing He et al. “Latent Video Diffusion Models for High-Fidelity Long Video Generation”. In: *arXiv* (2022).
- [175] Robin Rombach et al. “High-Resolution Image Synthesis With Latent Diffusion Models”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2022, pp. 10684–10695.
- [176] Yichun Shi et al. “MVDream: Multi-view diffusion for 3d generation”. In: *arXiv preprint arXiv:2308.16512* (2023).

- [177] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.
- [178] P Goyal. “Accurate, large minibatch SG D: training imagenet in 1 hour”. In: *arXiv preprint arXiv:1706.02677* (2017).
- [179] Andreas Blattmann et al. “Align your latents: High-resolution video synthesis with latent diffusion models”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 22563–22575.
- [180] Julius Plucker. “Xvii. on a new geometry of space”. In: *Philosophical Transactions of the Royal Society of London* 155 (1865), pp. 725–791.
- [181] Chuanxia Zheng and Andrea Vedaldi. “Free3D: Consistent Novel View Synthesis without 3D Representation”. In: *arXiv preprint arXiv:2312.04551* (2023).
- [182] Alec Radford et al. “Learning transferable visual models from natural language supervision”. In: *International conference on machine learning*. PMLR. 2021, pp. 8748–8763.
- [183] Emiel Hoogeboom, Jonathan Heek, and Tim Salimans. “simple diffusion: End-to-end diffusion for high resolution images”. In: *arXiv preprint arXiv:2301.11093* (2023).
- [184] Patrick Esser et al. “Scaling rectified flow transformers for high-resolution image synthesis”. In: *ICLR*. 2024.
- [185] Tong Wu et al. “Omniobject3D: Large-vocabulary 3D object dataset for realistic perception, reconstruction and generation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 803–814.
- [186] Laura Downs et al. “Google Scanned Objects: A high-quality dataset of 3D scanned household items”. In: *2022 International Conference on Robotics and Automation (ICRA)*. IEEE. 2022, pp. 2553–2560.
- [187] Ben Mildenhall et al. “Local light field fusion: Practical view synthesis with prescriptive sampling guidelines”. In: *ACM Transactions on Graphics (ToG)* 38.4 (2019), pp. 1–14.
- [188] Rasmus Jensen et al. “Large scale multi-view stereopsis evaluation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 406–413.
- [189] Jeremy Reizenstein et al. “Common Objects in 3D: Large-Scale Learning and Evaluation of Real-life 3D Category Reconstruction”. In: *International Conference on Computer Vision*. 2021.
- [190] Hongchi Xia et al. *RGBD Objects in the Wild: Scaling Real-World 3D Object Learning from RGB-D Videos*. 2024. arXiv: [2401.12592](https://arxiv.org/abs/2401.12592) [cs.CV].
- [191] Tinghui Zhou et al. “Stereo magnification: Learning view synthesis using multiplane images”. In: *arXiv preprint arXiv:1805.09817* (2018).
- [192] Lu Ling et al. “DI3dv-10k: A large-scale scene dataset for deep learning-based 3d vision”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 22160–22169.

- [193] Arno Knapitsch et al. “Tanks and temples: Benchmarking large-scale scene reconstruction”. In: *ACM Transactions on Graphics (ToG)* 36.4 (2017), pp. 1–13.
- [194] Haofei Xu et al. “Depthspat: Connecting gaussian splatting and depth”. In: *arXiv preprint arXiv:2410.13862* (2024).
- [195] Tsai-Shien Chen et al. “Motion-Conditioned Diffusion Model for Controllable Video Synthesis”. In: *arXiv preprint arXiv:2304.14404* (2023).
- [196] Jonathan T Barron et al. “Zip-nerf: Anti-aliased grid-based neural radiance fields”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, pp. 19697–19705.
- [197] Richard Zhang et al. “The unreasonable effectiveness of deep features as a perceptual metric”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 586–595.
- [198] Shakiba Kheradmand et al. “3D Gaussian Splatting as Markov Chain Monte Carlo”. In: *arXiv preprint arXiv:2404.09591* (2024).
- [199] Ziqi Huang et al. “VBench: Comprehensive Benchmark Suite for Video Generative Models”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024.
- [200] Jason J Yu et al. “Long-term photometric consistent novel view synthesis with diffusion models”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, pp. 7094–7104.
- [201] Huiyu Zhou, Yuan Yuan, and Chunmei Shi. “Object tracking using SIFT features and mean shift”. In: *Computer vision and image understanding* 113.3 (2009), pp. 345–352.
- [202] Baorui Ma et al. “You See it, You Got it: Learning 3D Creation on Pose-Free Videos at Scale”. In: 2024.
- [203] Shuzhe Wang et al. “DUS3R: Geometric 3D Vision Made Easy”. In: *CVPR*. 2024.
- [204] Robin Rombach et al. “High-resolution image synthesis with latent diffusion models”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 10684–10695.
- [205] Jonathan Ho and Tim Salimans. “Classifier-Free Diffusion Guidance”. In: *arXiv:2207.12598* (2022).
- [206] Adam Botach, Evgenii Zheltonozhskii, and Chaim Baskin. “End-to-End Referring Video Object Segmentation with Multimodal Transformers”. In: *CVPR*. 2022.
- [207] Martin A Fischler and Robert C Bolles. “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography”. In: *CACM* (1981).
- [208] Thomas Brox et al. “High accuracy optical flow estimation based on a theory for warping”. In: *ECCV*. 2004.

- [209] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *CACM* (2017).
- [210] Marek Simonik. *Record3D – Point Cloud Animation and Streaming*. 2019.
- [211] Hao Ouyang et al. “Real-Time Neural Character Rendering with Pose-Guided Multiplane Images”. In: *ECCV*. 2022.
- [212] David G Lowe. “Distinctive image features from scale-invariant keypoints”. In: *IJCV* (2004).
- [213] Marco Zuliani. “RANSAC for Dummies”. In: *Vision Research Lab, University of California, Santa Barbara* (2009).
- [214] G. Bradski. “The OpenCV Library”. In: *Dr. Dobb’s Journal of Software Tools* (2000).
- [215] Irwin Sobel. “An Isotropic 3x3 Image Gradient Operator”. In: *Presentation at Stanford A.I. Project 1968* (2014).
- [216] Tsung-Yi Lin et al. “Microsoft coco: Common objects in context”. In: *ECCV*. 2014.
- [217] Benjamin Biggs et al. “Who left the dogs out? 3d animal reconstruction with expectation maximization in the loop”. In: *ECCV*. 2020.
- [218] Richard A Newcombe et al. “Kinectfusion: Real-time dense surface mapping and tracking”. In: *IEEE ISMAR*. 2011.
- [219] Ricardo Martin-Brualla et al. “Nerf in the wild: Neural radiance fields for unconstrained photo collections”. In: *CVPR*. 2021.
- [220] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *ICLR*. 2014.
- [221] Jing He et al. *Lotus: Diffusion-based Visual Foundation Model for High-quality Dense Prediction*. 2024. arXiv: 2409.18124 [cs.CV]. URL: <https://arxiv.org/abs/2409.18124>.
- [222] Gal Metzer et al. “Latent-NeRF for shape-guided generation of 3D shapes and textures”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 12663–12673.
- [223] Yang Song and Stefano Ermon. “Generative Modeling by Estimating Gradients of the Data Distribution”. In: *arXiv:1907.05600* (2019).
- [224] Mohammed Suhail et al. “Light field neural rendering”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 8269–8279.
- [225] Stanislaw Szymanowicz, Christian Rupprecht, and Andrea Vedaldi. “Splatter image: Ultra-fast single-view 3d reconstruction”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2024, pp. 10208–10217.
- [226] Yicong Hong et al. “LRM: Large reconstruction model for single image to 3D”. In: *arXiv preprint arXiv:2311.04400* (2023).

- [227] Daniel Watson et al. *Novel View Synthesis with Diffusion Models*. 2022. arXiv: [2210.04628 \[cs.CV\]](#).
- [228] Ruoshi Liu et al. “Zero-1-to-3: Zero-shot One Image to 3D Object”. In: *International Conference on Computer Vision* (2023).
- [229] Ruoxi Shi et al. “Zero123++: a single image to consistent multi-view diffusion base model”. In: *arXiv preprint arXiv:2310.15110* (2023).
- [230] Yuan Liu et al. “SyncDreamer: Generating Multiview-consistent Images from a Single-view Image”. In: *arXiv preprint arXiv:2309.03453* (2023).
- [231] Antoine Mercier et al. “HexaGen3D: StableDiffusion is just one step away from Fast and Diverse Text-to-3D Generation”. In: *arXiv preprint arXiv:2401.07727* (2024).
- [232] Sherwin Bahmani et al. “4d-fy: Text-to-4D generation using hybrid score distillation sampling”. In: *arXiv preprint arXiv:2311.17984* (2023).
- [233] Petar Veličković et al. “softmax is not enough (for sharp out-of-distribution)”. In: *arXiv preprint arXiv:2410.01104* (2024).
- [234] Kiwhan Song et al. *History-Guided Video Diffusion*. 2025. arXiv: [2502.06764 \[cs.LG\]](#). URL: <https://arxiv.org/abs/2502.06764>.
- [235] Boyuan Chen et al. *Diffusion Forcing: Next-token Prediction Meets Full-Sequence Diffusion*. 2024. arXiv: [2407.01392 \[cs.LG\]](#). URL: <https://arxiv.org/abs/2407.01392>.
- [236] Stability AI. *Stable Diffusion 3.5*. <https://stability.ai/news/introducing-stable-diffusion-3-5>. 2024.

Appendix A

Monocular dynamic view synthesis: A reality check

A.1 Outline

In this Appendix, we describe in detail the following:

- Computation for effective multi-view factors (EMFs) in Section A.2.
- Computation for co-visibility mask and masked image metrics in Section A.3.
- Summary of existing works and correspondence readout in Section A.4.
- Summary of the capture setup and data processing for our iPhone dataset in Section A.5.
- Summary of the implementation details and remain differences in Section A.6.
- Additional results on the impact of effective multi-view in Section A.7.
- Additional results on per-sequence performance breakdown in Section A.8.
- Additional results on novel-view synthesis in Section A.9.
- Additional results on inferred correspondence in Section A.10.

For better demonstration, we strongly recommend visiting our [project page](#) for videos of the capture and result visualizations.

A.2 Computation for effective multi-view factors (EMFs)

To quantify the amount of effective multi-view in a sequence by the camera and scene motion magnitude, we propose two metrics as effective multi-view factors (EMFs), i.e., the Full EMF Ω and the angular EMF ω . Note that we design our metrics to be scale-agnostic such that we can compare them across different sequences of different world scales.

As in the main paper, we define a point $\mathbf{x}_t \in \mathbb{S}_t^2$ on the visible object’s surface and a camera parameterized by its origin \mathbf{o}_t at time $t \in \mathcal{T}$, where \mathcal{T} is the set of possible time steps.

A.2.1 Full EMF Ω : Ratio of camera-scene motion magnitude

We are interested in the relative scale of the camera motion compared to the object. Recall that we define Ω as the expected ratio over all visible pixels over time,

$$\Omega = \mathbb{E}_{t,t+1 \in \mathcal{T}} \left[\mathbb{E}_{\mathbf{x}_t \in \mathbb{S}_t^2} \left[\frac{\|\mathbf{o}_{t+1} - \mathbf{o}_t\|}{\|\mathbf{x}_{t+1} - \mathbf{x}_t\|} \right] \right]. \quad (\text{A.1})$$

The numerator is trivially computable given the camera information. We thus focus on the denominator, *i.e.*, the foreground 3D scene flow $\mathbf{x}_{t+1} - \mathbf{x}_t$.

We estimate 3D scene flow by combining the known cameras, dense 2D optical flow, and per-frame depth maps. We estimate the 2D optical flow using RAFT [38]. When metric depth is not available, *e.g.*, on previous datasets [4, 7, 5], we use DPT [39] for monocular depth estimation. Additionally, we need a foreground mask for the object, which we obtain through a video segmentation network [206]. For each pixel location \mathbf{u}_t at time t in the foreground mask, we can compute its 3D position \mathbf{x}_t by back-projection with the depth z_t . We then get the 2D pixel correspondence at time $t + 1$ by simply following the 2D optical flow $\mathbf{u}_{t+1} = \mathbf{u}_t + \mathbf{f}_{t \rightarrow t+1}(\mathbf{u}_t)$, where $\mathbf{f}_{t \rightarrow t+1}$ is a bilinearly interpolated forward flow map. After back-projection, we obtain the corresponding 3D point position \mathbf{x}_{t+1} at frame $t + 1$. In practice, extra care is needed for handling the unknown depth scale from model prediction and occlusion, discussed next.

Aligning depth maps of unknown scales. The DPT [39] model predicts a disparity map in Euclidean space with an unknown scale a and shift b . To resolve the scale and shift ambiguity in the predicted disparity maps, we make use of the sparse 3D points extracted by COLMAP. For a frame at time t , we first calculate the actual disparity $1/\tilde{z}_t$ of the sparse 3D points by projecting them onto the image, which usually results in sub-pixels. We then bilinearly interpolate the predicted disparity map to get the predicted disparity $1/\tilde{z}_t^*$. Scale a and shift b can then be estimated through linear regression via the relation:

$$\frac{1}{\tilde{z}_t} = a \cdot \frac{1}{\tilde{z}_t^*} + b.$$

Note that the sparse 3D points from COLMAP [148] are all located on the static background. When projecting the sparse 3D points onto the image, some points might be occluded by the moving objects in the foreground. We handle occluded points by fitting a and b using RANSAC [207], which ignores outliers and is robust in practice.

Handling occlusions. We identify occlusions using a forward-backward consistency check following the method of Brox *et al.* [208]. We briefly summarize their method here.

Concretely, we identify an occlusion by chaining the forward flow $\mathbf{f}_{t \rightarrow t+1}$ and backward flow $\mathbf{f}_{t+1 \rightarrow t}$ and thresholding based on warp consistency. For those pixels that have inconsistent forward and backward optical flows, defined by regions where chained forward and backward flows result in non-zero flow values, satisfying the following inequality:

$$\begin{aligned} & \|\mathbf{f}_{t \rightarrow t+1}(\mathbf{u}_t) + \mathbf{f}_{t+1 \rightarrow t}(\mathbf{u}_t + \mathbf{f}_{t \rightarrow t+1}(\mathbf{u}_t))\|_2^2 \\ & \geq 0.01 \cdot (\|\mathbf{f}_{t \rightarrow t+1}(\mathbf{u}_t)\|_2^2 + \|\mathbf{f}_{t+1 \rightarrow t}(\mathbf{u}_t + \mathbf{f}_{t \rightarrow t+1}(\mathbf{u}_t))\|_2^2) + 0.5. \end{aligned} \quad (\text{A.2})$$

The occluded pixels, along with the background pixels not belonging to the foreground mask, are excluded from the 3D scene flow computation.

Discussion. In practice, we find that the Ω metric relies on the model estimation quality, in particular, the monocular depth prediction. We therefore design a second metric by measuring camera angular speed ω . With some practical assumptions, it circumvents Ω 's limitation and does not rely on any external model estimates.

A.2.2 Angular EMF ω : Camera angular velocity

We propose to measure camera angular speed ω given the camera parameters, frame rate N and a single 3D look-at point \mathbf{a} obtained by triangulating all cameras, following Nerfies [5]. Recall that ω is computed as the scaled expectation,

$$\omega = \mathbb{E}_{t, t+1 \in \mathcal{T}} \left[\arccos \left(\frac{\langle \mathbf{a} - \mathbf{o}_t, \mathbf{a} - \mathbf{o}_{t+1} \rangle}{\|\mathbf{a} - \mathbf{o}_t\| \cdot \|\mathbf{a} - \mathbf{o}_{t+1}\|} \right) \right] \cdot N. \quad (\text{A.3})$$

When computing this metric, we assume that (1) the object moves at roughly constant speed, (2) the camera always fixates on the object, and (3) the distance between the camera and the object remains approximately the same over time. All sequences from existing works as well as ours meet these assumptions, except those from the NSFF [4] and NV-DYN [40] datasets. In their case, the cameras are always facing forward, breaking the assumption (2). However, we find that even though cameras are not fixated on the object since they are static, we can still compute the look-at point \mathbf{a} by considering the center of mass of the foreground visible surfaces in 3D. Both datasets provide accurate foreground segmentations and MVS depth, which we use to identify and back-project foreground pixels into 3D space. The final look-at point is computed as the average foreground points over all frames.

Note that existing works only provide extracted frames from each sequence without specifying the frame rate. We identify the frame rates by re-assembling the original video using different FPS candidates and hand-picking the one that results in the most natural object and camera motion, which are verified by the original authors [4, 7, 5]. We document per-sequence FPS for future reference in Table A.1.

A.3 Computation for co-visibility mask and masked image metrics

Code for both the co-visibility mask and masked image metrics are made publicly available on our [project page](#). In this section, we provide further details for their computation processes.

Dataset	Sequence	#Frames	FPS	Ω	ω
D-NeRF [3]	BOUNCINGBALLS	150	30	15.52	1945.52
	HELLWARRIOR	100	30	10.32	2984.15
	HOOK	100	30	25.82	1996.95
	JUMPINGJACKS	200	30	10.64	1969.47
	LEGO	50	30	17.00	2133.78
	MUTANT	150	30	12.64	1908.67
	STANDUP	150	30	14.22	2011.31
	TREX	200	30	13.70	2133.78
HyperNeRF [7]	3D PRINTER	207	15	3.13	251.37
	CHICKEN	164	15	7.38	212.58
	PEEL BANANA	513	15	1.26	237.66
Nerfies [5]	BROOM	197	15	3.40	128.54
	CURLS	57	5	1.20	138.55
	TAIL	238	15	3.30	160.55
	TOBY SIT	308	15	2.18	110.51
NSFF [4]	BALLOON1	24	15	2.44	57.63
	BALLOON2	24	30	0.76	76.97
	DYNAMIC FACE	24	15	4.57	83.17
	JUMPING	24	30	0.68	53.79
	PLAYGROUND	24	30	0.36	71.56
	SKATING	24	30	0.79	42.76
	TRUCK	24	30	0.22	19.41
	UMBRELLA	24	15	0.62	20.66
iPhone (Ours)	APPLE	475	30	0.75	3.79
	BACKPACK	180	30	0.26	5.59
	BLOCK	350	30	0.04	11.50
	CREEPER	210	30	0.23	14.05
	HANDWAVY	303	30	0.05	13.66
	HARU	200	60	0.30	30.32
	MOCHI	180	60	0.07	14.49
	PAPER WINDMILL	277	30	0.38	10.71
	PILLOW	330	30	0.06	13.19
	SPACE OUT	429	30	0.13	6.37
	SPIN	426	30	0.15	7.86
	SRIRACHA	220	30	0.18	18.56
	TEDDY	350	30	0.20	7.62
	WHEEL	250	30	0.03	58.45

Table A.1: **Per-sequence breakdowns of the statistics of different datasets.** As in the main paper, we consider the multi-camera captures from three representative existing datasets: D-NeRF [3], Nerfies [5] and HyperNeRF [7]. We also provide per-sequence breakdowns for both the multi-camera and single-camera captures from our proposed iPhone dataset.

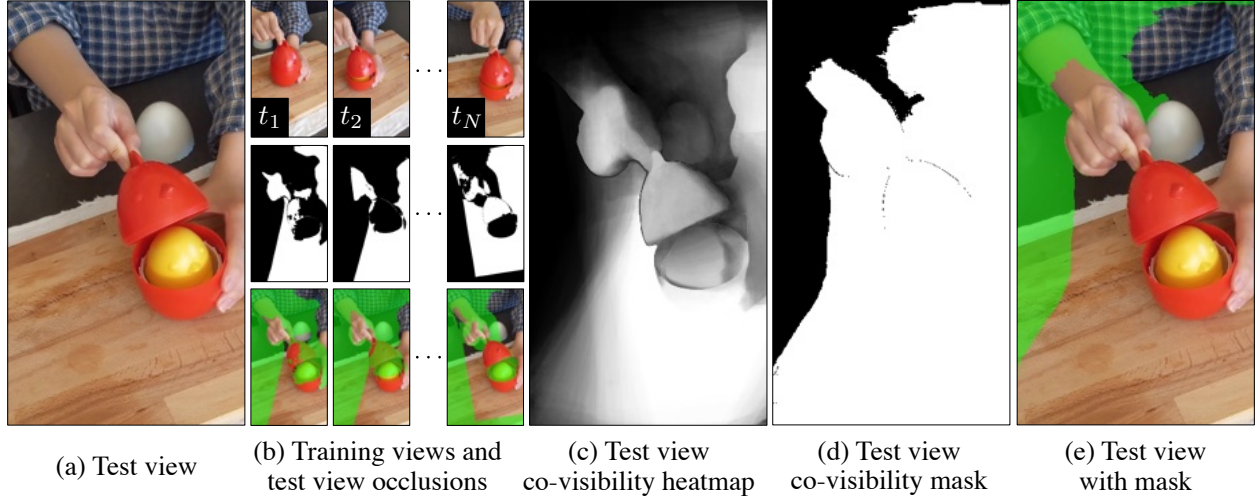


Figure A.1: **Illustration of the computation process for co-visibility.** Given a (a) test view, we first compute its pairwise (b) occlusions in all training views the forward-backward consistency check [208] based on the optical flow estimation from pre-trained RAFT [38]. The occlusions are visualized as binary masks in (b)’s second row, where black color indicates pixels without correspondence. We also visualize their overlays over the original test image. Then by summing up all occlusion maps, we compute the (c) test view co-visibility heatmap, which stores the number of times each test pixel is seen in training frames. Finally, we apply a threshold on the heatmap and obtain a binary (d) co-visibility mask. We also visualize its (e) overlay on the test image. Note that the occlusion maps are usually inaccurate due to noise in optical flow prediction, e.g., they miss the cover of the chicken toy in this example. Our conservative threshold strategy overcomes the noise and ensures that adequately seen regions are included in the final mask.

A.3.1 Co-visibility mask

In dynamic scenes, particularly for monocular capture with multi-camera validation, the test view contains regions that may not have been observed at all by the training camera. To circumvent this issue without resorting to camera teleportation, for each pixel in the test image, we propose “co-visibility” masking, which tests how many times a test pixel has been observed in the training images.

We visualize the computation process of the co-visibility mask in Figure A.1. Concretely, for each (a) test frame, we first check its (b) occlusion in each training frame by the forward-backward flow consistency check according to Equation A.2. We use RAFT [38] for optical flow estimation between each test frame and each training frame. Note that we visualize occlusion as both a binary mask and its overlay on the test image. For occlusion mask visualization, the black color indicates pixels with no correspondence in the training views. We then compute the (c) co-visibility heatmap by simply summing up all test view occlusion masks. This co-visibility heatmap stores the number of times that each pixel is seen in training views. For visualization purpose, we normalize the heatmap by the number of the training frames N . Finally, we apply a threshold β to the heatmap and

obtain a (d) binary co-visibility mask, which we also visualize with (e) its overlay on the test image. We adopt a conservative strategy and set $\beta = \max(5, 0.1 \cdot N)$, meaning that we deem a pixel “seen” during training and valid for evaluation when it is seen in 5 or 10% of training frames, whichever is larger. This strategy ensures high recall in the masking result, i.e., the final co-visible regions are adequately seen during training when the flow estimation is noisy. For example, as shown in the third row of Figure A.1 (b), the test view occlusions are inaccurate and miss the red cover of the chicken toy when it is visible in both frames. However, since the red cover is adequately seen over the whole sequence, it is still included in the final co-visibility mask.

A.3.2 Masked image metrics

In this work, we propose to only evaluate on regions that are adequately seen during training by co-visibility masking. We employ three masked image metrics, namely mPSNR, mSSIM and mLPIPS, which extend from their original definition, which we discuss next.

PSNR \rightarrow mPSNR. PSNR is originally defined as per-pixel mean squared error (MSE) in the log scale (with a constant negative multiplier). We compute mPSNR by simply taking the average of per-pixel PSNR scores over the masked region.

SSIM [9] \rightarrow mSSIM. Comparing to PSNR, SSIM is defined on the patch level: it considers the structural similarity within each patch. In practice, it is usually implemented as convolutions where kernels are defined by the pixels in each patch. We take inspiration from Liu *et al.* [44] and follow exactly their partial convolution implementation for this operation, where only the masked pixels are accounted for the final result.

LPIPS [10] \rightarrow mLPIPS [4, 42, 43]. LPIPS is also defined on patch level. Given two images, it computes their similarity distance in the feature space across different spatial resolution using a pretrained AlexNet model [209]. The final similarity score is the average over all distance maps. To compute mLPIPS, we follow the previous works [4, 42, 43] and first apply the co-visibility mask on the input images by zeroing out the unseen regions. Given the output distance maps at each spatial resolution, we then apply the same mask with downsampling and compute the masked average distance score. It should be noted that the pretrained AlexNet has a receptive field of 195^2 . Thus when the co-visibility mask is small (most of the pixels are not seen during training), this metric can be artificially low due to the zeroing operation.

A.4 Correspondence readout from existing works

In this section, we first review the formulation of the existing works and then describe the computation to read out correspondence from these models.

A.4.1 Formulation of existing works

A neural radiance field (NeRF) [36] represents a *static* scene as a continuous volumetric field F that transforms a point’s position \mathbf{x} and auxiliary variables \mathbf{w} (e.g., view direction, latent appearance vector) to color \mathbf{c} and density σ ,

$$F : (\mathbf{x}, \mathbf{w}) \mapsto (\mathbf{c}, \sigma). \quad (\text{A.4})$$

Here we briefly review representative approaches that extend NeRFs to dynamic scenes.

Nerfies [5] and HyperNeRF [7]. Similarly to traditional non-rigid reconstruction methods that explains non-rigid scenes with a static *canonical* space and a per-frame deformation model [21], Nerfies [5] capture a non-rigid scene with one canonical NeRF F and a per-time step view-to-canonical deformation $W_{t \rightarrow c}$ that takes a point \mathbf{x} with a time-conditioned latent vector φ_t to a canonical point \mathbf{x}_c ,

$$W_{t \rightarrow c} : (\mathbf{x}, \varphi_t) \mapsto \mathbf{x}_c. \quad (\text{A.5})$$

At each time step the resulting volumetric field is $F_t = F \circ W_{t \rightarrow c}$. HyperNeRF [7] addresses topological change on top of Nerfies by outputting a two-dimensional “ambient” coordinate \mathbf{w} encoding the topological change in addition to the canonical point \mathbf{x}_c ,

$$W_{t \rightarrow c} : (\mathbf{x}, \varphi_t) \mapsto (\mathbf{x}_c, \mathbf{w}). \quad (\text{A.6})$$

These two output variables are passed to the (topologically varying) canonical space mapping F . **Time-conditioned NeRF and NSFF [4].** Another way to handle non-rigid scenes is to directly map space-time to the output color and density by a time-conditioned latent vector φ_t , which we refer to as T-NeRF:

$$F_t : (\mathbf{x}, \varphi_t) \mapsto (\mathbf{c}, \sigma). \quad (\text{A.7})$$

Note that since T-NeRF implicitly handles deformation, it is difficult to compute correspondences over time. NSFF [4] augments T-NeRF’s implicit function F_t to output an explicit scene flow field $W_{t \rightarrow t+\delta}$ between adjacent time steps t and $t + \delta$,

$$W_{t \rightarrow t+\delta} : \mathbf{x} \mapsto \mathbf{x}', \quad \delta \in \{+1, -1\}. \quad (\text{A.8})$$

This explicit flow field is used to regularize motion and, as shown below, can be chained to compute long-range point correspondences across views and times.

A.4.2 Correspondence readout

Our goal is to find view-to-view correspondences such that given a set of key-points on a source image at time t_1 , we can find their correspondence on a target image at time t_2 .

For clarity, we start with assuming a known 3D view-to-view warp $W_{t_1 \rightarrow t_2}$, outlined in the last sub-section. The 2D correspondence \mathbf{u}_{t_2} given \mathbf{u}_{t_1} can be obtained by three steps, which we describe as “warp-integrate-project”. In the “warp” step, given the pixel location \mathbf{u}_{t_1} and camera π_{t_1} , we sample points on the ray passing from the camera center through the pixel $\pi_{t_1}^{-1}(\mathbf{u}_{t_1})$. Then, we

warp the sampled points toward their 3D correspondences in the target frame using the known 3D warp $W_{t_1 \rightarrow t_2}$. In the “integrate” step, we compute the expected 3D location for the source samples weighted by the probability mass w_{t_1} by volume rendering, as per NeRF [36]. We can use densities from either source or target frame, a choice that we find insensitive in practice. In our formulation, we use the densities from the source frame. Finally, in the “project” step, we project the expected 3D location to the target frame through the target camera π_{t_2} . The “warp-integrate-project” process can be written as

$$\mathbf{u}_{t_2} = \pi_{t_2} \left(\mathbb{E}_{\mathbf{x}_{t_1} \in \pi_{t_1}^{-1}(\mathbf{u}_{t_1})} [w_{t_1}(\mathbf{x}_{t_1}) \cdot W_{t_1 \rightarrow t_2}(\mathbf{x}_{t_1})] \right). \quad (\text{A.9})$$

Note that there are also other alternatives such as “warp-project-integrate” where integration happens after projecting warped points to 2D. We find in practice that these different approaches make little difference to the final results when the surface is dense such that w_t is concentrated near one point (almost one-hot) for each ray.

Nerfies [5] and HyperNeRF [7]. We can compose $W_{t_1 \rightarrow t_2}$ by an inverse map $W_{t_1 \rightarrow c}$ and a forward map $\tilde{W}_{c \rightarrow t_2}$,

$$W_{t_1 \rightarrow t_2}(\mathbf{x}_{t_1}) = \tilde{W}_{c \rightarrow t_2}(W_{t_1 \rightarrow c}(\mathbf{x}_{t_1})). \quad (\text{A.10})$$

We solve for the forward map given the inverse map through optimization:

$$\tilde{W}_{c \rightarrow t}(\mathbf{x}_t) = \arg \min_{\mathbf{x}_c} \|W_{t \rightarrow c}(\mathbf{x}_t) - \mathbf{x}_c\|_2^2. \quad (\text{A.11})$$

We use the Broyden solver for root-finding, as per SNARF [45], and initialize \mathbf{x}_c with \mathbf{x}_t .

NSFF [4]. We can compose $W_{t_1 \rightarrow t_2}$ by chaining the scene flow predictions through time. Concretely we have

$$W_{t_1 \rightarrow t_2}(\mathbf{x}_{t_1}) = W_{t_2-1 \rightarrow t_2} \left(\cdots W_{t_1+1 \rightarrow t_1+2}(W_{t_1 \rightarrow t_1+1}(\mathbf{x}_{t_1})) \right). \quad (\text{A.12})$$

A.5 Summary of the capture setup and data processing for our iPhone dataset

Our capture setup has 7 multi-camera captures (MV) and 7 single camera captures (SV). We evaluate novel-view synthesis on the multi-camera captures and correspondence on all captures.

Multi-camera captures. For multi-camera captures, we employ three cameras: one hand-held camera to capture monocular video for training and two stationary mounted cameras for validation. The two validation cameras face inward from two distinct viewpoints with large baseline. This wide-baseline setup enables us to better evaluate the shape modeling quality for novel-view synthesis. We use the “Record3D” app [210] on iPhone to record both RGB and depth information at each time

step. Note that we only collect depth information for training views given that we will only use the depths for supervision. We discuss the preprocessing procedure for the training video sequence in the “Single-camera captures” paragraph below.

To synchronize multiple cameras, we leverage the “audio-based multi-camera synchronization” functionality in Adobe Premiere Pro, as per [211], which achieves millisecond-level accuracy. In Figure A.2, we show visualizations of our multi-camera captures after time synchronization. To ensure that our input sequence covers most of the scene regions in evaluation, we intentionally move the training camera in front of each test camera at certain frames. When we do so, that particular test frame is excluded due to severe occlusion (shown as “Excluded” in the figure). For the WHEEL sequence (last row), we only employ the right camera due to the limited physical space to set up the multi-camera rig in that scene.

After time synchronization, we calibrate the multi-camera system. The Record3D app provides camera parameters and poses at each time step, but the poses only relate to each other *within* each capture sequence. In fact, each camera pose is recorded as relative pose to the first frame in each sequence, with the first pose being identity. We therefore need to solve the relative $SE(3)$ transforms between the first frame in each test sequence with respect to the first training frame. This problem can be formulated as a Perspective-n-Point (PnP) problem where, given a set of 3D points and their corresponding 2D pixels in two sequences, we aim to solve the camera pose. In practice, given a training RGBD frame and a testing RGB frame, we compute a set of 2D correspondences by SIFT feature matching [212] and obtain their 3D point positions (in the training sequence’s world space) by back-projecting the 2D keypoints with the training frame depth map. This process is repeated for all time steps. We exploit our problem structure by constraining the camera poses within each test sequence to be the same, i.e., static camera. We use the RANSAC PnP solver [213] in OpenCV [214].

Single-camera captures. We treat the single-camera captures as the training sequence in our multi-camera capture setup. In effect, the single-camera capture setup will not have validation data for novel-view synthesis evaluation. We preprocess the depth data for the training sequence by applying a Sobel filter [215] to filter out inaccurate depth values around object edges. In Figure A.3, we visualize our depth data before and after filtering. We find that NeRF is particularly sensitive to depth noise and this filtering step is necessary. Finally, we manually annotate keypoints for correspondence evaluation. For sequences of humans and quadrupeds (dogs or cats), we annotate keypoints based on the skeleton defined in the COCO challenge [216] and StanfordExtra [217]. For sequences that focus on more general objects (e.g., our BLOCK and TEDDY sequences), we manually identify and annotate 5 to 15 trackable keypoints across frames. We visualize keypoint annotations (with skeleton if available) for both our proposed iPhone dataset and the Nerfies-HyperNeRF dataset in Figure A.4.

Note that both Nerfies [5] and HyperNeRF [7] use background regularization which requires a point cloud of the background static scene. We first extract the object mask over time by MTTR, an off-the-shelf video segmentation network [206], which takes a text prompt of the foreground object as input. Since our foreground objects are quite diverse (e.g., backpack and block), the

segmentation results are usually noisy. Thus we apply TSDF Fusion [218] to the background point clouds over the whole sequence to get a completed background point cloud. We find that this point cloud can be noisy when segmentation fails, and that it is necessary to manually filter the background point cloud to make sure that it does not include any foreground regions. We consider this manual process a weakness of the previous background regularization [5].

A.6 Summary of the implementation details and remaining differences

To ensure a fair comparison, we align numerous training details between the models that we investigate in this paper: T-NeRF, NSFF [4], Nerfies [5] and HyperNeRF [7]. Code and checkpoints are available on our [project page](#).

To start with, we align the total number of rays seen during training. We add support of ray undistortion [7] in the third-party implementation of NSFF [48] to make sure that the training rays are the same across codebases. All models are trained with view-dependency modeling turned on. We did not find appearance encoding [219] helpful in terms of quantitative results. This might due to the lighting difference between training and validation captures – a common issue in evaluation discussed in mip-NeRF 360 [49].

T-NeRF, Nerfies, and HyperNeRF share the exact same training setup since they are implemented within our codebase. We follow the hyper-parameters specified in their official repositories. We use a batch size $B = 6144$ for a total number of iterations $N = 2.5 \times 10^5$, optimized by ADAM [220] with an initial learning rate $\eta = 1 \times 10^{-3}$ exponentially decayed to 1×10^{-4} at the end. We use this training recipe for all of our experiments across all datasets. On 4 NVIDIA RTX A4000 or 2 NVIDIA A100 GPUs with 24GB memory, it takes roughly 12 hours to train a T-NeRF and 24 hours to train a Nerfies or a HyperNeRF. In Table A.2, we show that our codebase reproduces the numbers from the original papers and official repositories.

Due to no publicly available code to train NSFF on the Nerfies-HyperNeRF dataset. We adapt and extend the third-party implementation of NSFF (which we find to perform better than the official repo [47]). We confirm the finding from HyperNeRF that the default hyper-parameters in the NSFF paper are not suitable for long video sequences, and use their hyper-parameters instead. In Table A.3, we check on one sequence that our modified re-implementation of NSFF can reproduce the numbers from the ones we obtain by running the released code. On 1 NVIDIA RTX A4000 or NVIDIA A100 GPU, it takes roughly 72 to train a NSFF. With better implementation, we hypothesize that the training process can be largely accelerated.

While we try to ensure the fairness in our comparison, there are still four main remaining differences, namely: (1) static scene stabilization, (2) sampling and rendering, (3) NeRF coordinates, and (4) flow supervision. First, Nerfies and HyperNeRF use additional background points from SfM system as supervision to stabilize the static region of the scene, which we find sensitive to foreground segmentation errors as mentioned in Section A.5. On the other hand, NSFF stabilizes the static region by composing the samples from a time-invariant static NeRF and a time-varying dynamic NeRF. Sec-

Method	BROOM $\Omega = 3.40, \omega = 128.54$			CURLS $\Omega = 1.20, \omega = 138.55$			TAIL $\Omega = 3.30, \omega = 160.55$			TOBY SIT $\Omega = 2.18, \omega = 110.51$		
	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow
Nerfies [5]	19.40	-	0.323	-	-	-	-	-	-	-	-	-
Nerfies (repo)	19.40	-	0.325	24.40	-	0.392	21.90	-	0.245	18.44	-	0.384
Nerfies (our reimpl.)	19.70	0.216	0.296	24.04	0.670	0.245	21.79	0.314	0.236	18.48	0.355	0.375
HyperNeRF [7]	19.30	-	0.296	-	-	-	-	-	-	-	-	-
HyperNeRF (repo)	19.30	-	0.308	24.60	-	0.363	22.10	-	0.226	18.40	-	0.330
HyperNeRF (our reimpl.)	19.36	0.210	0.314	24.59	0.686	0.247	22.16	0.329	0.231	18.41	0.345	0.339

Method	3D PRINTER $\Omega = 3.13, \omega = 251.37$			CHICKEN $\Omega = 7.38, \omega = 212.58$			PEEL BANANA $\Omega = 1.26, \omega = 237.66$		
	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow
Nerfies [5]	20.20	-	0.115	26.00	-	0.084	21.70	-	0.157
Nerfies (repo)	20.20	-	0.118	26.80	-	0.081	22.00	-	0.179
Nerfies (our reimpl.)	20.30	0.639	0.115	26.54	0.823	0.079	21.11	0.693	0.174
HyperNeRF [7]	20.00	-	0.111	26.90	-	0.079	23.30	-	0.133
HyperNeRF (repo)	20.10	-	0.110	27.70	-	0.076	22.20	-	0.140
HyperNeRF (our reimpl.)	20.12	0.638	0.110	27.74	0.834	0.077	22.25	0.729	0.144

Table A.2: **Our re-implementation reproduces Nerfies [5]’s and HyperNeRF [7]’s results.** The official numbers for both Nerfies and HyperNeRF are taken from the HyperNeRF paper. Our results matches closely to their numbers and the ones that we obtained by running the officially released repositories (denoted as “repo”). All models are trained under the teleporting setting.

Method	JUMPING $\Omega = 0.68, \omega = 53.79$		
	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow
NSFF (repo)	27.41	0.900	0.057
NSFF (our reimpl.)	27.80	0.908	0.051

Table A.3: **Our modified third-party re-implementation reproduces NSFF [4]’s results on one sequence from Yoon *et al.* [40].** Due to the absence of per-sequence results in the original paper, we compare to the numbers that we obtained by evaluating the officially released checkpoints (denoted as “repo”). Our results matches closely to their numbers. All models are trained under the teleporting setting.

ond, Nerfies and HyperNeRF sample $S = 128$ points during the coarse stage, and another $2S$ points during the fine stage, evaluating $3S = 384$ points in total. NSFF, on the other hand, only samples S points for dynamic NeRF and another S points for static NeRF, without coarse-to-fine sampling, evaluating $2S = 256$ points in total. Third, Nerfies and HyperNeRF sample points in world space, while NSFF samples in normalized device coordinates (NDC), which can cause issues when applying to non-forward-facing scenes like the ones we use in this paper. Finally, NSFF uses additional optical

flow supervision, while Nerfies and HyperNeRF do not. In fact, we consider the fact that NSFF can leverage correspondence supervision as a merit in the sense that it is non-trivial to apply optical flow supervision to Nerfies and HyperNeRF since their warp representation is not fully invertible.

A.7 Additional results on the impact of effective multi-view

In Figure A.5, we provide more qualitative comparisons between models that are trained with and without camera teleportation on the Nerfies-HyperNeRF dataset.

A.8 Additional results on per-sequence quantitative performance breakdown

We document the per-sequence quantitative performances of different models on both the Nerfies-HyperNeRF dataset (under non-teleporting setting) in Table A.4 and the proposed iPhone dataset in Table A.5.

A.9 Additional results on novel-view synthesis

We provide additional novel-view synthesis qualitative results under the non-teleporting setting. In Figure A.6, we show qualitative results on the Nerfies-HyperNeRF dataset. In Figure A.7, we show qualitative results on the multi-camera captures from the proposed iPhone dataset. All models except NSFF [4] are trained with all the additional regularizations that we find helpful through ablation, denoted with “++” to distinguish with the original models. In Figure A.8, we show qualitative results on the single-camera captures from the proposed iPhone dataset. We render novel views using the camera pose from the first captured frame. Finally, in Figure A.9, we show the rendering results with and without co-visibility mask applied.

A.10 Additional results on inferred correspondence

In Table A.6, we provide additional quantitative results of the inferred correspondence on the single-camera captures from the proposed iPhone dataset. In Figure A.10 and A.11, we provide additional qualitative results of the inferred correspondence on both the Nerfies-iPhone dataset and the proposed iPhone dataset. Note that all models are trained with additional regularizations on the proposed iPhone dataset except NSFF.

BROOM ($\Omega = 2.57, \omega = 60.4$)					CURLS ($\Omega = 0.90, \omega = 118.7$)			
Method	mPSNR \uparrow	mSSIM \uparrow	mLPIPS \downarrow	PCK-T \uparrow	mPSNR \uparrow	mSSIM \uparrow	mLPIPS \downarrow	PCK-T \uparrow
T-NeRF	20.04(20.17)	0.344 (0.257)	0.590(0.624)	-	21.86(21.75)	0.677(0.597)	0.284(0.341)	-
NSFF	20.36 (20.46)	0.335(0.247)	0.776 (0.813)	0.119	18.74(18.85)	0.616(0.531)	0.378 (0.423)	0.212
Nerfies	19.34(19.51)	0.293(0.202)	0.294(0.327)	0.460	23.28 (23.03)	0.707 (0.630)	0.220(0.266)	0.782
HyperNeRF	19.04(19.23)	0.288(0.197)	0.279(0.313)	0.471	23.13(22.98)	0.700(0.625)	0.220(0.266)	0.838

TAIL ($\Omega = 1.31, \omega = 28.6$)					TOBY-SIT ($\Omega = 1.28, \omega = 26.4$)			
Method	mPSNR \uparrow	mSSIM \uparrow	mLPIPS \downarrow	PCK-T \uparrow	mPSNR \uparrow	mSSIM \uparrow	mLPIPS \downarrow	PCK-T \uparrow
T-NeRF	22.56 (22.11)	0.460(0.385)	0.305(0.365)	-	18.53(18.53)	0.428(0.330)	0.421(0.471)	-
NSFF	21.94(21.72)	0.461 (0.388)	0.522 (0.579)	0.323	18.66 (18.65)	0.429 (0.329)	0.600 (0.634)	0.666
Nerfies	21.46(21.17)	0.385(0.305)	0.213(0.261)	0.645	18.45(18.41)	0.423(0.326)	0.249(0.307)	0.914
HyperNeRF	21.54(21.13)	0.382(0.301)	0.218(0.263)	0.623	18.40(18.33)	0.422(0.324)	0.242(0.300)	0.883

3DPRINTER ($\Omega = 1.22, \omega = 59.4$)					CHICKEN ($\Omega = 1.52, \omega = 33.5$)			
Method	mPSNR \uparrow	mSSIM \uparrow	mLPIPS \downarrow	PCK-T \uparrow	mPSNR \uparrow	mSSIM \uparrow	mLPIPS \downarrow	PCK-T \uparrow
T-NeRF	19.69 (18.60)	0.665 (0.591)	0.205(0.238)	-	25.54 (24.41)	0.802 (0.764)	0.131(0.158)	-
NSFF	16.89(16.26)	0.526(0.426)	0.443 (0.492)	0.797	21.47(20.72)	0.671(0.619)	0.290 (0.325)	0.604
Nerfies	19.67(18.81)	0.661(0.588)	0.148(0.175)	0.998	23.78(22.71)	0.784(0.742)	0.114(0.142)	0.978
HyperNeRF	19.58(18.73)	0.656(0.583)	0.147(0.175)	0.994	24.90(23.88)	0.792(0.753)	0.101(0.125)	1.000

PEEL-BANANA ($\Omega = 0.33, \omega = 33.8$)				
Method	mPSNR \uparrow	mSSIM \uparrow	mLPIPS \downarrow	PCK-T \uparrow
T-NeRF	22.64 (22.07)	0.787 (0.721)	0.142(0.185)	-
NSFF	18.68(18.62)	0.613(0.530)	0.293 (0.335)	0.233
Nerfies	19.97(19.85)	0.677(0.609)	0.161(0.206)	0.514
HyperNeRF	21.34(21.08)	0.707(0.641)	0.135(0.173)	0.540

Table A.4: **Per-scene breakdowns of the quantitative results on the Nerfies-HyperNeRF dataset.** Numbers in gray are calculated without using the co-visibility mask. All models are trained under the non-teleporting setting.



Figure A.2: **Visualizations of the multi-camera captures after time synchronization from the proposed iPhone dataset.** In each row, we visualize the frames from both the training camera and two testing cameras at two time steps. We intentionally move the training camera in front of each test camera at certain times to ensure that our input sequence covers most of the scene in evaluation. When a particular test frame depicts the training camera, we exclude the test frame (denoted as “Excluded”). For the WHEEL sequence in the last row, we only employ the right test camera due to limited space to set up the multi-camera rig.

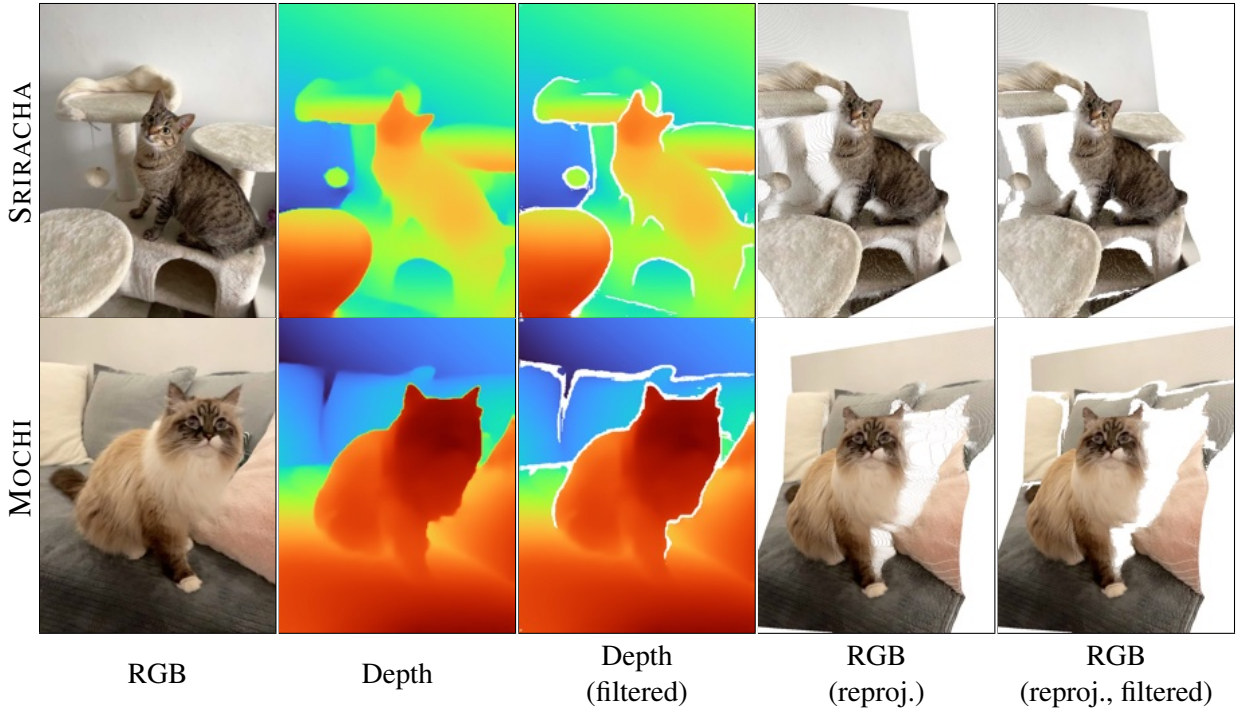


Figure A.3: **Visualizations of the depth filtering during data preprocessing of the proposed iPhone dataset.** The depth sensing is particularly noisy around the object edges, which we filtered out by Sobel filter [215]. We visualize the re-projected RGB image with the original (2nd column) or filtered depth (3rd column) from the captured view to the first view in each sequence at the last two columns. Without filtering (4th column), there are erroneous floaters which cause too much noise for training supervision. With filtering (last column), we have a crisper depth map which is used for improving the state-of-the-art methods.



Figure A.4: **Visualizations of the keypoint annotation during data preprocessing of the proposed iPhone dataset.** We manually annotate keypoints for correspondence evaluation. For sequences of humans and quadrupeds (dogs or cats), we annotate based on the skeleton defined in the COCO challenge [216] and StanfordExtra [217]. For sequences that focus on more general objects, we manually identify and annotate 5 to 15 trackable keypoints across frames.

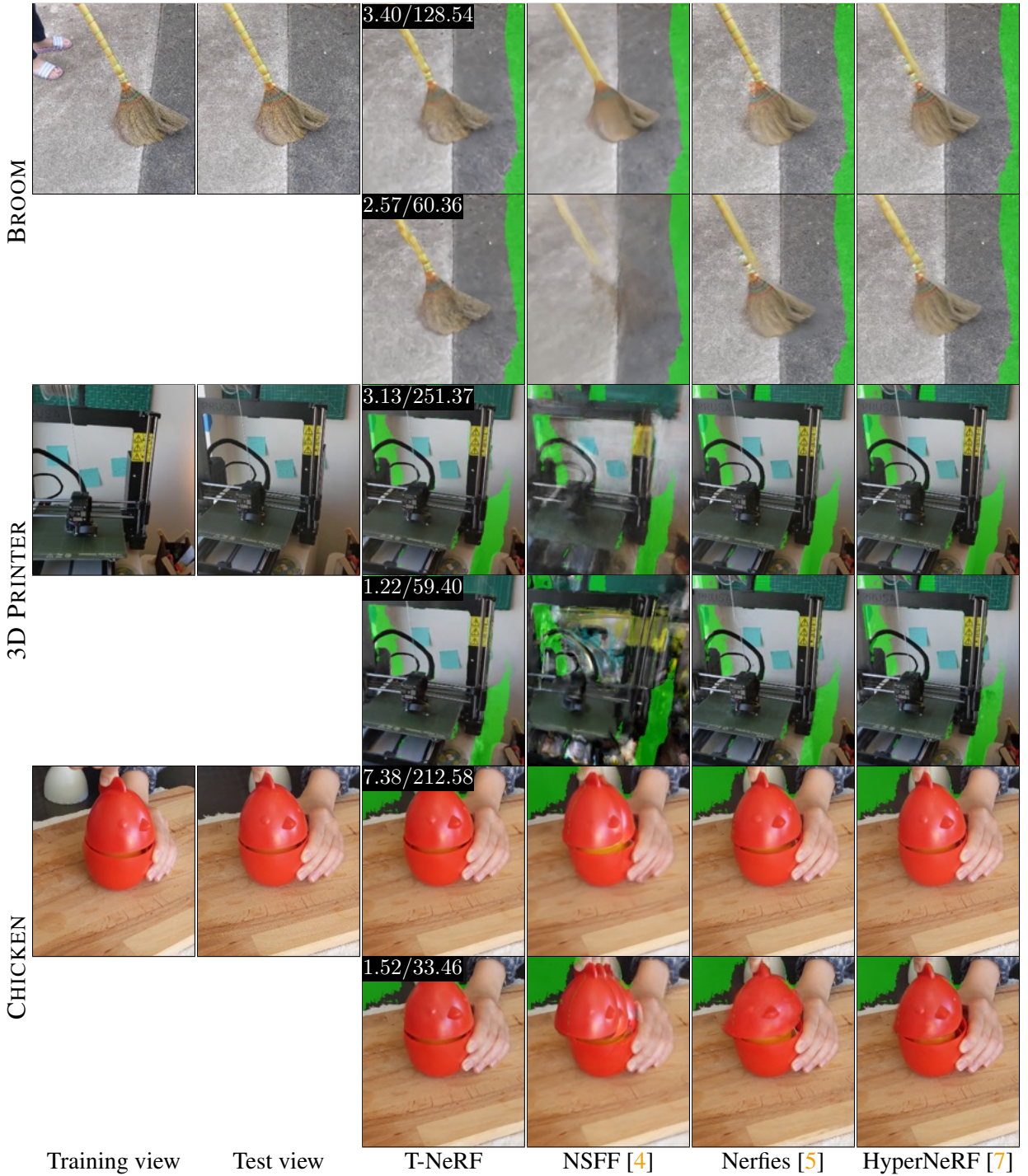


Figure A.5: **Additional qualitative results on the impact of effective multi-view on the Nerfies-HyperNeRF dataset.** Ω/ω metrics of the input sequence are shown on the top-left. We compare the existing camera teleporting setting and our non-teleporting setting. For every two rows, we show the results trained with and without camera teleportation in the first and second rows. Two settings use the same set of co-visibility masks computed from common training images.

APPLE ($\Omega = 0.75, \omega = 3.8$)					BLOCK ($\Omega = 0.04, \omega = 11.5$)			
Method	mPSNR \uparrow	mSSIM \uparrow	mLPIPS \downarrow	PCK-T \uparrow	mPSNR \uparrow	mSSIM \uparrow	mLPIPS \downarrow	PCK-T \uparrow
T-NeRF	17.43(15.98)	0.728(0.375)	0.508 (0.598)	-	17.52(17.15)	0.669(0.521)	0.346(0.449)	-
NSFF	17.54(16.50)	0.750(0.432)	0.478(0.548)	0.599	16.61(16.34)	0.639(0.494)	0.389(0.482)	0.274
Nerfies	17.64 (16.34)	0.743(0.411)	0.478(0.563)	0.318	17.54 (17.35)	0.670 (0.528)	0.331(0.424)	0.216
HyperNeRF	16.47(16.07)	0.754 (0.425)	0.414(0.505)	0.132	14.71(14.93)	0.606(0.460)	0.438 (0.517)	0.180

PAPER-WINDMILL ($\Omega = 0.38, \omega = 10.7$)					SPACE-OUT ($\Omega = 0.13, \omega = 6.4$)			
Method	mPSNR \uparrow	mSSIM \uparrow	mLPIPS \downarrow	PCK-T \uparrow	mPSNR \uparrow	mSSIM \uparrow	mLPIPS \downarrow	PCK-T \uparrow
T-NeRF	17.55 (17.55)	0.367(0.349)	0.258(0.268)	-	17.71(17.04)	0.591(0.521)	0.377 (0.438)	-
NSFF	17.34(17.35)	0.378(0.362)	0.211(0.218)	0.113	17.79(17.25)	0.622(0.560)	0.303(0.359)	0.812
Nerfies	17.38(17.39)	0.382 (0.366)	0.209(0.215)	0.107	17.93 (18.10)	0.605(0.546)	0.320(0.369)	0.859
HyperNeRF	14.94(14.98)	0.272(0.254)	0.348 (0.361)	0.163	17.65(17.79)	0.636 (0.578)	0.341(0.390)	0.598

SPIN ($\Omega = 0.15, \omega = 7.9$)					TEDDY ($\Omega = 0.20, \omega = 7.6$)			
Method	mPSNR \uparrow	mSSIM \uparrow	mLPIPS \downarrow	PCK-T \uparrow	mPSNR \uparrow	mSSIM \uparrow	mLPIPS \downarrow	PCK-T \uparrow
T-NeRF	19.16(18.17)	0.567(0.441)	0.443 (0.490)	-	13.71(13.32)	0.570 (0.331)	0.429(0.565)	-
NSFF	18.38(16.97)	0.585 (0.445)	0.309(0.380)	0.177	13.65(12.91)	0.557(0.302)	0.372(0.508)	0.801
Nerfies	19.20 (18.59)	0.561(0.436)	0.325(0.377)	0.115	13.97 (13.91)	0.568(0.327)	0.350(0.479)	0.775
HyperNeRF	17.26(16.52)	0.540(0.414)	0.371(0.437)	0.083	12.59(12.78)	0.537(0.304)	0.527 (0.635)	0.291

WHEEL ($\Omega = 0.03, \omega = 58.5$)				
Method	mPSNR \uparrow	mSSIM \uparrow	mLPIPS \downarrow	PCK-T \uparrow
T-NeRF	15.65 (14.42)	0.548 (0.405)	0.292(0.363)	-
NSFF	13.82(13.19)	0.458(0.312)	0.310(0.366)	0.394
Nerfies	13.99(13.35)	0.455(0.307)	0.310(0.366)	0.408
HyperNeRF	14.59(13.31)	0.511(0.359)	0.331 (0.402)	0.346

Table A.5: **Per-scene breakdowns of the quantitative results on the proposed iPhone dataset.** Numbers in gray are calculated without using the co-visibility mask.

Method	CREEPER	BACKPACK	HANDWAVY	HARU	MOCHI	PILLOW	SRIRACHA	MEAN
NSFF [4]	0.560	0.269	0.178	0.699	0.624	0.154	0.616	0.443
Nerfies++	0.708	0.329	0.685	0.942	0.908	0.575	0.737	0.698
HyperNeRF++	0.702	0.260	0.708	0.817	0.891	0.602	0.617	0.657

Table A.6: **Additional quantitative results of the PCK-T evaluation on the single-camera captures from the proposed iPhone dataset.** The correspondence evaluation is applicable when multi-camera validation is not available. All numbers are computed with $\alpha = 0.05$.

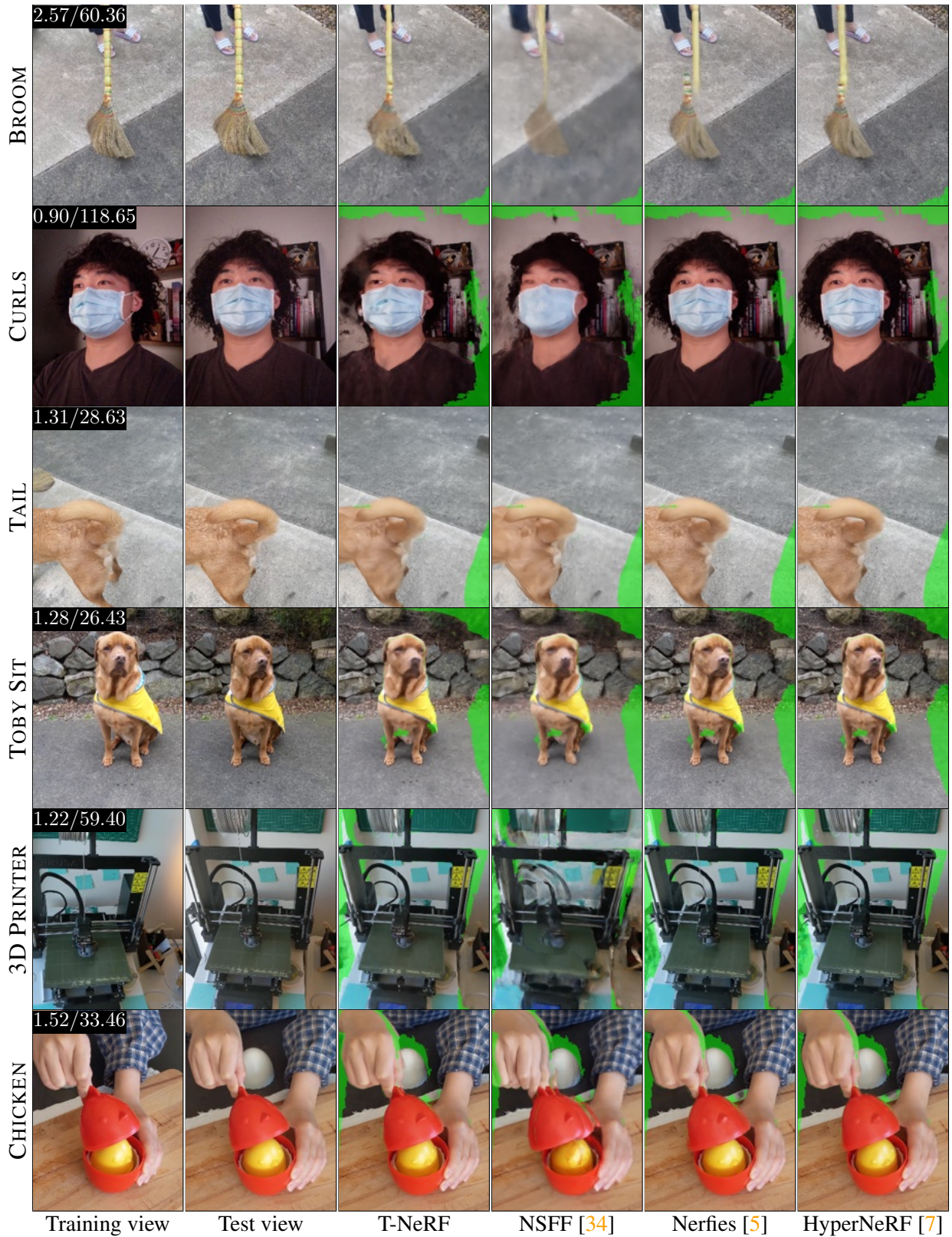


Figure A.6: **Additional qualitative results on the Nerfies-HyperNeRF dataset without camera teleportation.** Ω/ω metrics of the input sequence are shown on the top-left.

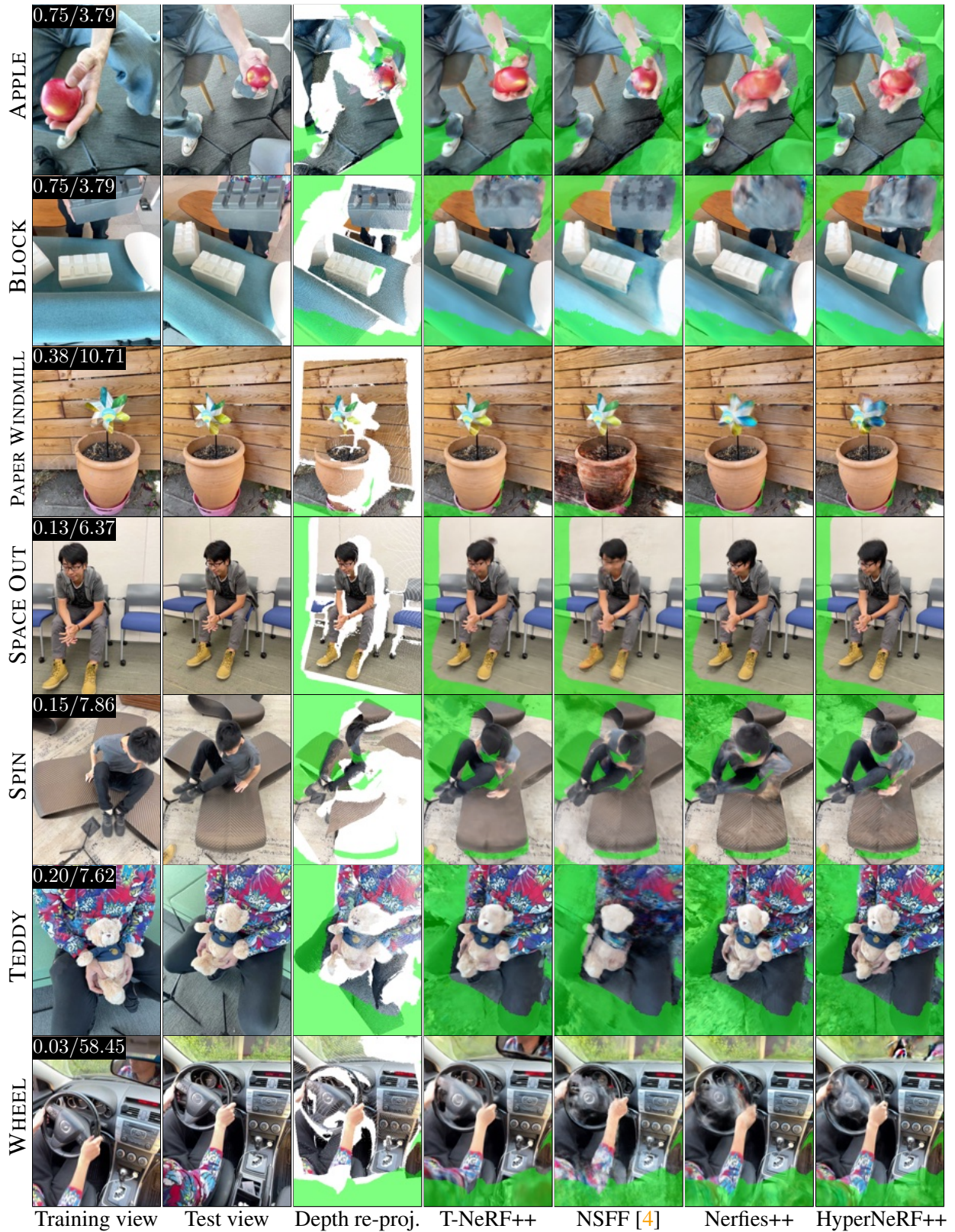


Figure A.7: **Additional qualitative results on the multi-camera captures from the proposed iPhone dataset.** Ω/ω metrics of the input sequence are shown on the top-left. The models shown here are trained with all the additional regularizations (+B+D+S) except NSFF.

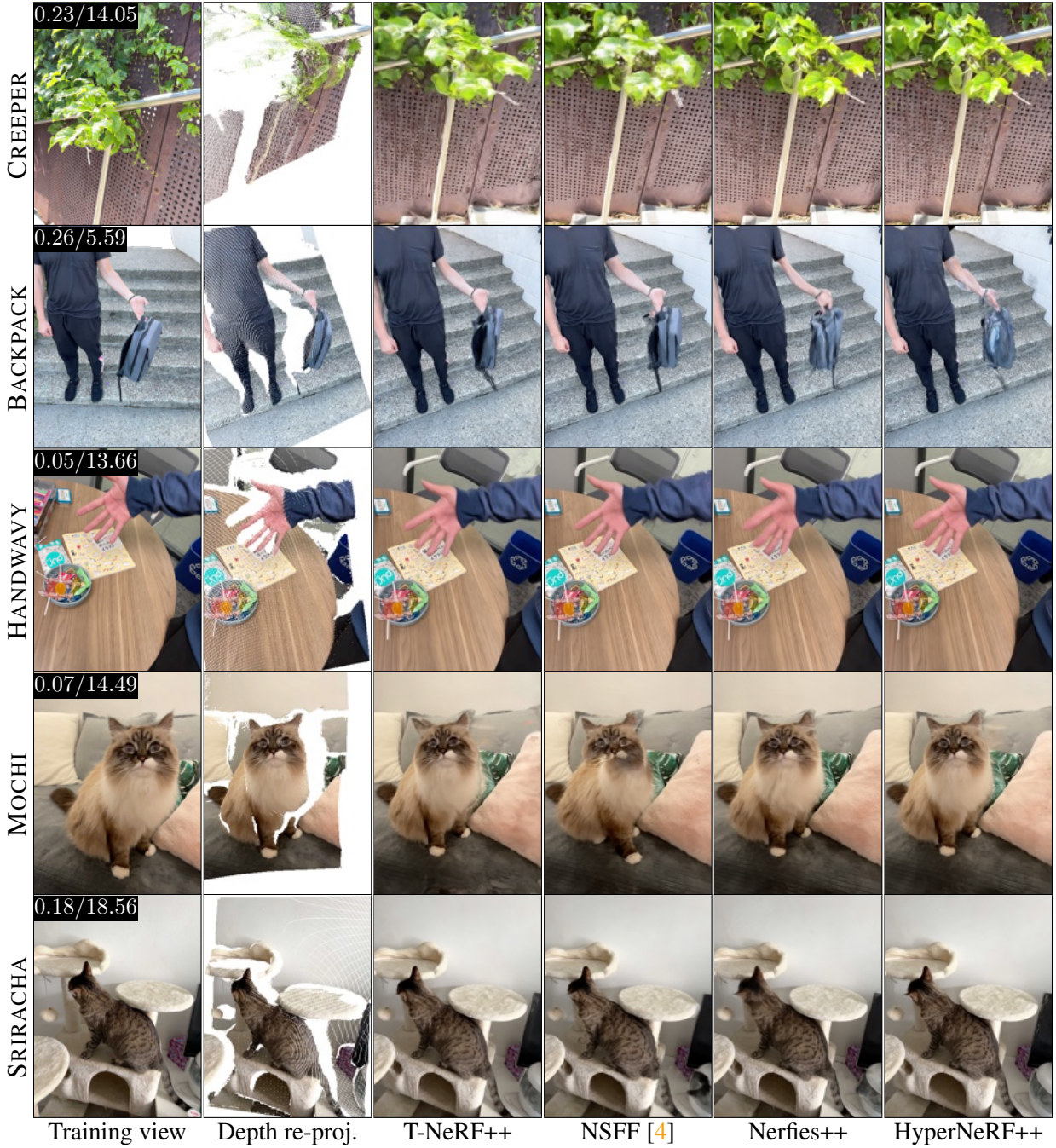


Figure A.8: **Additional qualitative results on the single-camera captures from the proposed iPhone dataset.** Ω/ω metrics of the input sequence are shown on the top-left. The models shown here are trained with all the additional regularizations (+B+D+S) except NSFF. We re-render the scene from the first viewpoint in each sequence. Note that there are no ground-truth validation frames.

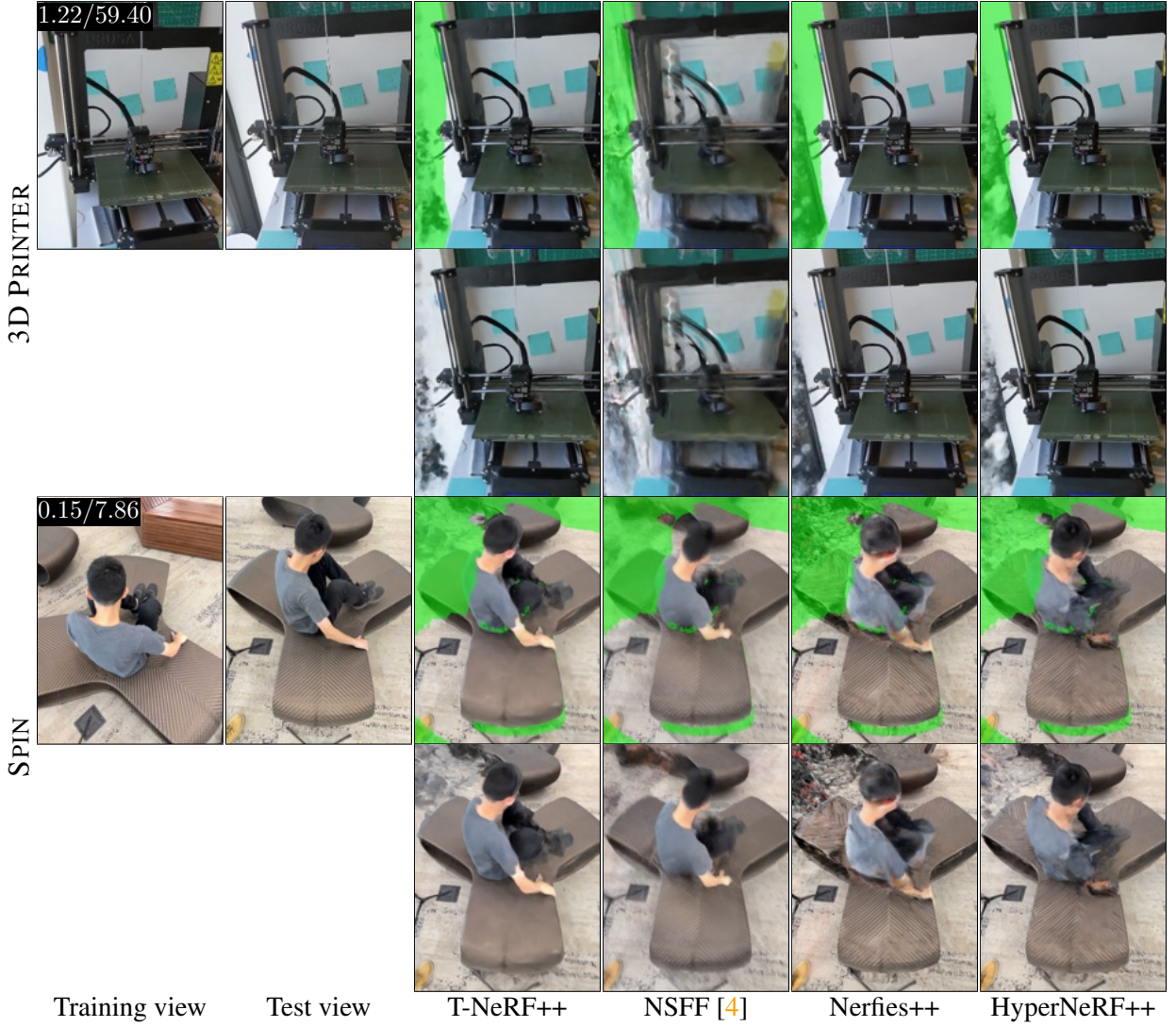


Figure A.9: **Additional qualitative results on the full image rendering on both the Nerfies-HyperNerF dataset and the proposed iPhone dataset.** Ω/ω metrics of the input sequence are shown on the top-left. All models are trained under non-teleporting setting. For every two rows, we show the results with and without applying co-visibility mask. All models are not able to reconstruct the unseen regions.

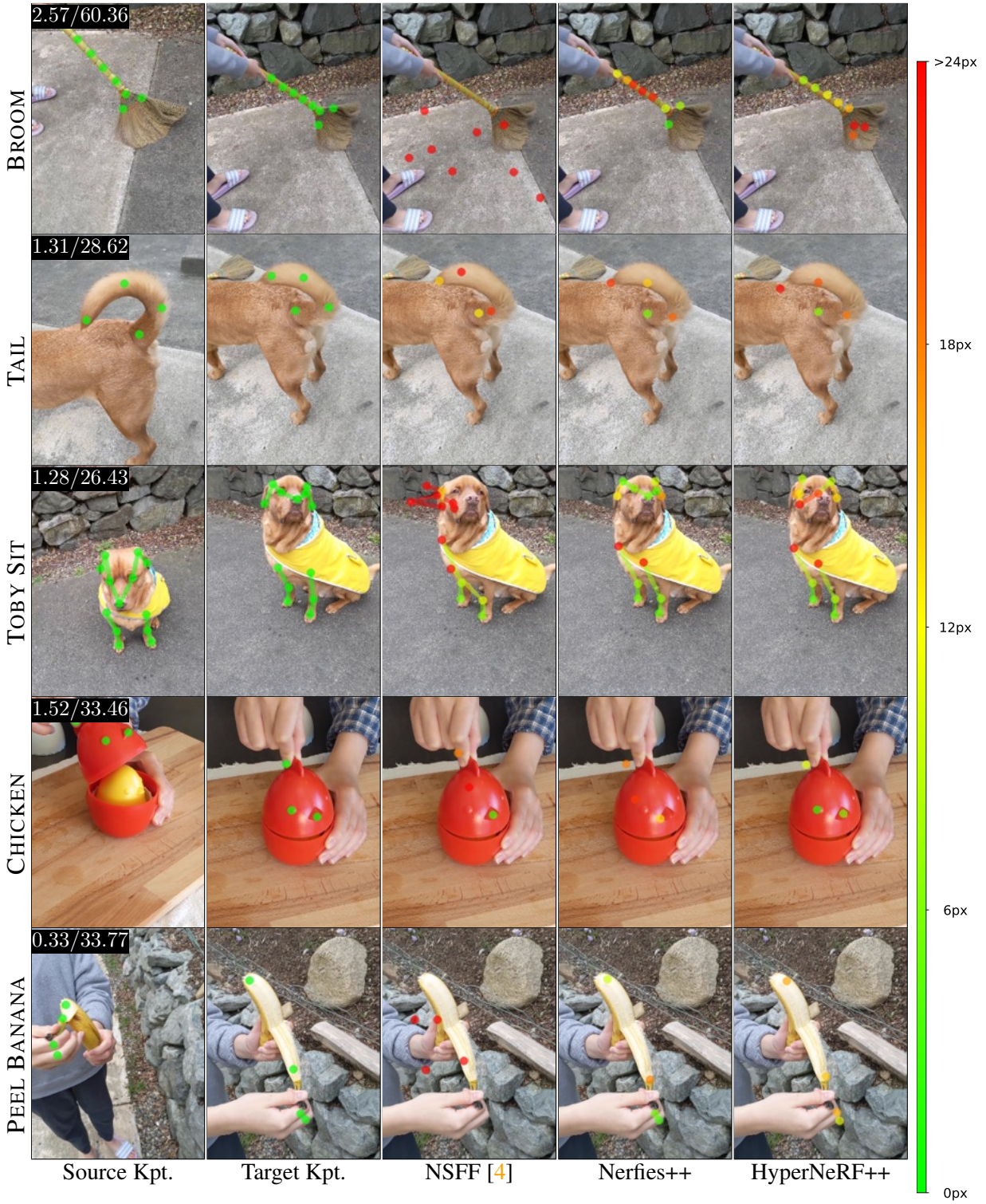


Figure A.10: **Additional qualitative results of keypoint transferring on the Nerfies-HyperNeRF dataset without camera teleportation.** Ω/ω metrics of the input sequence are shown on the top-left. All models are trained under non-teleporting setting. Transferred keypoints are colorized by a heatmap of end-point error, overlaid on the ground-truth target frame.

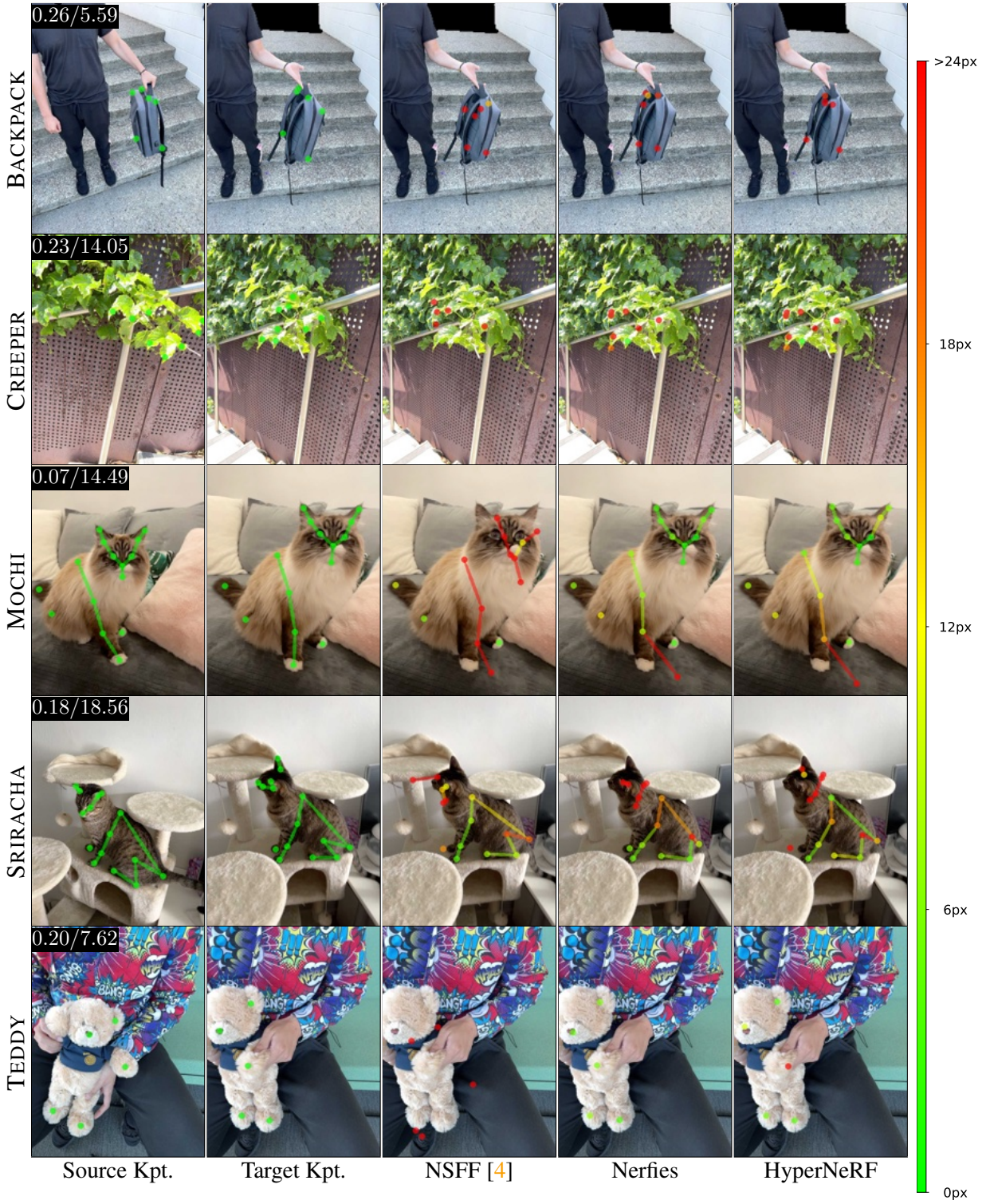


Figure A.11: **Additional qualitative results of keypoint transferring on the proposed iPhone dataset.** Ω/ω metrics of the input sequence are shown on the top-left. All models are trained under non-teleporting setting. Transferred keypoints are colorized by a heatmap of end-point error, overlaid on the ground-truth target frame.

Appendix B

Shape of Motion: 4D reconstruction from a single video

B.1 Outline

In this Appendix, we describe in detail the following:

- Additional preprocessing details in Section B.2.
- Additional training details in Section B.3.
- Initialization details in Section B.3.1.
- Training details including optimization and loss functions in Section B.3.2.
- Additional evaluation details in Section B.4.
- Visualization of Kubric experiment in Section B.5.
- NVIDIA dataset evaluation in Section B.6.

B.2 Additional preprocessing details

Obtaining camera poses. Our method takes video sequences with known camera poses as input. To obtain camera poses for in-the-wild videos with moving objects, we adopt one of these two approaches depending on the type of input camera motion: (1) if there is sufficient camera motion parallax, we use COLMAP [148]’s SfM pipeline to obtain the camera poses and the sparse point clouds for the static regions, where we exclude keypoints in the foreground masks during the feature extraction stage. The foreground masks are generated using Track-Anything [141], a flexible and interactive tool for video object tracking and segmentation. The static point clouds produced by COLMAP [148] can then be used for aligning the affine-invariant monocular depth maps from Depth Anything [138]. (2) If the video is captured by a roughly stationary camera (small interframe camera baseline), COLMAP tends to fail catastrophically. Initial results on DAVIS use camera poses estimated with Unidepths [142] and DroidSLAM [143]. Specifically we first employ Unidepths to

predict metric depth and camera intrinsics. Using these predictions, we then estimate camera poses with DroidSLAM.

Aligning monocular depth maps. The disparity output from Depth Anything [138] model is affine-invariant, so we need to align them with the cameras and reconstruction derived from COLMAP or DroidSLAM. To do so, we solve for a per-frame global scale and shift parameters that minimizes ℓ_2 distance between the monocular disparity from Depth Anything and the disparity derived from SfM sparse point clouds or depth outputs from DroidSLAM.

Computing long-range 2D tracks. We utilize TAPIR [97] to compute long-range 2D tracks for a video. While the ideal scenario would involve computing full-length tracks for every pixel in every frame (i.e., exhaustive pairs of correspondences), this approach is prohibitively computationally expensive. We therefore only compute full-length tracks for pixels located on a grid for foreground moving objects in each frame. In our experiment, we set grid interval to 4 (i.e., sampling a query point every 4 pixels). Due to our low-dimensional motion representation, we observe that our method can effectively operate with semi-dense 2D tracks without significant performance degradation. During training, we filter out correspondences that TAPIR predicts with high uncertainty or those that are occluded.

B.3 Additional training details

B.3.1 Initialization details

During initialization stage, we first solve a Procrustes alignment problem for each cluster b , where we estimate $\text{SE}(3)$ transformation between point sets $\{\mathbf{X}_0\}_b$ and $\{\mathbf{X}_\tau\}_b$ for all $\tau = 0, \dots, T$. We exclude point pair when either one of them is occluded, and weight the point pair using the uncertainty score predicted by TAPIR [97] when solving Procrustes. This process produces the initialization for the set of basis functions $\{\mathbf{T}_{0 \rightarrow t}^{(b)}\}_{b=1}^B$. To initialize the motion coefficient $\mathbf{w}^{(b)}$ for each track, we compute the distance between the 3D location of the track in the canonical frame and the 3D location of each cluster center, and initialize each of the corresponding motion coefficient value to be exponentially decay with the distance.

We then optimize the $\boldsymbol{\mu}_0$, $\mathbf{w}^{(b)}$, and set of basis functions $\{\mathbf{T}_{0 \rightarrow t}^{(b)}\}_{b=1}^B$ to fit the observed 3D tracks lifted by monocular depths and 2D tracks. Specifically, we enforce an ℓ_1 loss between each of our predicted 3D track and its corresponding observed 3D track. In addition, we enforce the motion bases to be temporally smooth by adding an ℓ_2 regularization on the acceleration of both the quaternion and the translation vector. We optimize the parameters using Adam [220] optimizer for 2k steps, where the initial learning rates for $\boldsymbol{\mu}_0$, $\mathbf{w}^{(b)}$, and $\{\mathbf{T}_{0 \rightarrow t}^{(b)}\}_{b=1}^B$ are 1×10^{-3} , 1×10^{-2} and 1×10^{-2} , respectively. All learning rates are exponentially decayed to $\frac{1}{10}$ of their initial values during the optimization process.

B.3.2 Training details

Gaussian initialization. The aforementioned initialization stage gives the initialization of the mean of each Gaussian in the canonical frame. We follow the original 3D-GS [123] paper to initialize the scale s , rotation \mathbf{R}_0 , and opacity o of each Gaussian in the canonical frame. The color c of each Gaussian is initialized as the pixel color at the projected location in the canonical frame.

Optimization. We use Adam [220] Optimizer to optimize all scene parameters. The learning rates for Gaussian’s canonical mean μ_0 , opacity o , scale s , rotation \mathbf{R}_0 (parameterized as quaternion) and color c are set to 1.6×10^{-4} , 1×10^{-2} , 5×10^{-3} , 1×10^{-3} , and 1×10^{-2} , respectively. The learning rates for the $\mathbb{SE}(3)$ motion bases and the motion coefficients are set to 1.6×10^{-4} and 1×10^{-2} respectively. During each training iteration, we randomly select a batch of 8 query frames. For each query frame, we render the color, mask, depth, and the 3D track locations for 4 randomly selected target frames.

Loss weights and details. Our first set of loss functions enforces our rendered results to match the per-frame pixelwise color, depth, and masks inputs. The coefficients for depth loss λ_{depth} and mask loss λ_{mask} are set to 0.5 and 1.0, respectively. We additionally add regularization to the estimated surface geometry via a depth gradient loss and per-Gaussian scale penalty. In particular, We add a pixelwise ℓ_1 loss of spatial gradient between the depth renderings $\hat{\mathbf{D}}_t$ and corresponding reference depth maps \mathbf{D} , with a weight set to 1.0. In addition, we enforce the foreground Gaussian to be isotropic by incorporating a regularization term that penalizes standard deviations of the scale s along all three axes.

Our second set of losses supervise the motion of the Gaussian. The weights for the 2D tracking loss $\lambda_{\text{track-2d}}$ and the track depth loss $\lambda_{\text{track-depth}}$ are set to 2.0 and 0.1, respectively. The $L_{\text{track-2d}}$ is applied on normalized pixel coordinates (i.e., divided by the maximum image edge length). For the rigidity loss L_{rigidity} , we use foreground part masks from SAM automatic segmentation independently on each frame. For each training iteration, we sample 4 part masks for each query frame. For each mask, we sample 32 center points and find their 16 nearest neighbors within the mask. We compute the 3D track locations for all point samples in 4 target frames, and regularize the distances between the center points and their neighbors in the target frames to be similar to their distances in the query frame. We let $\lambda_{\text{rigidity}} = 0.1$. We weight the neighbor distances with an exponential kernel $\exp(-\beta\|x - \bar{x}\|)$ with $\beta = 2$, where \bar{x} is the center point and x is one of its neighbors. We additionally add motion smoothness regularization that enforces an ℓ_2 loss on acceleration of the motion translation bases and motion quaternion bases, with a weight set to 0.1 and an ℓ_2 loss on the acceleration along z -axis (in the camera frame) of the μ_t , both through finite difference approximation.

Training with 2D Gaussian Splatting (2DGS). To enhance scene geometry estimation, we replace 3D Gaussian Splatting with 2D Gaussian Splatting [147]. Specifically, we employ an off-the-shelf monocular normal estimator [221] to generate monocular normal maps, \tilde{N} , which are then used to supervise both the rendered normals and the normals derived from the rendered depth. The

supervision is achieved through the following losses:

$$L_n = \sum_i \omega_i (1 - n_i^T \tilde{N}) \quad (\text{B.1})$$

$$L_N = 1 - N^T \tilde{N} \quad (\text{B.2})$$

where the summation is over the splats intersect the current ray, ω_i represents the blending weight for i -th splat, and n_i denotes the normal of i -th splat. Here, N represents the normal derived from rendered depth map. This supervision ensures that the orientation and depth of the 2D splats are aligned with the monocular normal prediction.

B.4 Additional evaluation details

In our evaluation, since the synthetic Kubric dataset [146] comes with groundtruth camera poses, we directly use the groundtruth camera poses for our experiments. For iPhone dataset [136], we observed that the provided camera poses from ARKit are not accurate, we thus perform an additional global bundle adjustment using COLMAP [148] to refine the camera poses while fixing the camera intrinsics. To maintain metric scale after refinement, we compute a global $\text{SIM}(3)$ transformation for each scene to align the refined camera poses with the original metric-scale camera poses. This allows us to evaluate 3D tracking performance in metric scale.

B.5 Visualization of Kubric experiment

We find qualitatively that the optimized motion coefficients of the scene representation are coherently grouped with each moving object in the scene. We demonstrate this in Figure 3.6, where we show the first 3 PCA components of our optimized motion coefficients of evaluation scenes.

B.6 NVIDIA dataset evaluation

We conduct experiments on seven scenes from the NVIDIA Dynamic Scenes dataset [40], following the evaluation protocol of Dynamic Gaussian Marbles [151]. Specifically, we use video footage from the static camera 4 for training and employ videos from cameras 3, 5, and 6 for evaluation. For consistency with Dynamic Gaussian Marbles [151]’s experiments, we use images at half-resolution for all seven experiments, though our method is compatible with high-resolution images as well. Camera poses are estimated using COLMAP [148], and depths are predicted with Depth Anything [138] and subsequently aligned with the COLMAP point cloud. The quantitative results reported in this paper differ from those in DGM [151]. Specifically, we compute covisibility masks between training and test views and apply them during evaluation, whereas DGM [151] inpaints unobserved regions in test views for evaluation. This difference in evaluation procedure accounts for the variation in reported performance. We will make the covisibility masks publicly available to facilitate future research.

Appendix C

Stable Virtual Camera: Generative view synthesis with diffusion models

C.1 Outline

In this Appendix, we describe in detail the following:

- Broader impact and limitations in Section C.2.
- Related work in Section C.3.
- Benchmark datasets and evaluation setup in Section C.4.
- Additional experimental results in Section C.5.
- Additional qualitative results in Section C.5.1.
- Additional quantitative results in Section C.5.2.
- Discussion on samples versus 3DGS, padding strategies, and long-trajectory artifacts in Section C.5.3.

C.2 Broader impact and limitations

Broader impact. SEVA significantly advances immersive 3D experiences by synthesizing realistic and temporally consistent views from sparse camera inputs, addressing key limitations in NVS. Inspired by James Cameron’s pioneering Virtual Camera technology—which enabled filmmakers to intuitively navigate virtual environments and visualize precise camera trajectories—our generative AI-driven model similarly allows users to create intricate, controllable camera paths without the typical complexity of dense view captures or explicit 3D reconstructions. By generalizing across arbitrary viewpoint changes and enabling temporally smooth rendering without NeRF distillation, our approach simplifies the NVS pipeline, enhancing accessibility for content creators, developers, and researchers. This facilitates applications ranging from virtual cinematography and gaming to digital heritage preservation, substantially broadening the usability and scalability of NVS.

Limitations. The performance of SEVA is constrained by the scope of its training data, resulting in reduced quality for certain types of scenes. Specifically, input images featuring humans, animals, or dynamic textures (e.g., water surfaces) typically lead to degraded outputs. Additionally, highly ambiguous scenes or complex camera trajectories pose challenges; for instance, trajectories that intersect with objects or surfaces may cause noticeable flickering artifacts. Similar issues arise for extremely irregularly shaped objects or when target viewpoints significantly diverge from the provided input viewpoints.

C.3 Related work

Novel view synthesis. While traditional NVS has been studied for nearly several decades, it has recently achieved remarkable success with the help of techniques such as NeRF [36, 222] and diffusion models [171, 223]. Using these techniques, there are broadly two ways of generating novel views : 1) estimate a 3D representation using multiple sparse input views, then regress the novel views from this intermediate representation, 2) directly estimate the novel views from the sparse input views, either in a single shot in a feed-forward manner, or in multiple sampling steps using diffusion models.

Feed-forward models. Approaches like LFNR [224] and LVSM [162] directly generate target views and leverage data-driven learning to capture 3D inductive biases. While often efficient, these methods struggle with the inherent diversity of generative NVS, limiting their capacity to model multiple plausible solutions. In contrast, our approach frames generative NVS through a diffusion perspective, enabling us to sample diverse, plausible solutions during inference, thereby addressing ambiguities and enhancing generation capacity.

Intermediate representation models. Techniques such as NeRF [36] and Gaussian Splatting [123] have made significant progress on per-scene optimization from input views by building 3D representations efficiently. Several works show that these representations can then be used to regress novel views. pixelNeRF [155] builds a NeRF from multiple input views; Splatter Image [225], pixelSplat [166], and MVSplat [167] build a 3D representation using Gaussian Splatting; LRM [226] builds a triplane representation. However, these optimization-based methods cannot creatively synthesize missing regions, and rely on tens, if not hundreds, of posed input images which limits their practicality in real-world applications.

Diffusion-based models. Our work falls within this category, where target novel views are generated in multiple steps through a denoising diffusion process [171, 223]. As mentioned earlier, existing diffusion-based methods can be divided into two main types: *image models* and *video models*.

Image models are designed to synthesize distant viewpoints [227, 228, 229]. However, these early practices only generate one viewpoint at a time, and lack multi-view consistency, often resulting in jittery and inconsistent samples when generating along a camera trajectory. Works

such as MVDream [176], SyncDreamer [230] and HexGen3D [231] generate multiple fixed views simultaneously. However, these models only generate specific views given a conditional image, not arbitrary viewpoints.

To obtain consistent 3D objects, these models necessitate NeRF distillation, either through Score Distillation Sampling (SDS) [156, 164] or directly upon completely sampled images [158, 159].

Video models can produce smooth video sequences by maintaining certain constraints relative to the input views [169]. However, they are generally limited to smaller camera motions due to the natural frame rate in video training. Some works use video diffusion models to generate 4D scenes [232]. But in those works, the video diffusion models do not contribute to the consistency of the 3D object itself, that part is handled by image-based diffusion models such as MVDream.

C.4 Benchmark

We collect 10 commonly used datasets to benchmark NVS, encompassing a diverse range of scene distributions and complexities, shown in Table C.1.

Small-viewpoint *versus* large-viewpoint NVS. In Section 4.4.1, we split NVS tasks into two categories: small-viewpoint and large-viewpoint NVS based on the disparity between \mathbf{I}^{inp} and \mathbf{I}^{gt} . Formally, for each target view, we consider the minimal distance between the CLIP [182] feature of that view and those of all input views. Averaging across all target views yields the CLIP distance, $\mathcal{D}_{\text{CLIP}}(\mathbf{I})$. Splits with $\mathcal{D}_{\text{CLIP}}(\mathbf{I}) \leq 0.11$ are grouped as small-view NVS, while those with $\mathcal{D}_{\text{CLIP}}(\mathbf{I}) > 0.11$ are grouped as large-view NVS. We concrete in Table C.1 a detailed task setup including the choice of datasets and splits (depending on which scenes from each dataset and which views from each scene are used).

Choice of scenes. We follow the choices of scenes for splits adopted from previous works. For our split, we use all scenes from the dataset without specification.

For the Tanks and Temples dataset, the 2 chosen scenes in our (O) split are TRAIN and TRUCK. For the DL3DV-140 dataset, the 10 test scenes we choose in O split are:

- 165F5AF8BFE32F70595A1C9393A6E442ACF7AF 019998275144F605B89A306557
- 341B4FF3DFD3D377D7167BD81F443BEDAFBFF0 03BF04881B99760FC0AEB69510
- 3BB3BB4D3E871D79EB71946CBAB1E3AFC7A8E3 3A661153033F32DEB3E23D2E52
- 3BB894D1933F3081134AD2D40E54DE5F0636BD 8B502B0A8561873BB63B0DCE85
- 9E9A89AE6FED06D6E2F4749B4B0059F35CA97F 848CEDC4A14345999E746F7884
- CD9C981EEB4A9091547AF19181B382698E9D9E EE0A838C7C9783A8A268AF6AEE
- D4FBEB A0168AF8FDD B2FC695881787AEDCD62F 477C7DCEC9EBCA7B8594BBD95B
- E78F8CEBD2BD93D960BFAEAC18FAC0BB2524F1 5C44288903CD20B73E599E8A81
- ED16328235C610F15405FF08711EAF15D88A05 03884F3A9CCB5A0EE69CB4ACB5
- F71AC346CD0FC4652A89AFB37044887EC3907D 37D01D1CEB0AD28E1A780D8E03.

For the WildRGBD dataset, the 20 test scenes we choose in O split are: For the WildRGBD dataset, the 20 test scenes we select for the O split are:


















	type	split	#scene	$(\mathbf{I}^{\text{inp}}, \mathbf{I}^{\text{tgt}}) \sim \mathcal{V}$	P	$\mathcal{D}_{\text{CLIP}}(\mathbf{I})$
Small-viewpoint NVS						
OmniObject3D [185]		O (dynamic orbit)	308	✓	3	0.11
GSO [186]		O (dynamic orbit)	300	✓	3	0.11
RealEstate10K [191]		D [163]	128	✓	1	0.09
		R [158]	10	✓	1	0.08
		P [166]	6474	✓	3	0.03
		V [160]	10	✓	2	0.04
LLFF [187]		R [158]	8	✓	2	0.11
DTU [188]		R [158]	15	✓	1	0.04
CO3D [189]		R [158]	20	✓	3	0.09
		V [160]	10	✓	2	0.09
WildRGB-D [190]		O_e (1/3 orbit)	20	✓	3	0.07
		O_h (full orbit)			6	0.11
Mip-NeRF360 [49]		R [158]	9	✗	6	0.11
DL3DV-140 [192]		O	10	✓	6	0.10
		L [168]	140	✓	32	0.05
Tanks and Temples [193]		V [160]	22	✓	2	0.10
		L [168]	2	✓	32	0.10
Large-viewpoint NVS						
OmniObject3D [185]		S [169] (dynamic orbit)	308	✓	1	0.16
GSO [186]		S [169] (dynamic orbit)	300	✓	1	0.18
CO3D [189]		R [158]	20	✓	1	0.15
WildRGB-D [190]		O_h (full orbit)	20	✓	1	0.19
					3	0.14
Mip-NeRF360 [49]		R [158]	9	✗	1	0.19
					3	0.13
DL3DV-140 [192]		O	10	✓	1	0.21
					3	0.12
Tanks and Temples [193]		O	2	✓	1	0.21
					3	0.18
					6	0.16
					9	0.14

Table C.1: **Statistics for NVS benchmark.** We consider 10 publicly available datasets commonly used for evaluating NVS, encompassing both object-level and scene-level data. Views from Mip-NeRF360 [49] derive from several disjoint captures following different camera trajectories, thus all views $(\mathbf{I}^{\text{inp}}, \mathbf{I}^{\text{tgt}}) \sim \mathcal{I}$. P denotes the number of input views. Depending on the disparity between \mathbf{I}^{inp} and \mathbf{I}^{tgt} , we group NVS tasks into small-viewpoint NVS (top panel) where target views are similar to input views and large-viewpoint NVS (bottom panel) where target views are more different to input views.

Method	dataset	OO3D GSO		RE10K				LLFF		DTU		CO3D		WRGBD		Mip360	DL3DV		T&T		
	split	O	O	D [163]	V [160]	P [166]	R [158]		R [158]		V [160] R [158]		O _e	O _h	R [158]	O	L [168]	V [160]	L [168]		
	P	3	3	1	1	2	1	3	1	3	1	3	3	6	6	6	32	1	32		
Regression-based models																					
Long-LRM [168]		-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0.262	-	0.375		
MVSplat [167]		0.411	0.387	0.224	0.237	0.128	0.254	0.142	0.542	0.497	0.386	0.310	0.634	0.614	0.504	0.643	0.556	0.527	0.425	0.519	0.568
DepthSplat [194]		0.404	0.372	0.217	0.245	0.119	0.236	0.177	0.530	0.465	0.369	0.304	0.618	0.603	0.499	0.530	0.534	0.481	0.404	0.462	0.528
LVSM [162]		-	-	-	-	0.098	-	-	-	-	-	-	-	-	-	-	-	-	-		
Diffusion-based models																					
MotionCtrl [195]		-	-	0.500	0.386	-	-	-	-	-	-	0.443	-	-	-	-	-	-	0.473	-	
4DiM [163]		-	-	0.302	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-		
ViewCrafter [160]		0.427	0.379	0.220	0.178	0.203	0.287	0.164	0.620	0.435	0.485	0.272	0.324	0.513	0.324	0.639	0.464	0.558	-	0.283	-
SEVA		0.049	0.041	0.194	0.231	0.061	0.308	0.073	0.389	0.181	0.316	0.158	0.318	0.278	0.215	0.237	0.319	0.232	0.155	0.354	0.236

(a) LPIPS↓

Method	dataset	OO3D GSO		RE10K				LLFF		DTU		CO3D		WRGBD		Mip360	DL3DV		T&T		
	split	O	O	D [163]	V [160]	P [166]	R [158]		R [158]		V [160] R [158]		O _e	O _h	R [158]	O	L [168]	V [160]	L [168]		
	P	3	3	1	1	2	1	3	1	3	1	3	3	6	6	6	32	1	32		
Regression-based models																					
Long-LRM [168]		-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0.775	-	0.590		
MVSplat [167]		0.554	0.621	0.788	0.769	0.869	0.812	0.857	0.283	0.358	0.576	0.624	0.403	0.370	0.405	0.368	0.312	0.487	0.512	0.394	0.314
DepthSplat [194]		0.636	0.689	0.801	0.745	0.887	0.820	0.824	0.299	0.396	0.601	0.638	0.429	0.402	0.436	0.417	0.324	0.513	0.564	0.413	0.359
LVSM [162]		-	-	-	-	0.906	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
Diffusion-based models																					
MotionCtrl [195]		-	-	0.267	0.587	-	-	-	-	-	-	0.502	-	-	-	-	-	-	0.384	-	
4DiM [163]		-	-	0.463	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
ViewCrafter [160]		0.538	0.647	0.792	0.798	0.710	0.806	0.830	0.146	0.454	0.542	0.671	0.641	0.483	0.465	0.376	0.354	0.469	-	0.563	-
SEVA		0.935	0.942	0.615	0.693	0.847	0.700	0.892	0.384	0.602	0.652	0.750	0.585	0.647	0.670	0.646	0.395	0.546	0.661	0.437	0.505

(b) SSIM↑

Table C.2: **LPIPS↓ (top) and SSIM↑ (bottom) on small-viewpoint set NVS.** For all results with $P = 1$, we sweep the unit length for camera normalization due to the model’s scale ambiguity. O_e and O_h denote the easy and hard split of our split. Underlined numbers are run by us using the officially released code.

- BALL/SCENE_563
- APPLE/SCENE_234
- MICROWAVE/SCENE_143
- SCISSOR/SCENE_489
- BUCKET/SCENE_294
- KEYBOARD/SCENE_092
- SHOE/SCENE_868
- KETTLE/SCENE_399
- CLOCK/SCENE_524
- HAT/SCENE_039
- BACKPACK/SCENE_264
- SCISSOR/SCENE_958
- TRUCK/SCENE_232
- HANDBAG/SCENE_575

Method	dataset	OO3D		GSO	CO3D	WRGBD		Mip360		DL3DV		T&T			
	split	S [169]	S [169]	R [158]		O _h		R [158]		O		O			
	<i>P</i>	1	1	1	1	3	1	3	1	3	1	3	6	9	
SV3D [169]		0.158	0.140	-	-	-	-	-	-	-	-	-	-	-	-
DepthSplat [194]		0.610	0.543	0.756	0.732	0.588	0.691	0.491	0.580	0.405	0.774	0.706	0.611	0.487	
CAT3D [159]		-	-	-	-	-	-	0.488	-	-	-	-	-	-	-
ViewCrafter [160]		0.634	0.559	0.789	0.775	0.603	0.723	0.540	0.616	0.576	0.755	0.671	0.604	0.546	
SEVA		0.160	0.137	0.445	0.423	0.289	0.573	0.364	0.484	0.316	0.571	0.463	0.387	0.328	

(a) **LPIPS**↓

Method	dataset	OO3D		GSO	CO3D	WRGBD		Mip360		DL3DV		T&T			
	split	S [169]	S [169]	R [158]		O _h		R [158]		O		O			
	<i>P</i>	1	1	1	1	3	1	3	1	3	1	3	6	9	
SV3D [169]		0.850	0.880	-	-	-	-	-	-	-	-	-	-	-	-
DepthSplat [194]		0.549	0.612	0.385	0.234	0.335	0.206	0.291	0.349	0.452	0.304	0.315	0.326	0.367	
CAT3D [159]		-	-	-	-	-	-	0.294	-	-	-	-	-	-	-
ViewCrafter [160]		0.463	0.575	0.277	0.225	0.321	0.199	0.264	0.323	0.400	0.312	0.328	0.337	0.343	
SEVA		0.857	0.873	0.536	0.505	0.603	0.282	0.377	0.360	0.480	0.342	0.385	0.427	0.452	

(b) **SSIM**↑

Table C.3: **LPIPS**↓ (top) and **SSIM**↑ (bottom) on large-viewpoint set NVS. For all results with $P = 1$, we sweep the unit length for camera normalization due to the model’s scale ambiguity. Underlined numbers are run by us using the officially released code.

Method	small-viewpoint				large-viewpoint	Method	small-viewpoint				large-viewpoint
	RE10K	LLFF	DTU	CO3D	Mip360		RE10K	LLFF	DTU	CO3D	Mip360
ZipNeRF [196]	0.332	0.373	0.383	0.652	0.705	ZipNeRF [196]	0.774	0.574	0.601	0.496	0.271
ZeroNVS [164]	0.422	0.512	0.223	0.566	0.680	ZeroNVS [164]	0.675	0.359	0.716	0.581	0.316
ReconFusion [158]	0.144	0.203	0.124	0.398	0.585	ReconFusion [158]	0.910	0.724	0.875	0.662	0.358
CAT3D [159]	0.132	0.181	0.121	0.351	0.515	CAT3D [159]	0.917	0.731	0.844	0.666	0.377
SEVA	0.078	0.164	0.107	0.256	0.435	SEVA	0.961	0.735	0.867	0.702	0.454

(a) **LPIPS**↓(b) **SSIM**↑

Table C.4: **LPIPS**↓ (top) and **SSIM**↑ (bottom) on 3DGS renderings for set NVS. Results are reported on the ReconFusion [158] split with $P = 3$.

Method	split	small-viewpoint					large-viewpoint		
		V [160]					O		
	dataset	RE	CO3D	T&T	RE	DTU	WR	DL	T&T
MotionCtrl [195]		0.386	0.443	0.473	-	-	-	-	-
DepthSplat [194]		0.224	0.532	0.415	0.134	0.253	0.452	0.572	0.685
ViewCrafter [160]		0.178	0.283	0.324	0.120	0.187	0.346	0.566	0.674
SEVA		0.231	0.318	0.353	0.079	0.159	0.284	0.329	0.514
SEVA (+ temp.)		0.228	0.312	0.356	0.078	0.156	0.280	0.328	0.510

(a) **LPIPS**↓

Method	split	small-viewpoint					large-viewpoint		
		V [160]					O		
	dataset	RE	CO3D	T&T	RE	DTU	WR	DL	T&T
MotionCtrl [195]		0.587	0.502	0.384	-	-	-	-	-
DepthSplat [194]		0.723	0.486	0.408	0.844	0.723	0.447	0.539	0.497
ViewCrafter [160]		0.798	0.641	0.563	0.868	0.739	0.464	0.523	0.456
SEVA		0.693	0.585	0.437	0.890	0.756	0.613	0.475	0.363
SEVA (+ temp.)		0.695	0.590	0.436	0.891	0.760	0.616	0.476	0.369

(b) **SSIM**↑

Table C.5: **LPIPS**↓ (top) and **SSIM**↑ (bottom) on trajectory NVS. For the V [160] split, $P = 1$ with unit length swept; for the O split, $P = 3$. RE, WR, and DL denote RE10K, WRGBD, and DL3DV, respectively. Underlined numbers are run by us using the officially released code.

- PINEAPPLE/SCENE_182
- TRAIN/SCENE_033
- REMOTE_CONTROL/SCENE_453
- BOWL/SCENE_673
- TV/SCENE_062

Full test scenes are chosen for the remaining datasets.

Choice of input and target views. We follow the same setup for splits adopted from previous works, by using the same set of input and target views. For split defined ourselves, we detail the choice of views as below. For the WildRGB-D [190] dataset, which consists of scenes captured while orbiting around an object, we define two splits with different difficulty levels. O_e represents the easy set, where each scene is trimmed to one-third of the original sequence (i.e., approximately 120 degrees of rotation). In contrast, O_h corresponds to the hard set, using the full original sequence (i.e., approximately 360 degrees of rotation). We first uniformly subsample 21 frames from the scene, and randomly choose P frames as input views with the remaining frames as target views. For

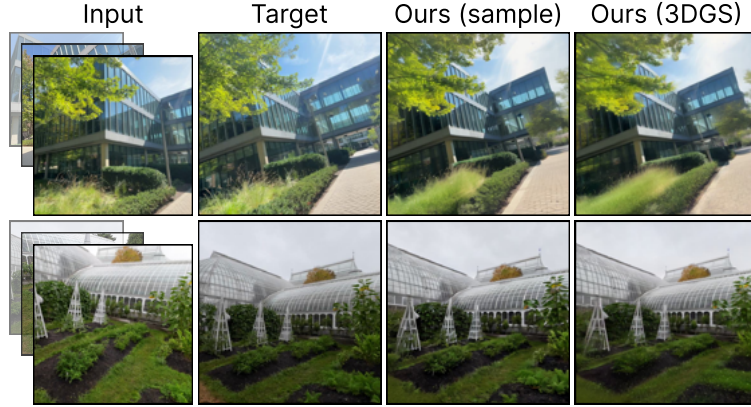


Figure C.1: **3DGS versus samples.** The model generates consistent renderings that closely resemble those from 3DGS [123], with minimal perceptual differences.

each scene from DL3DV-140 [192] and Tanks and Temples [193] datasets, we selected target frames by using every 8th frame of the original sequence. For the remaining frames, we applied K -means clustering ($K = 32$) on a 6-dimensional vector formed by concatenating the camera translation and the unit vector of the camera direction.

C.5 Additional experiments

C.5.1 Qualitative results

We provide additional single-view conditioning sampling results with a diverse set of camera motions and effects on a variety of image prompts: a text-prompted object-centric scene (Figure C.3), a text-prompted scene (Figure C.4), a real-world object-centric scene (Figure C.5), and a real-world scene (Figure C.6). SEVA demonstrates strong generalization, adapting robustly across a wide range of scenarios.

C.5.2 Quantitative results

We provide additional quantitative evaluation results of our model against baselines on set NVS and trajectory NVS, measured using LPIPS [197] and SSIM [9], in Tables C.3, C.2, C.4, and C.5.

C.5.3 Discussion

Samples versus 3DGS. We compare our samples to their 3DGS distillation on the O split of DL3DV, shown in C.1. First, we note that our samples contain plausible hallucinations when uncertainty is high (first row, building on the right). Second, we note that our 3DGS renderings

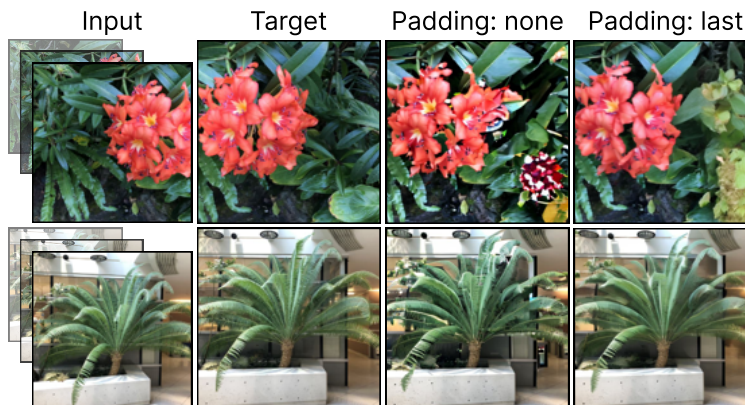


Figure C.2: **Padding.** Padding the last elements within one forward reduces artifacts compared to changing T .

remain sharp and are close to the samples. These results suggest that our samples are 3D consistent enough.

Padding T when $P + Q < T$. We analyze the effect of different padding strategies when $P + Q < T$ in Figure C.2. We observe that zero-shot generalization of T to $P + Q$ without padding leads to abnormal color overflows. This is in stark contrast to the excessive blurriness observed when generalizing T when $P + Q \gg T$ in sparse-view regime (section 4.4.5). Hypothetically, sampling with a T unseen during training induces a distribution shift in the attention scores [233]. Specifically, a smaller T sharpens the attention distribution, whereas a larger T disperses it. This shift may explain the contrasting behavior observed when using the model for sampling. Training the model with a dynamically varying T during training could mitigate this issue by exposing the model to a broader range of attention score distributions, improving generalization across different T .

Artifacts on long-trajectory NVS. We observe that the results tend to become increasingly saturated, particularly when the target views are far from the input views and share no content overlap, such as in open-ended exploration and navigation. The concurrent work [234] explores the concept of Diffusion Forcing [235] for long video rollouts, achieving high-generation quality. Applying diverse noise to the input views during training can be beneficial, as it enables the refinement of high-level details in all anchor views within the memory bank during sampling, thereby mitigating the accumulation of saturation. We leave this for future work.



Figure C.3: **Diverse camera motions and effects.** Single-view conditioning with a text-prompted object-centric scene. The image is generated using SD 3.5 [236] with the text prompt, “A cute firefly dragon in its natural habitat.”

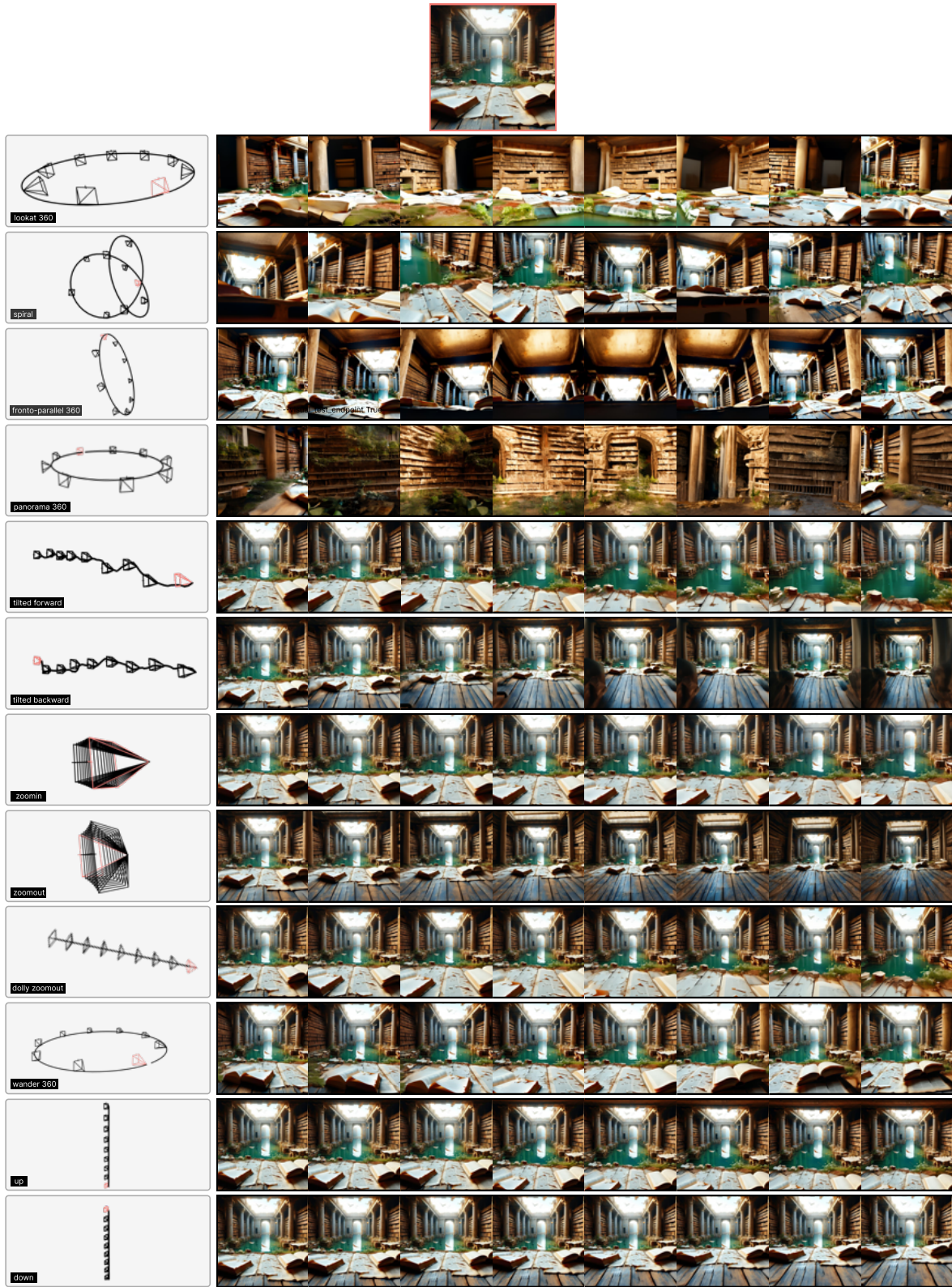


Figure C.4: **Diverse camera motions and effects.** Single-view conditioning with a text-prompted scene. The image is generated using SD 3.5 [236] with the text prompt, “Wide view of the interior of the famed Library of Alexandria, elegantly set behind a time-worn wreckage by a lake. ”



Figure C.5: **Diverse camera motions and effects.** Single-view conditioning with a real-life object-centric scene.

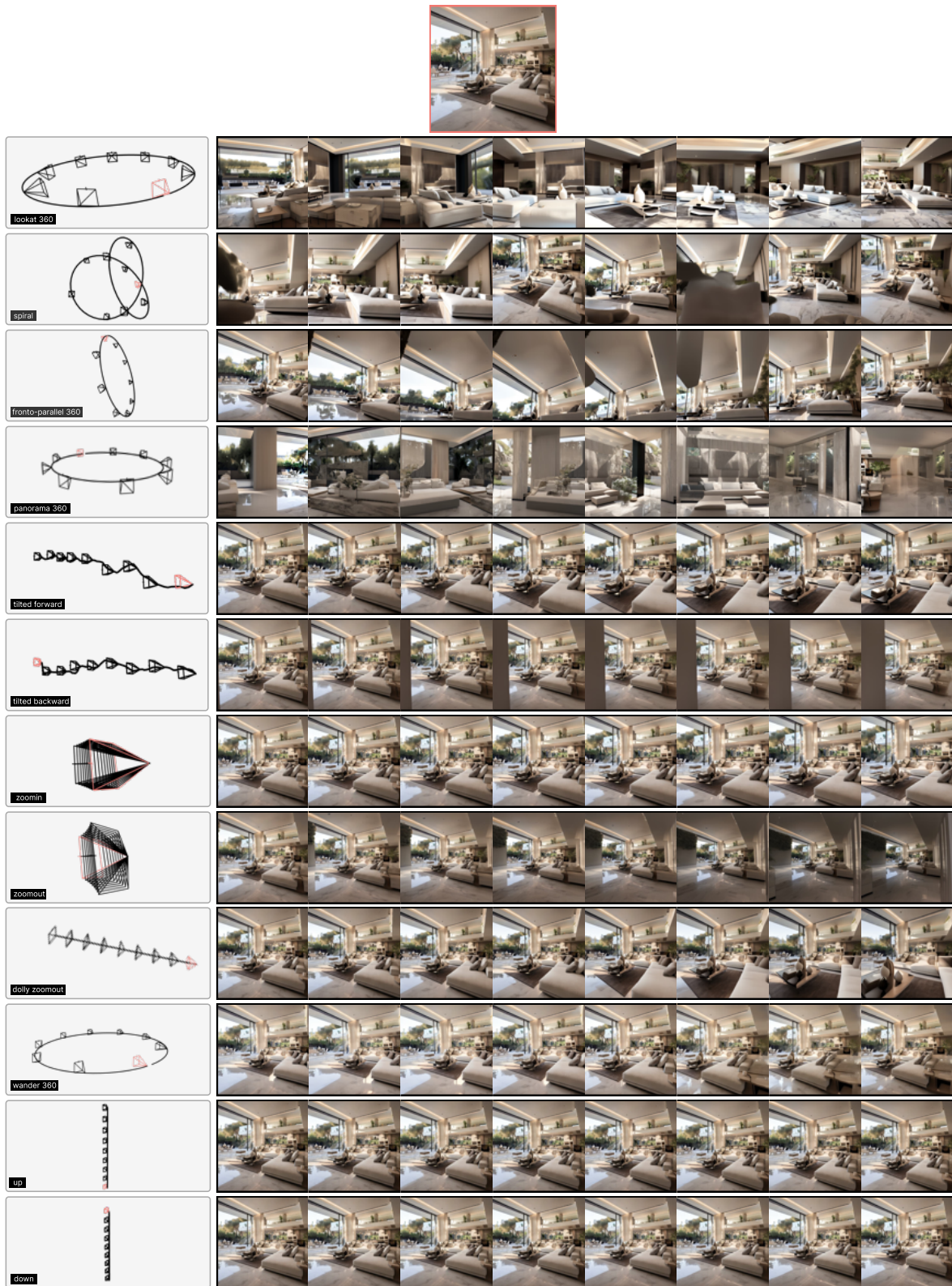


Figure C.6: **Diverse camera motions and effects.** Single-view conditioning with a real-life scene.